Xanadu Build or modify applications

Last updated: 05/04/2026

Some examples and graphics depicted herein are provided for illustration only. No real association or connection to ServiceNow products or services is intended or should be inferred.

Please read the ServiceNow Website Terms of Use at www.servicenow.com/terms-of-use.html

Company Headquarters
2225 Lawson Lane
Santa Clara, CA 95054
United States
(408) 501-8550

servicenow

Table of Contents

Building applications

Learn how to become an application developer using ServiceNow AI Platform low-code tools. Start with what you know and use a library of reusable components and published applications to modernize your legacy processes.

Get started

Learning about creating applications

Learn basic information about application development.

Phase 1: Planning your application

Plan how your application will work.

Phase 2: Developing your application

Add components and content to your application.

Phase 3: Testing and debugging your application

Verify that the application meets your business requirements.

Phase 4: Deploying your application

Deploy your application to your production environment.

Phase 5: Maintaining your application

Review the status of your application and make changes as needed.

What's new

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity.

App development phases

Learning about creating applications

Decide whether you want to build a new application or extend an existing application. Check the ServiceNow Store and the ServiceNow Community for existing solutions.

Before building your first application, you may want to learn some basic information about application development. This phase is optional, and you can complete it at any time while you work on other phases.

- How ServiceNow AI Platform is made up of tables and records. Learn how to convert a spreadsheet into record data.
- How to Get a development instance to practice creating applications.
- How to find out Licensing for which application features require a subscription.
- How to contact Support for developers to ask questions about application development.

Phase 1: Planning your application

The application development process starts with planning. Consider how the application will work, who will use it, and how it will improve your users' experience. Your application plan should answer the following questions:

- What are the goals, objectives, and outputs of your application?
- Who uses your application?
- Who has access to parts of the application?
- What tasks do people complete with your application?
- Where does the data come from?
- How do people interact with your application?
- What processes must the application support?
- What UI experience does the application use?
- Is there an existing application available on the ServiceNow Store or the ServiceNow Community that you can use or extend?
- What subscriptions does your application require?

Phase 2: Developing your application

During the development phase, you add the components and content of your application. Most applications consist of the following:

Data

Information is stored in your application via tables that you configure. For example, employee phone numbers or office locations.

Experience

Experiences are graphical interfaces that your users interact with. For example, you can create a portal where users find information, submit requests, or complete business tasks.

Logic and automation

Automate all the work in your application by adding logic and automation. For example, you can build a flow that sends a notification to the admin when someone makes a request.

Security

Configure roles and access controls to limit who can use your application. For example, you can restrict access to application data to users who have a specific role.

Choose a builder that matches the type of user experience that your application provides.

- See Build apps using App Engine Studio to learn about low-code development.
- See Build workflows to learn about creating automation with Workflow Studio or Playbooks.
- See Builder library to learn about specialized application resources.

Phase 3: Testing and debugging your application

Verify that the application meets your business requirements. Your testing should cover the following elements:

- Record operations, such as create, read, update, and delete.
- User interface elements, such as views and UI policies.
- Runtime operations, such as business rules and event script actions.

Phase 4: Deploying your application

After successfully testing an application, deploy it to your production environment with your builder tool.

- See Managing app development using the App Engine Management Center to learn about the App Engine Management Center
- See ServiceNow application repository to learn about the Application Repository.
- See System update sets to learn about classic deployment using update sets.

Phase 5: Maintaining your application

Use your Phase 2 builder tool to update and modify your application. Use your Phase 3 testing tool to verify that your application still functions properly.

Applications and features

- Learning about developing on the ServiceNow AI Platform
- Planning your application
- Developing your application
- Testing and debugging applications
- Deploying applications
- Maintaining your application

Learning about developing on the ServiceNow AI Platform

Using the ServiceNow Creator Workflow products, you can create intuitive experiences that run on the powerful ServiceNow AI Platform.

In this document

Use this document to discover how to:

- Find all ServiceNow developer products
- Understand low-code and pro-code development
- Understand ServiceNow licenses
- Get support as a developer
- Understand the various ServiceNow user interfaces
- Learn the basics of being a developer

Please use the reader feedback feature to suggest additional topics.

What's new

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity.

Find ServiceNow developer products quickly

Find the ServiceNow developer product that can help you build amazing products on the ServiceNow AI Platform.

Find the developer product you need

Use the following tools to find ServiceNow developer products:

- ServiceNow Widget that guides you to developer products
- ServiceNow Table that lists developer products

If you're still not sure where to start, contact your sales representative. They can help you figure out what problems you're trying to solve and how to solve them.

Finding ServiceNow developer products

Use the following widget to find ServiceNow developer products that help you create applications that run on the powerful ServiceNow AI Platform.

Find the products that help you build applications

To find the ServiceNow developer products and information that are right for you, select one of the following tiles.

I am comfortable coding

Build ServiceNow apps with custom code.

I do not code or write scripts

Build ServiceNow apps within a low code environment.

Pro-code tools: What you can do to build apps

Select an option below.

Build apps

Learn about developing on the ServiceNow platform.

Manage my apps

Customize your ServiceNow applications.

Engage with my customers

Learn how you can communicate with your customers using the ServiceNow platform.

Create reports

Create reports with your data on the ServiceNow platform.

Learn about the ServiceNow platform

See what you can do with the ServiceNow platform.

Get help

Find resources to help you with the ServiceNow platform.

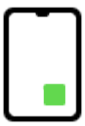
Build apps with pro-code tools

Use the ServiceNow platform to build apps.

ServiceNow Studio

Build custom applications using integrated builders and tools in one place.

Mobile App Builder



Build ServiceNow mobile applications.

VS Code Extensions

Edit your ServiceNow applications in Visual Studio Code with the help of the ServiceNow Extensions for VS Code.

AES

Build apps in a structured environment for fast development and deployment.

Manage apps with pro-code tools

Manage your ServiceNow apps.

Customize my apps

Customize your apps on the ServiceNow platform.

Automate my apps

Automate your ServiceNow applications.

Secure my apps

Secure your ServiceNow applications.

Test my apps

Test your apps on the ServiceNow platform.

Deploy my apps

Deploy your apps with the ServiceNow platform.

Maintain my apps

Maintain your apps on the ServiceNow platform.

Customize my apps with pro-code tools

Customize your ServiceNow apps.

UI Builder

Learn how to create custom experiences with the ServiceNow platform.

Table Builder

Customize your data with Table Builder.

Workspace Builder

Customize your data with Workspace Builder.

Scripts

Use custom scripts with the ServiceNow platform.

Automate apps with pro-code tools

Automate your ServiceNow apps.

Automation Discovery

Identify automation opportunities for your workflows on the ServiceNow platform.

Flows in Workflow Studio

Enables process owners to automate work with the ServiceNow platform.

Decision Builder

Enable developers to decouple decision logic from their code by creating and maintaining decision rules.

Playbooks

Enable process owners to author cross-enterprise workflows and create a single, unified process.

Business rules

Use business rules to accomplish tasks like automatically changing values in form fields when certain conditions are met.

Secure apps with pro-code tools

Secure your ServiceNow apps.

App Engine Studio roles and permissions

Control who is permitted to use or edit your apps.

Edge Encryption

Encrypt sensitive data on your company premises before sending it over the Internet.

UI Policies

Dynamically change the behavior of information on a form and control custom process flows for tasks.

Roles

Control access to features and capabilities in applications and modules.

Access Control

Control which Customer Service and Support employees can access your instance, and when.

Users and Groups

Manage the individuals who can access your instance by defining them as users in the system.

Test apps with pro-code tools

Test your ServiceNow apps.

Automated Test Framework

Create and run automated tests to confirm that your instance works after making a change.

Test Management

Streamline the management of testing processes to help you deliver software products more efficiently and with fewer errors.

Script Debugger

Debug scripts using session logs and ServiceNow AI Platform debugging tools such as a walk-through script debugger and error messages that display in the UI.

Script Tracer

Filter your debugging search to quickly narrow down script problems.

Session Log

View and download the session log for business rules, script includes, and a custom UI.

User impersonation

Impersonate other authenticated users for testing purposes and view impersonation logs.

NOW Command Line Interface (CLI)

Perform instance operations from your local system with a command-line interface.

Deploy apps with pro-code tools

Deploy your ServiceNow apps.

The Next Experience

App Engine Management Center

Track and manage your App Engine Studio (AES) requests, deployments, applications, and collaborative developers using the App Engine Management Center (AEMC) in your production instance.

Core UI Tech Stack

Application Repository

Learn about developing on the ServiceNow platform.

Cloud Provisioning and Governance

The ServiceNow® Cloud Provisioning and Governance application provides a single interface to access cloud resources, publish cloud offerings to a catalog, and manage the usage of those resources.

Maintain apps with pro-code tools

Maintain your ServiceNow apps.

The Next Experience

System Update Sets

Allow administrators to group a series of changes into a named set and then move them as a unit to other systems for testing or deployment.

Service Mapping

The ServiceNow® Service Mapping application discovers all application services in your organization and builds a comprehensive map of all devices, applications, and configuration profiles used in these application services.

Engage with customers with pro-code tools

Engage with your customers using ServiceNow apps.

Chat bot

Virtual Agent Designer

Create and manage virtual agent topics using the ServiceNow platform.

Virtual Agent Dashboard

The Conversational Analytics Dashboard helps you improve Virtual Agent interactions with users by providing deep insights into conversational data.

Artificial intelligence

Natural Language Understanding Workbench

Use the NLU Workbench to create morphological representations of human language.

Platform Analytics

Optimize processes and increase productivity with Platform Analytics, virtual agents, and machine learning.

Predictive Intelligence

Provide a layer of artificial intelligence that empowers features and capabilities across ServiceNow applications to provide better work experiences

Reporting on data from pro-code apps

Most applications that you create have some level of reporting requirements. Reports should be actionable to drive change.

"Reporting" generally refers to showing the data inside facts tables like Incident [incident]. You can also create key performance indicators (KPIs) to track changes in this data over time, through the Performance Analytics application.

Currently the ServiceNow AI Platform® is in transition between two user interfaces for showing this information:

- The older Core UI technology. This UI includes the Reporting application, which only shows data directly from tables, and PA Widgets, which show data from Performance Analytics indicators. Both reports and PA widgets can be placed on Core UI responsive dashboards. For more information, see Reporting, dashboards, and Performance Analytics in the Core UI.
- The newer Platform Analytics technology. This UI includes Data Visualizations, which let you report on data from any source. These visualizations can be placed on Platform Analytics dashboards, along with Platform Analytics filters. For more information, see Platform Analytics experience.

All Platform Analytics objects are rooted in the Next Experience UI Framework and are available to developers. However, a non-developer can still build their own objects through the Platform Analytics experience, without using UI Builder. For more information, see Platform Analytics experience.

Note:
Although "reporting" is a general term, this documentation usually uses "report" and "reporting" to refer to the Core UI Reporting application and "visualization" or "visualize" to refer to Platform Analytics data visualizations, for disambiguation.

Follow these guidelines when building reports or data visualizations:

- Creating reports or visualizations on large tables can negatively impact performance. Be sure to filter by a date range or other limiting criteria rather than showing all records on the table.
- Grouping by fields that contain many possible values can negatively impact performance.
- If loading a report or visualization gives a Long running transaction timer message, consider adding more data filters to reduce the load time.
- If you need to export a report, data visualization, or dashboard on a regular basis, schedule the export and email.

In Platform Analytics, you have several possibilities for showing multiple data visualizations on one page:

- Create the data visualizations and the dashboards entirely inside the Platform Analytics experience. This approach does not require a developer role or special technical knowledge, and probably should be explored before you try a more complex solution.
- Create data visualization components in a generic UI Builder page, along with filters and other components. This approach gives you the most freedom as a developer, but also requires the most configuration.
- Create the data visualizations and the dashboards inside the Platform Analytics experience, but then place the dashboards inside UI Builder pages using the Dashboard page template. This approach lets you use the convenience of the Platform Analytics experience to create dashboards, data visualizations, and filters in your own experiences/workspaces. It also gives you the freedom to customize the page configuration partially. For more information, see Creating Platform Analytics pages in your own workspace.
- Create a technical dashboard and populate it inside UI Builder. This approach is almost the same as creating your own UI Builder page from scratch, but the dashboard is available in the dashboard library, has dashboard details, and can be shared like other dashboards. For more information, see Technical dashboards.

Low-code and no-code tools: What you can do to build apps

When building low-code apps, you can customize them, create reports on their usage, and more.

Build apps

Learn about developing on the ServiceNow platform.

Manage my apps

Customize your ServiceNow applications.

Engage with my customers

Learn how you can communicate with your customers using the ServiceNow platform.

Create reports

Create reports with your data on the ServiceNow platform.

Learn about the ServiceNow platform

See what you can do with the ServiceNow platform.

Get help

Find resources to help you with the ServiceNow platform.

Build apps with low-code and no-code tools

Use the ServiceNow platform to build apps.

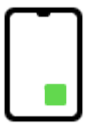
Creator Studio

Build request-fulfill apps with no coding experience required.

AES

Build apps in a structured environment for fast development and deployment.

Mobile App Builder



Build ServiceNow mobile applications.

Manage apps with low-code tools

Manage your ServiceNow apps.

Customize my apps  Customize your apps on the ServiceNow platform.	**Automate my apps**  Automate your ServiceNow applications.	**Secure my apps**  Secure your ServiceNow applications.
Test my apps  Test your apps on the ServiceNow platform.	**Deploy my apps**  Deploy your apps with the ServiceNow platform.	**Maintain my apps**  Maintain your apps on the ServiceNow platform.

Customize my apps with low-code tools

Customize your ServiceNow apps.

UI Builder  Learn how to create custom experiences with the ServiceNow platform.	**Table Builder**  Customize your data with Table Builder.
Workspace Builder  Customize your data with Workspace Builder.	

Automate apps with low-code tools

Automate your ServiceNow apps.

Automation Discovery

Identify automation opportunities for your workflows on the ServiceNow platform.

Flows in Workflow Studio

Enables process owners to automate work with the ServiceNow platform.

Decision Builder

Enable developers to decouple decision logic from their code by creating and maintaining decision rules.

Playbooks

Enable process owners to author cross-enterprise workflows and create a single, unified process.

Business rules

Use business rules to accomplish tasks like automatically changing values in form fields when certain conditions are met.

Secure apps with low-code tools

Secure your ServiceNow apps.

App Engine Studio roles and permissions

Control who is permitted to use or edit your apps.

Edge Encryption

Encrypt sensitive data on your company premises before sending it over the Internet.

UI Policies

Dynamically change the behavior of information on a form and control custom process flows for tasks.

Roles

Control access to features and capabilities in applications and modules.

Access Control

Control which Customer Service and Support employees can access your instance, and when.

Users and Groups

Manage the individuals who can access your instance by defining them as users in the system.

Test apps with low-code tools

Test your ServiceNow apps.

Automated Test Framework

Create and run automated tests to confirm that your instance works after making a change.

Test Management

Streamline the management of testing processes to help you deliver software products more efficiently and with fewer errors.

Script Debugger

Debug scripts using session logs and ServiceNow AI Platform debugging tools, such as a walk-through script debugger and error messages that display in the UI.

Script Tracer

Filter your debugging search to quickly narrow down script problems.

Session Log

View and download the session log for business rules, script includes, and a custom UI.

User impersonation

Impersonate other authenticated users for testing purposes and view impersonation logs.

NOW Command Line Interface (CLI)

Perform instance operations from your local system with a command-line interface.

Deploy apps with low-code tools

Deploy your ServiceNow apps.

The Next Experience

App Engine Management Center

Track and manage your App Engine Studio (AES) requests, deployments, applications, and collaborative developers using the App Engine Management Center (AEMC) in your production instance.

Core UI Tech Stack

Application Repository

Learn about developing on the ServiceNow platform.

Cloud Provisioning and Governance

The ServiceNow® Cloud Provisioning and Governance application provides a single interface to access cloud resources, publish cloud offerings to a catalog, and manage the usage of those resources.

Maintain apps with low-code tools

Maintain your ServiceNow apps.

The Next Experience

System Update Sets

Allow administrators to group a series of changes into a named set and then move them as a unit to other systems for testing or deployment.

Service Mapping

The ServiceNow® Service Mapping application discovers all application services in your organization and builds a comprehensive map of all devices, applications, and configuration profiles used in these application services.

Engage with customers with low-code tools

Engage with your customers using ServiceNow apps.

Chat bot

Virtual Agent Designer

Create and manage virtual agent topics using the ServiceNow platform.

Virtual Agent Dashboard

The Conversational Analytics Dashboard helps you improve Virtual Agent interactions with users by providing deep insights into conversational data.

Artificial intelligence

Natural Language Understanding Workbench	Platform Analytics	Predictive Intelligence
Use the NLU Workbench to create morphological representations of human language.	Optimize processes and increase productivity with Platform Analytics, virtual agents, and machine learning.	Provide a layer of artificial intelligence that empowers features and capabilities across ServiceNow applications to provide better work experiences

Reporting on data from low-code apps

Most applications that you create have some level of reporting requirements. Reports should be actionable to drive change.

"Reporting" generally refers to showing the data inside facts tables like Incident [incident]. You can also create key performance indicators (KPIs) to track changes in this data over time, through the Performance Analytics application.

Currently the ServiceNow AI Platform® is in transition between two user interfaces for showing this information:

- The older Core UI technology. This UI includes the Reporting application, which only shows data directly from tables, and PA Widgets, which show data from Performance Analytics indicators. Both reports and PA widgets can be placed on Core UI responsive dashboards. For more information, see Reporting, dashboards, and Performance Analytics in the Core UI.
- The newer Platform Analytics technology. This UI includes Data Visualizations, which let you report on data from any source. These visualizations can be placed on Platform Analytics dashboards, along with Platform Analytics filters. For more information, see Platform Analytics experience.

  All Platform Analytics objects are rooted in the Next Experience UI Framework and are available to developers. However, a non-developer can still build their own objects through the Platform Analytics experience, without using UI Builder. For more information, see Platform Analytics experience.

Note:
Although "reporting" is a general term, this documentation usually uses "report" and "reporting" to refer to the Core UI Reporting application and "visualization" or "visualize" to refer to Platform Analytics data visualizations, for disambiguation.

Follow these guidelines when building reports or data visualizations:

- Creating reports or visualizations on large tables can negatively impact performance. Be sure to filter by a date range or other limiting criteria rather than showing all records on the table.
- Grouping by fields that contain many possible values can negatively impact performance.

- If loading a report or visualization gives a Long running transaction timer message, consider adding more data filters to reduce the load time.
- If you need to export a report, data visualization, or dashboard on a regular basis, schedule the export and email.

In Platform Analytics, you have several possibilities for showing multiple data visualizations on one page:

- Create the data visualizations and the dashboards entirely inside the Platform Analytics experience. This approach does not require a developer role or special technical knowledge, and probably should be explored before you try a more complex solution.
- Create data visualization components in a generic UI Builder page, along with filters and other components. This approach gives you the most freedom as a developer, but also requires the most configuration.
- Create the data visualizations and the dashboards inside the Platform Analytics experience, but then place the dashboards inside UI Builder pages using the Dashboard page template. This approach lets you use the convenience of the Platform Analytics experience to create dashboards, data visualizations, and filters in your own experiences/workspaces. It also gives you the freedom to customize the page configuration partially. For more information, see Creating Platform Analytics pages in your own workspace.
- Create a technical dashboard and populate it inside UI Builder. This approach is almost the same as creating your own UI Builder page from scratch, but the dashboard is available in the dashboard library, has dashboard details, and can be shared like other dashboards. For more information, see Technical dashboards.

What ServiceNow product solves your problem

Use this table to find ServiceNow products that help solve your problem.

I want to	Using	ServiceNow product or topic
Get an overview of developing		• What is ServiceNow? • Get a development instance • Demos • How the ServiceNow AI Platform works • Licensing • What is low-code development?
Get help developing applications		• Getting started on the ServiceNow AI Platform • ServiceNow developer community • Customer support • Knowledge base • Customer Success Center • ServiceNow training

I want to	Using	ServiceNow product or topic
		• Additional support • Known error portal
Plan to create an application		• Delegated development • Delegated development in Creator Studio • Delegated development in App Engine Studio • Source control • Common Service Data Model
Create an application	• No code • Next Experience	• Creator Studio • App Engine Studio
	• Coding and scripting • Core UI tech stack • Forms and lists	• ServiceNow Studio • ServiceNow Extensions for Visual Studio Code • REST APIs • Guided Application Creator • ServiceNow AI Platform forms, fields, and lists
	Mobile platforms	• Mobile App Builder • Mobile Card Builder
Customize an application	UI pages	• UI Builder • Workspace Builder (in App Engine Studio only)
	Data, tables	Table Builder
	Scripts	• No-code scripting (Workflow Studio - Building custom actions) • ServiceNow Studio • Code editor • Client-side scripting • Server-side scripting • Script Debugger • ServiceNow CLI

I want to	Using	ServiceNow product or topic
	Service Portals	• Create and edit a page using the Service Portal Designer • Service Portal widgets • Configuring Next Experience themes and preferences
Add automation		• Automation Discovery • Workflow Studio • Exploring decision tables • Playbooks • Journey designer • Classic Business rules
Integrate my applications with ServiceNow	Automation and flows	• Integration Hub • Flows in Workflow Studio
	APIs	• REST API reference • Server API reference • Client API reference • Browse APIs by product • Workflow Studio - Building custom actions • Integration Hub Remote Process Sync
Secure an application	No code (App Engine Studio only)	Configure AES personas and roles
	Coding and scripting, Core UI tech stack	• Edge Encryption • UI policies • Managing roles • ServiceNow® access control • User administration
Engage customers	Customize chat bot dialogs	• Getting started with Virtual Agent Designer • Conversational Analytics Dashboard

I want to	Using	ServiceNow product or topic
	Artificial intelligence	• Analyze and optimize business processes NOW Intelligence • Predictive Intelligence
Test an application	For App Engine Studio apps	Automated Test Framework
	Coding, Core UI tech stack	• Automated Test Framework • Test Management applications • Script Debugger • Script Tracer • Session Log • Impersonate a user • ServiceNow CLI
Deploy an application	No code (App Engine Studio only)	App Engine Management Center
	Coding, Core UI tech stack	• ServiceNow application repository • Cloud Provisioning and Governance
Maintain an application		• Maintaining and monitoring the ServiceNow AI Platform • Maintaining and monitoring the ServiceNow AI Platform System update sets • Service Mapping • Instance Data Replication • MetricBase
Create reports		• CMDB 360 view in CMDB Workspace • Analyze and optimize business processes - Performance Analytic Reports • Analytics Hub • Analytics Center • Conversational Analytics Dashboard
Copy an instance		Instance Data Replication
Personalize your experience		Personalize your experience

Programming basics

Software development usually encompasses a standard set of steps. If you're new to developing applications, read on.

Your job as a developer

If you're new to developing applications, if you don't even know how to code, ServiceNow has a low-code development platform called Creator Studio that makes creating a basic request-fulfill application possible for you. Development might be new waters for you to swim in but you will find the low-code tools easy to use. So, don't be shy about developing an app!

Software developers do many things but what they have in common is solving problems using computer code. ServiceNow offers coding platforms that require no coding experience, platform coding experience, and expert coding experience. The less coding required, the more work the coding platform does for you. So, even experienced software developers will appreciate no-, low-, and platform-level coding products, such as Creator Studio and ServiceNow Studio.

Software development life cycle

The common stages of developing an application are:

1. Define the problem.
2. Plan the solution.
3. Code the solution.
4. Test the application.
5. Deploy the application.
6. Document the application.

These steps are commonly referred to as the software development life cycle. ServiceNow developer documentation is grouped according to these steps.

Do you really need to create a new application

Your first step is to determine if creating an application is necessary. Consider:

- Can you extend an existing application instead of creating a new one?
- How many people will the application serve? If not many, is it worth the effort?
- Are you willing to maintain the application as things change over time?
- How often will anyone use this application? If not often, is it worth the effort?

If creating a new application makes sense, then move forward.

Defining the problem

If you're planning to create an application, you must have some notion of what you want it to do. You might not have a detailed idea of the solution, however. At a minimum, you need to define in detail:

- Desired application output
- Data going into the application

The data going in typically comes from a table of data. You need to know the table name and the type of data in it. You can use ServiceNow tools, like Table Builder, to store and customize your data.

The desired output might be a dashboard, data stored in a table, or a piece of equipment being sent to an employee. Before you decide on the output, show a mockup of your output to a lot of people to see if they can suggest improvements to the output. Do not skip this step.

Planning the solution

There are various ways to create an algorithm that produces the output you want. The easiest methodology is to use a flowchart that starts with the input data and shows each step in the process that leads to the output. Using a flowchart enables you to focus on the big picture of what you want to do and avoid how to do it. Look at your flowchart critically to optimize the process.

You may need to get permissions to use tables of data or even permission to create an application. For example, citizen developers are granted app development permissions through App Engine Management Center.

For more information about ServiceNow planning tools, see Planning your application.

Coding the application

To code the application, you can use:

- Low-code ServiceNow tools, such as App Engine Studio.
- Platform developer ServiceNow tools, such as ServiceNow Studio.
- Pro-dev source code tools, such as ServiceNow IDE and ServiceNow SDK.

For more information about ServiceNow coding platforms, see Developing your application.

Testing your application

Testing your application requires inputting data to make sure the outcome is accurate. It is important to input a lot of data, even incorrect data, to see how well your application handles correct and incorrect inputs. Put in data that is too large or the wrong type. Put in no data at all. Try to break your application to prevent your users from doing so. Handling error conditions gracefully is critical for a good customer experience.

For more information about ServiceNow testing tools, see Testing and debugging applications.

Deploying your application

Now that you've tested your application, it's time to deploy it so customers can use it. Application deployment is done by system administrators not developers.

For information about ServiceNow deployment tools, see Deploying applications.

Maintaining your application

Maintaining the application deals with issues such as overloading a server, application errors, and feature updates. Accounting for feature updates is a developer's job. Seeing how well an application is working is a system administrator's job.

For more information about ServiceNow tools that help you maintain your application, see Maintaining your application.

Documenting your application

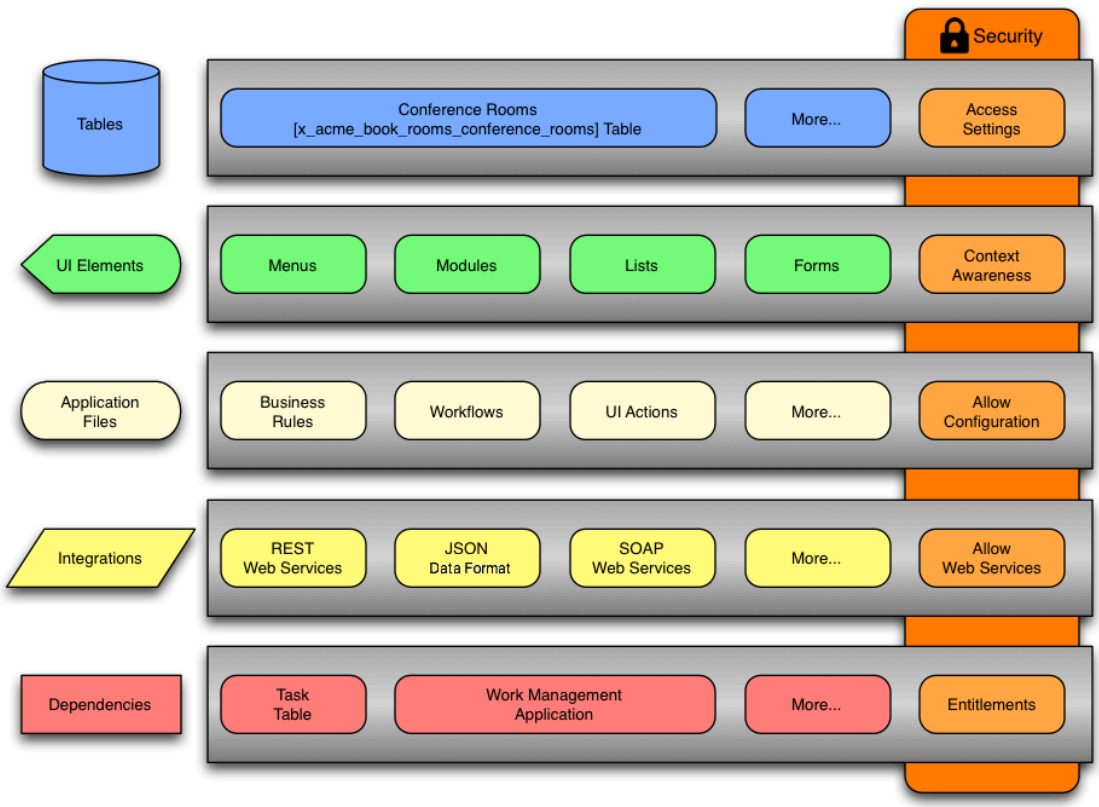
Application development is never complete until the application is documented. There are many reasons to document your application:

- Internal notes help other developers work on your application. It takes a lot of time to figure out the reasoning behind programming decisions and methodology. Documenting those helps future developers maintain your application.
- Users need to know how to use your application. As transparent as you believe the application is, you will find many people who get frustrated trying to use it. Do a usability test on your application and documentation to see where users run into problems. The broader the audience, the more important the application, the more important the documentation. Consider using an experienced technical writer to write the documentation.

Anatomy of an application

Applications consist of several types of files and records that collectively deliver a service.

Application model



Custom application record

The custom application record defines and identifies an application and all its associated artifacts.

It is similar to a system dictionary record for a table or column in that it stores the most current configuration of an application. The system automatically creates a custom application record during the application creation process. Application developers can use this record to perform the following tasks.

- Change the application name
- Change the application version

- View the scope the system uses to identify application files and configuration records
- Enable scoped administration
- Manage design and runtime access to the application
  - Select what JavaScript standards the application supports
  - Select how the system tracks runtime API operations
  - Permit or restrict access to tables from other applications
- Monitor or enforce subscriptions
- Select the default menu in which to display application modules
- Set the user role required to access the application
- Add or update a logo
- View all application files
- View resources from other applications on which the application depends
- View the run-time resource to which the application has been granted access
- View the design-time resources to which the application has been granted access

Application versions

Each installed application has a version as defined by its application developer in the custom application record.

The system uses this version information to determine whether there are updates available from the ServiceNow application repository or ServiceNow Store.

Application scope

Application scoping protects applications by identifying and restricting access to application files and data.

Administrators can specify what parts of an application are accessible to other applications from:

- The Custom application record
- Each application Table record

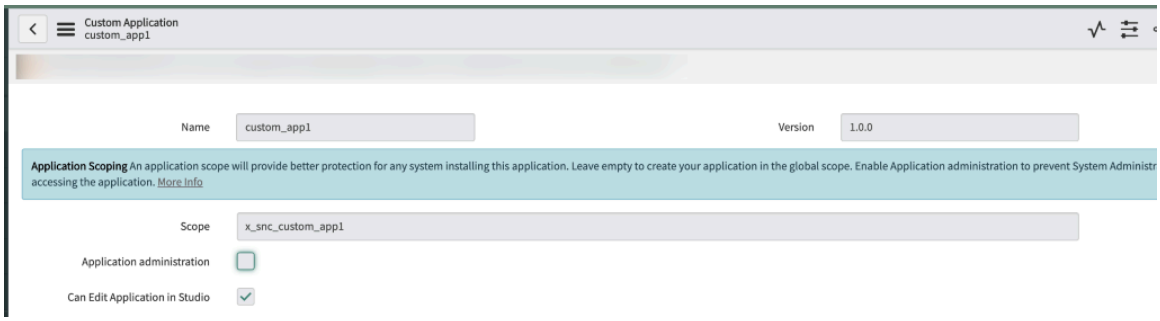
For example, suppose that you create a conference room booking application in its own application scope. By default, the application can access and change its own tables and business logic but other applications can't unless you give them explicit permission. The application scope ensures:

- The conference room booking application does not interrupt core business services.
- Other applications do not interfere with its normal functioning.

By default, all custom applications have a private scope that uniquely identifies them and their associated artifacts with a namespace identifier. The application scope prevents naming conflicts and allows the contextual development environment to determine what changes, if any, are permitted. Application developers specify an application scope when they create an application.

Tip:
Global apps can alter data that you don't intend to alter. You should leverage scoped apps to create new tables, and consider transitioning old ones to scoped apps. This allows you to split responsibilities with Delegated Development. To learn more about delegated development, see Delegated development and deployment.

Selecting the **Can Edit Application in Studio** option does not affect any custom or global applications in development on an instance. If you are the owner and choose to publish the application, you can restrict the development of customizations in Studio of. If you set **Can Edit Application in Studio** to false and then publish it, those who download the application to their own instances won't be able to edit the application in Studio. But they will have access to Legacy - Source Control integration features inside of Studio.



Related topics

Runtime access to applications tables

Global scope

The global scope is a special application scope that identifies applications developed prior to application scoping, or applications intended to be accessible to all other global applications.

Applications in the global scope do not append a unique namespace identifier to the application name. Global applications can have naming conflicts and data collisions when developers create multiple global applications with the same name.

Since all global applications are in the same scope, they bypass scope protections. Global applications allow other global applications access to their tables to

- Read records
- Run API requests
- Create configuration records

Typically, only applications provided by ServiceNow are in the global scope. However, any custom applications created before application scope was implemented are also in the global scope. Additionally, developers with the sn_g_app_creator.app_creator role can create applications in the global scope. For more information, see Allow global application development in Guided Application Creator.

Applications in the global scope are eligible for upload to the application repository, but not to the ServiceNow Store.

Related topics

Publish an application to the application repository

Publish an application to the ServiceNow Store

Namespace identifier

The system adds a namespace identifier to the front of application artifacts such as tables, scripts, and configuration records.

The identifier cannot be changed or removed from application artifacts to ensure that they are always associated to the proper application and that they have a unique name.

The system generates a namespace identifier from the following information:

Elements used to generate a namespace identifier

Element	Requirements	Sample Value
The prefix characters for a scoped application.	Scoped applications always start with an x_ prefix.	x_
The instance customer prefix (glide.appcreator.company.code)	This string is two to five characters long. ServiceNow generates this prefix for each customer. The instance stores the prefix in the *glide.appcreator.company.code* system property.	acme
The application ID	This string can be up to 40 characters long. If the string is longer than 18 characters, the system truncates the application name and appends it to prefix `x_<glide.appcreator.company.code>_`. Application developers set this ID when they create the application. The system uses the application name by default.	book_rooms

The example values generate a namespace identifier of x_acme_book_rooms.

Namespace identifier examples

The following examples illustrate generating namespace identifiers for applications, tables, and fields.

Example namespace identifiers

Action	Element generated	Explanation
1. Generate a namespace identifier for a private scope application called Book Rooms.	x_acme_book_rooms	This is a combination of the vendor prefix and applicatio
2. Generate a namespace identifier for a global scope	None	The system does not generate namespace prefixes for g applications.

Example namespace identifiers (continued)

Action	Element generated	Explanation
application called Marketing Events.		
3. Add the conference rooms table to the Book Rooms application.	x_acme_book_rooms_conference_rooms	This table is in the Book Rooms scope so begins with th namespace identifier.
4. Add a Marketing Event table to a global application.	u_marketing_event	Custom tables in the global scope always use the `u_` namespace identifier.
5. Select Book Rooms in the application picker and add the **Capacity** field on the Conference Rooms table.	capacity	The field is in the same scope as the table so it does not have its own namespace identifier. However, to dot-walk to the field in a script, you would use the full path including the table namespace identifier: `x_acme_book_rooms_conference_rooms.cap`
6. Select Book Rooms in the application picker and add the **Theme** field to the Marketing Event table.	x_acme_book_rooms_theme	The field is in a different scope from the table so it is prefixed with the `x_acme_book_rooms` namespace identifier. To dot-walk to the field in a script, you would use full path including the field namespace identifier: `u_marketing_event.x_acme_book_rooms_th` **Note:** This example assumes that the Marketing Event tab allows other application scopes to add fields.
7. Select Marketing Events in the application picker and add the **Theme** field to the Marketing Event table.	u_theme	Custom fields in the global scope use the **u_** prefix. To dot-walk to the field in a script, you would use `u_marketing_event.u_theme`.

Application tables

Application developers create tables and their associated lists and forms for users to add and update records.

An application owns its tables and determines whether other applications can access resources from them. For example, the Book Rooms application can store conference room data in the Conference Rooms [x_acme_book_rooms_conference_rooms] table and permit other applications to read this data.

The system uses standard access controls to manage user access to application data. During application creation, developers can specify an application-specific user role for these access controls. They can also use application access settings to manage runtime and design time access to application tables.

Note:
Certain ServiceNow AI Platform subscriptions include custom table entitlements. You can create custom tables for any purpose, up to the entitlement limit in the subscription. To learn more about how your usage administrator maps the custom tables that you create to subscriptions, see Map custom tables to a product subscription in Subscription Management.

Be aware of these database limitations:

- The system can only have a maximum of 1000 columns per table. Although 1000 columns is a specified limit, this limit doesn't mean you can physically have 1000 columns within a table. The number of columns within a table is defined by the database used in the ServiceNow datacenter, not by the ServiceNow AI Platform.
- Every table, regardless of storage engine, has a maximum row size of 65,535 bytes. Storage engines may place additional constraints on this limit, reducing the effective maximum row size.
- The system can't have more than 10 medium-length or longer **String** fields to a single table. Attempting to save a large number of characters in 11 or more String fields can result in the following error: Syntax Error or Access Rule Violation detected by database (Row size too large (> 8126).
- When you create fields, the u_ prefix is automatically added to the column name. If the column label that you enter contains leading numeric characters, they are replaced by the u_ prefix.

For more information on database limitations and general questions on tables in your ServiceNow instance, see KB0749585.

Related topics

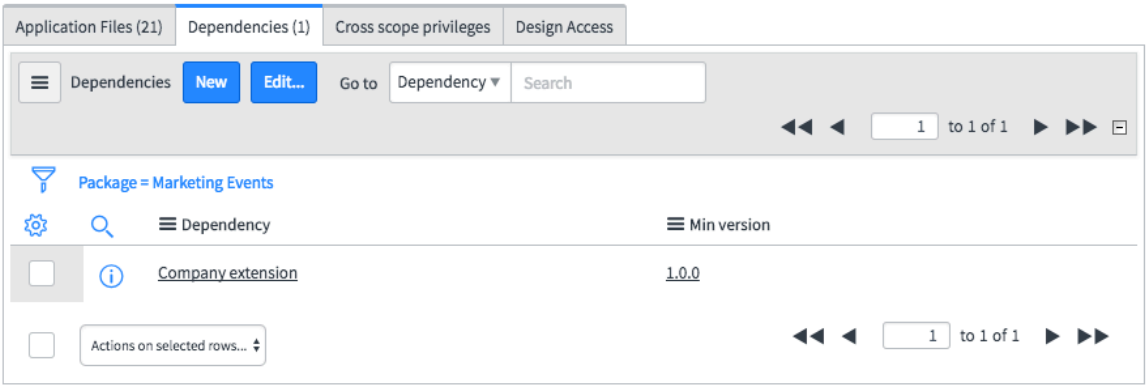
Application access settings

Dependencies for custom applications

Every custom application record includes a related list identifying its dependencies on other applications.

Administrators can review this list to determine whether an application poses any risk to existing processes or data. Application developers can use this list to ensure that their applications have the proper access to other applications.

Sample application dependencies



Application files

Application files are configuration records that allow developers to extend application functionality.

Application developers create application files when they add configuration records for application logic such as business rules, workflows, and script includes. Scoped applications do not own these tables, but they do own the records (files) within these tables. For example, adding a business rule to check for available rooms from the Conference Room table adds an application file to the Business Rule [sys_script] table. Application developers can view the complete list of application files from the custom application record.

The Application File [sys_metadata] table is the parent table for all tables that contain configuration records. It provides a series of standard fields that define the attributes for a configuration record. Tables that contain configuration records extend the Application File table. For example, the Business Rule [sys_script] table extends the Application File table.

Developers do not create application file records directly from the Application File table. Instead, they create or modify configuration records, and the system creates or modifies the associated application file record. Most configuration record tables do not display a reference field or related list for the Application File table. By default, only Applications [sys_scope] and Tables [sys_db_object] have a reference to the Application File table.

The system also tracks application file records in Update Sets. Whenever you change an application file record or its related configuration record, the system adds a corresponding record in the Customer Updates [sys_update_xml] table. The system uses a single update record, ensuring that a configuration record and its application file are kept in sync when transferring applications between instances. To avoid collision, users are warned when they edit an application file that has been previously edited in another Update Set.

Administrators can:

- View file properties for configuration records.
- Protect application files from changes during upgrades.
- View parent-child relationships between configuration records.

Related topics

Legacy - Add an application file to an application

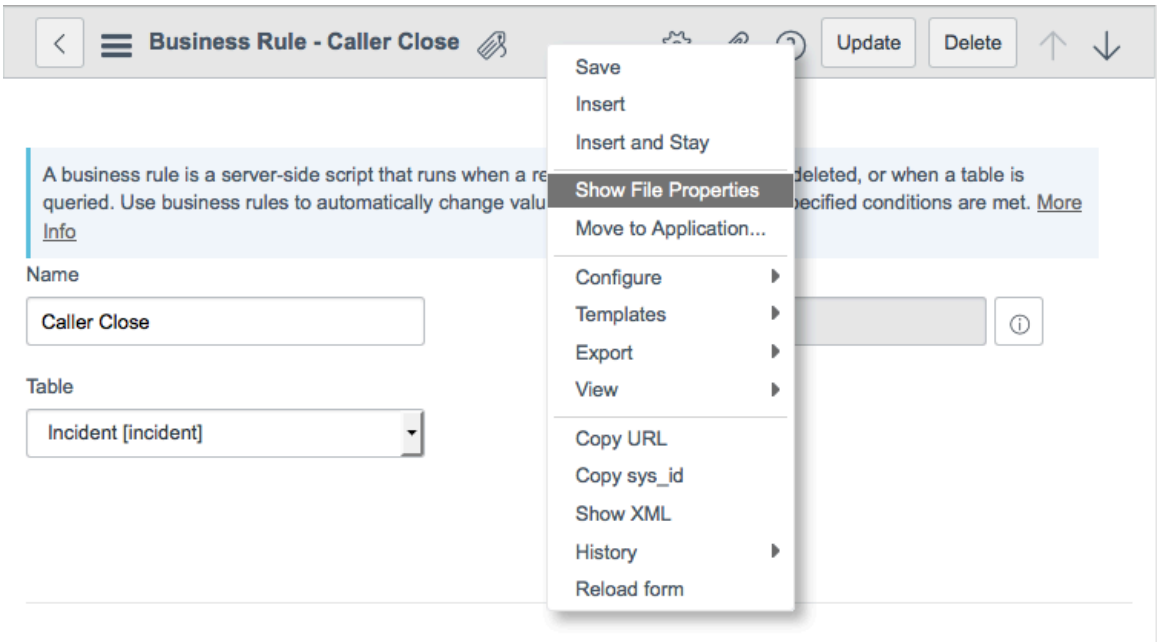
View file properties

Administrators can view the application file properties of a single record.

Before you begin
Role required: admin

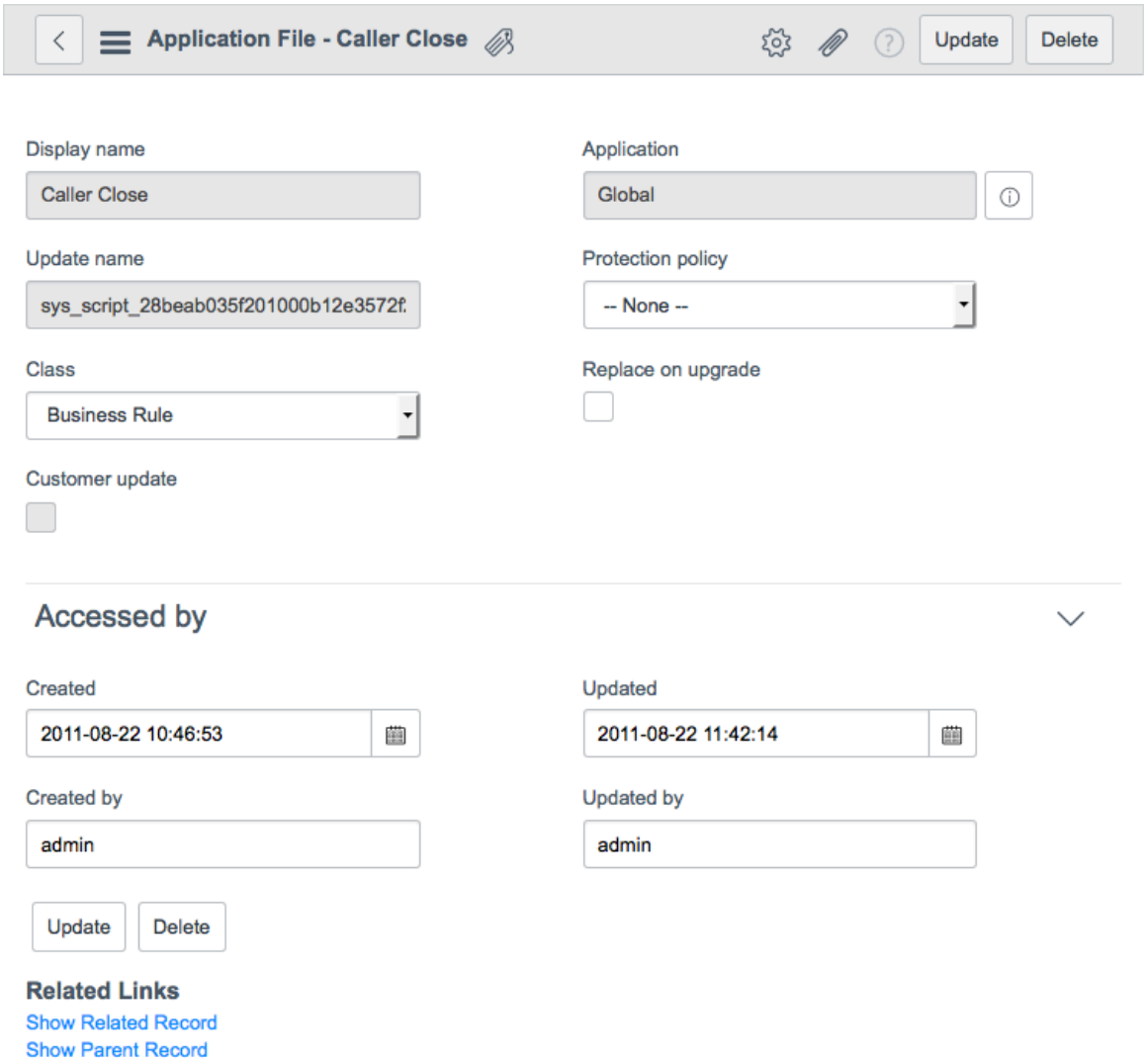
Procedure

1. Navigate to the form view of the configuration record.
   For example, navigate to **All** > **System Definition** > **Business Rules** and select a business rule for the Incident table.
2. Right-click the form header and select **Show File Properties**.



   The Application File table provides the standard fields that define the attributes for the configuration record.

3. To return to the configuration record view, click the **Show Related Record** related link.



4. Navigate between a customer update record, the file properties view, and the configuration record view.

Option	Description
Show Related Record related link	Navigate to the configuration record
Show Parent Record related link	Navigate to the parent record of the current configuration record.
Descendants related link	View child configuration records, such as a field label translation.

Application File form

Use the Application File form to view relationships between applications and configuration records.

Application File form fields

Field	Description
Display name	Display name for the configuration record.
Update name	Unique identifier for the configuration record. This value is used to identify versions and updates of the record.
Class	Table that contains the configuration record.
Application	Application that contains the configuration record.
Protection policy	Policy that determines if the configuration record is protected from changes. See Protected Application Files.
Created	Creation date of the configuration record.
Created by	User who created the configuration record.
Updated	Last update date for the configuration record.
Updated by	User who last updated the configuration record.
Related Record Versions	Version records for the related configuration record. A version record is created every time a user changes the related record. Use this list to compare versions of the configuration record or to revert to a previous version. See Versions.
Related Record Updates	Local update records for the related configuration record. An update record is created for the most recent change to the related record in a given Update Set. See Update Sets.

Note:
You may find it useful to manually add an **Owned By** (owned_by) field to this form. It indicates, for each application, who in IT owns the application and is responsible for maintaining information about it. The Data Certification function uses this field to send certification tasks every quarter, or at any time interval that is configured. You can assign certification tasks to other users, but using this field limits configuration effort. To learn more about adding a field to a form, and about Data Certification, see Using the form designer and Data Certification.

Application file protection policy

A read-only protection policy prevents anyone from modifying an application file or its related record.

Some application code shipped with the ServiceNow system requires special protection. Only a ServiceNow employee can alter the protection policy and then modify the application file or its related record. A ServiceNow employee cannot edit protected files without changing the policy designation first.

To prevent customizations from being overwritten by system upgrades, the upgrade process automatically skips changes to customer-updated records. If you modify an application file or related record that is later designated as *Read-only* in an upgrade, the application file maintains the default protection policy of *Write*. You keep the existing modifications and can continue modifying the records.

Note:
Reverting a customized file to its baseline state causes the record to inherit the new protection policy as well. For example, a record going from a *Write* protection policy to a *Read-only* protection policy.

Relationships between configuration records

The Application File Types table defines parent-child relationships between configuration records.

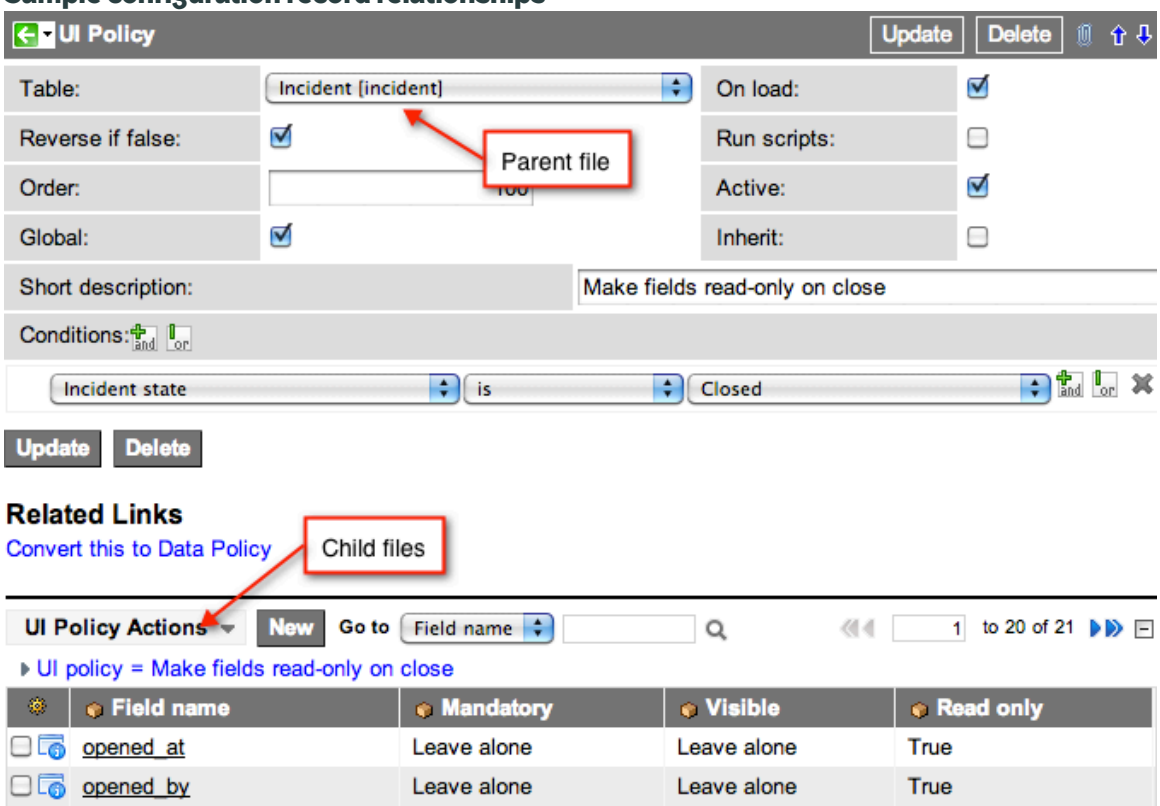
The system uses this structure to keep configuration records that normally belong together in the same application.

Note:
Do not modify the Application File Types table as it provides system functionality.

For example, consider the parent-child relationships for a UI policy.

- The UI policy is a child of the application table.
- UI policy actions are children of the UI policy.
- UI policy actions have a parent UI policy and a grandparent application table.
- The UI policy actions and the UI policy are all descendants of the application table.

Sample configuration record relationships



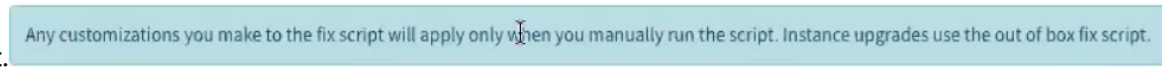
Fix scripts

A fix script is server-side JavaScript code that you run after an application is installed or upgraded.

Include fix scripts to make changes that are necessary for the data integrity or product stability of an application.

Administrators can create, manage, and run fix scripts. Users with the script_fix_admin role can create and manage fix scripts but cannot run fix scripts.

Note:
An annotation with the following message shows up when you open an existing fix script or create a new fix script.



Create a fix script

Create fix scripts to ensure the system installs or updates an application properly.

Before you begin
Role required: script_fix_admin or admin

About this task
Use fix scripts to add, update, and delete data, including rules, scripts, and property settings.

Procedure

1. Navigate to **All** > **System Definition** > **Fix Scripts**.
2. Click **New**.
3. Define the fix script by filling in the fields on the form.

Field	Description
Name	Unique, descriptive name for the fix script.
Unloadable	Option to create Customer Update records [sys_update_xml] when the fix script runs. Fix script tests enforce the Unloadable option.
Before	Option to run the fix script before installing or upgrading the application.
Record for rollback	Option to record execution of the script so that it can be reverted if needed.
Description	Description of the fix script.
Script	Code for the fix script.

4. Click **Submit**.
5. Test the fix script and make any necessary changes.

Test a fix script

Test your fix scripts to ensure they install or update applications as expected.

Before you begin
Role required: admin

About this task
Fix scripts add, update, and delete data, including rules, scripts, and property settings.

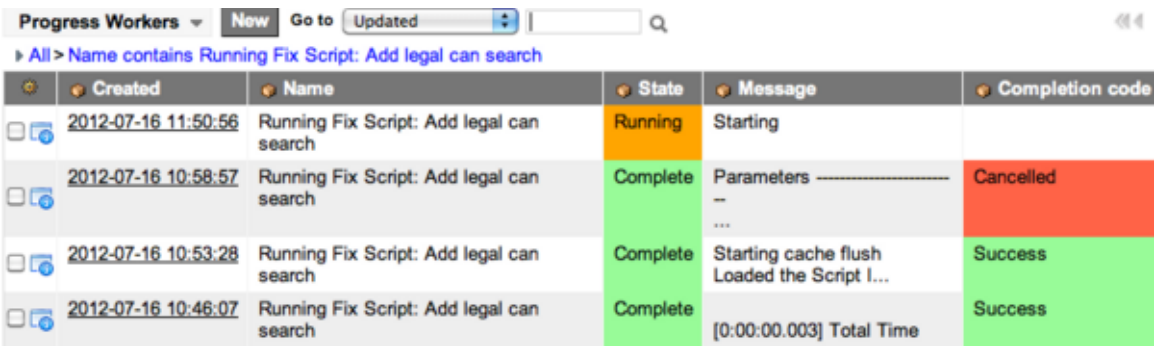
Note:
Do not test fix scripts on production instances.

Procedure

1. Open the fix script record.
2. Review the code design to ensure that it can run more than once on the same system without causing damage.
   For example, you may write a fix script that adds a role to a property by default. Design the script so that it can run multiple times on the same system without overwriting the existing data, even if it is not necessary to run the script again after the initial installation.
3. Click the **Run Fix Script** related link.
   Tests enforce the Unloadable option.
4. Confirm how to run the script.

Option	Description
Proceed in Background	Use this option for long-running scripts, or if you do not know the expected execution time.
Proceed	Use this option to run the script immediately and display the results in a confirmation win dow.

5. Review the results from the Progress Workers related list, and make any necessary changes.
6. To manually stop a running fix script:
   a. From the **Progress Workers** related list, select a worker in the `Running` state.
   b. Select the **Cancel job** related link.



Run fix scripts

Fix scripts run at the first installation or updated after they have been added to an application. However, previous fix scripts do not run on subsequent application updates, and you must manually run any necessary fix scripts.

Before you begin
Role required: admin

About this task
To run a fix script:

Procedure
1. Navigate to **All** > **System Definition** > **Fix Scripts**.
2. Edit the filter to search for your application name.
   For example, `Application | is | Book Rooms`).
3. Open the fix script record.
4. Click **Run Fix Script**.

Fulfillment tables

To enable a production instance to enforce entitled usage of your ServiceNow Store App, you configure the tables where only record owners or subscribed app users can make updates.

For any table that you, the developer, create or extend to support a custom application, you can specify that the table is a fulfillment table. In a fulfillment table, only a subscribed fulfiller user can perform a fulfiller action (typically, create/update/delete a not-own record).

In contrast, for a table that is not a fulfillment table, any user—even a user who is not subscribed —can act as a requester. The intent is to allow the usage admin to enable subscription enforcement on any production instance that implements the application.

Ownership of records in a fulfillment table

To enable the system to identify a fulfiller action, you define how to determine ownership of any record in the table. The developer of the application specifies the set of conditions that determine whether a user owns the record. For example, *`UserA`* owns a record in a task table if *`UserA`* opened the record or another resource opened the record on behalf of *`UserA`*.

For task-extended tables, time cards, and apps that require a subscription, the system sets the table as a fulfillment table by default and auto-assigns the ownership condition. For tables that you create to support your app, you can mark the table as a fulfillment table and can specify the ownership condition (for example, use the filter **[opened_by][is][currentUser] OR [caller_id][is] [currentUser]**).

System default conditions for ownership

Action	Ownership condition [owner_condition]
task extension	opened_by (read-only)
catalog request	requested_for (read-only)
other tables in apps that require a subscription	sys_created_by (read-only)
tables created by developer for app that requires a subscription	Specified by developer

Specify that a table is a fulfillment table

You configure a table as a fulfillment table to enable the system to prevent updates by users who are not subscribed to the app. For ServiceNow Store apps, you configure a table as a fulfillment table to enforce that fulfillment usage complies with your subscription use policy.

Before you begin
Role required: usage_admin, admin

Procedure

1. While working on a table, open the **Table Subscription Configuration** related list.
2. Select **Fulfillment table**.
3. Specify how the system determines ownership of records in the table so that both end users who own a record and subscribed fulfiller users can update the record: Specify the **Ownership condition**.
   For example, set the filter as **[opened_by][is][currentUser] OR [caller_id][is][currentUser]**.

Personal developer instance guide

ServiceNow® offers free personal developer instances (PDIs) to Developer Program members to learn, explore, and experiment with the ServiceNow AI Platform®.

ServiceNow® offers free personal developer instances (PDIs) to Developer Program members who want to develop applications or improve their skills on the ServiceNow AI Platform®. PDIs provide members with a sandbox environment to experiment with the ServiceNow AI Platform and build greater value for their businesses. The ServiceNow® Developer Program allows each member to have only one instance to maximize this valuable resource for all Developer Program members.

Personal Developer Instances (PDIs) can be used to test application ideas without impacting a customer or partner instance. PDIs are intended for you to use to learn, explore, and experiment with the ServiceNow AI Platform®. PDIs are not intended to be used for business or production purposes. Using a PDI for business or production is a violation of the ServiceNow® Developer Program Agreement. With a PDI, you receive admin-level access to all of the features the ServiceNow AI Platform® has to offer. Use your PDI on a continuous basis and it is yours forever.

Learn more on the ServiceNow Developer Site.

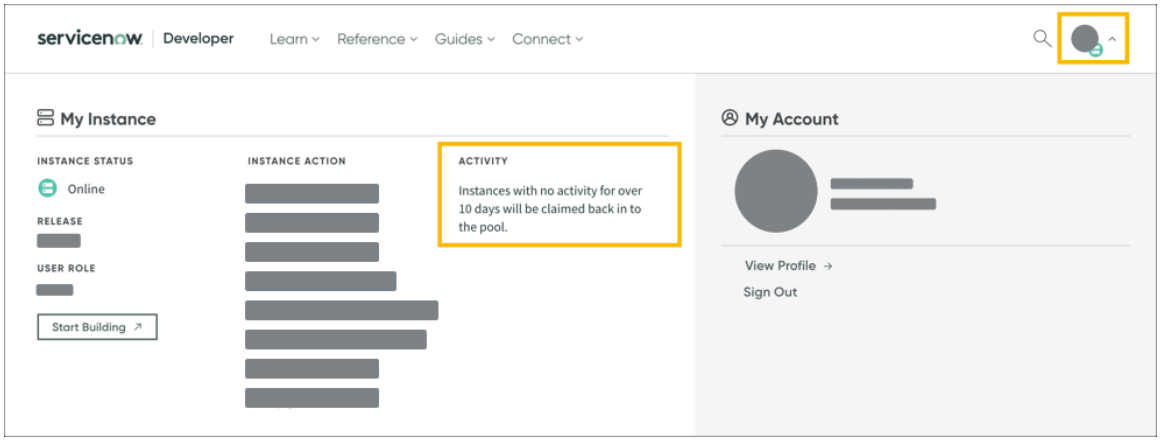
Understanding PDIs

PDIs require you to select a specific release and maintain consistent usage to avoid hibernation.

Sign in to the Developer Site to keep your PDI active. After a period of inactivity, your instance is reclaimed. Back up your work regularly to prevent data loss.

Note:
To save your work, commit your work to source control or create update sets. Work that is not saved may be lost due to instance reclamation.

When a PDI is reclaimed, the instance is unassigned from you, reset to its original state, and reassigned to another Developer Program member. Reclaiming instances allows the Developer Program to support active program members with free instances. The **Account** menu on the Developer Site header includes a reminder in the Activity section that you have ten days of inactivity.



Keep your PDI active by signing in or performing developer-type activities on your instance such as:

- Creating an application
- Creating a table
- Adding a field to a table
- Creating a Script Include

Hibernation

To keep the developer program free and provide PDIs to everyone who wants one, instances hibernate when they are idle. When a PDI hibernates, the database and application server shut down to conserve computing resources. All your data is preserved when the PDI hibernates. You can wake up a hibernating instance by signing in to the Developer Site. Waking an instance from hibernation resets the ten-day inactivity countdown.

In addition, you should also be aware of the following:

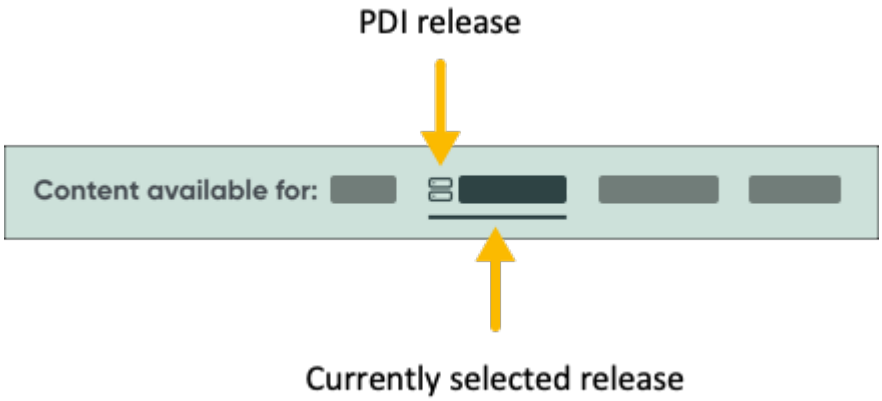
- PDIs are returned to the pool of available instances if they go unused for ten days. Duration may change due to availability.
- Availability is not guaranteed.
- PDIs cannot be clone targets or clone sources for customer or partner instances.
- PDIs cannot be linked via team development to customer or partner instances since they belong to ServiceNow®.
- PDIs cannot publish to the ServiceNow Application Repository or the ServiceNow® Store.
- PDIs cannot work with Machine Learning, Instance Data Replication, or MetricBase.
- Many ServiceNow® Store applications cannot be installed on PDIs.

Mainentance

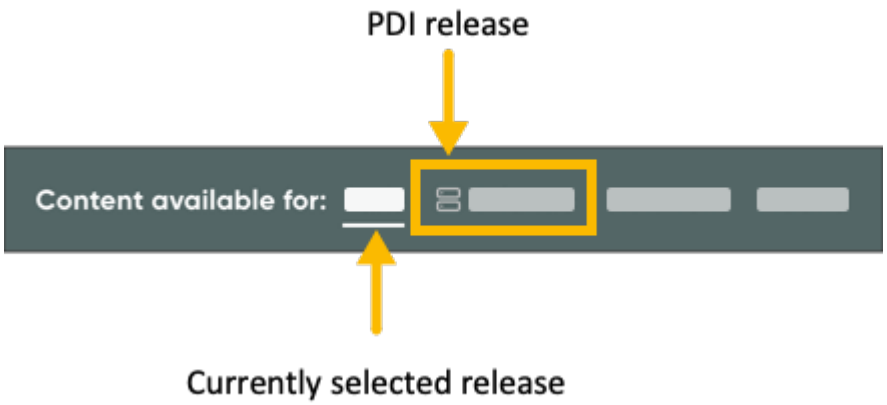
ServiceNow® occasionally needs to perform maintenance on the underlying infrastructure associated with PDIs. Maintenance includes updating and patching software, firmware, and replacing hardware. During maintenance, your PDI may not be accessible. The length of time required for maintenance depends on the type of maintenance. If available, the My Instance page indicates the expected completion time for planned maintenance. Your work and data on your PDI is unaffected by planned maintenance.

PDIs and the release selector

When content is available for different releases, use the release selector to select a release. The server icon next to the release name identifies the release of your PDI. The bar under the release name identifies the selected release. When working on your PDI, be sure the content matches your PDI.



If the content does not match your PDI release, click the release with the server icon to view the content that matches the release of your PDI.



The selector only shows releases for which the current page of content is available except when the content is not available for your PDI release. When the content is not available for your PDI release, the PDI release is grayed out and cannot be selected.

If your PDI is not on the latest release, follow the steps for upgrading your PDI to get the latest release.

Get a development instance

Request a personal development instance to build and test applications.

Before you begin
Role required: none

About this task
Get a free ServiceNow personal developer instance (PDI). Build applications within a sandbox to test application builders and deployment strategies.

Procedure

1. Navigate to the Developer Site in your browser.
2. Select **Sign In** or **Sign up and Start Building** if you do not have a ServiceNow developer account.
3. After logging into the ServiceNow Developer Site, select **Guides**.
4. Select **Guides Overview**.
5. Select **Personal Developer Instance (PDI) Guide** in the side navigation.
6. Follow the instructions to obtain your personal developer instance.

Obtaining a PDI

Get a new instance to run a PDI on the latest generally available release.

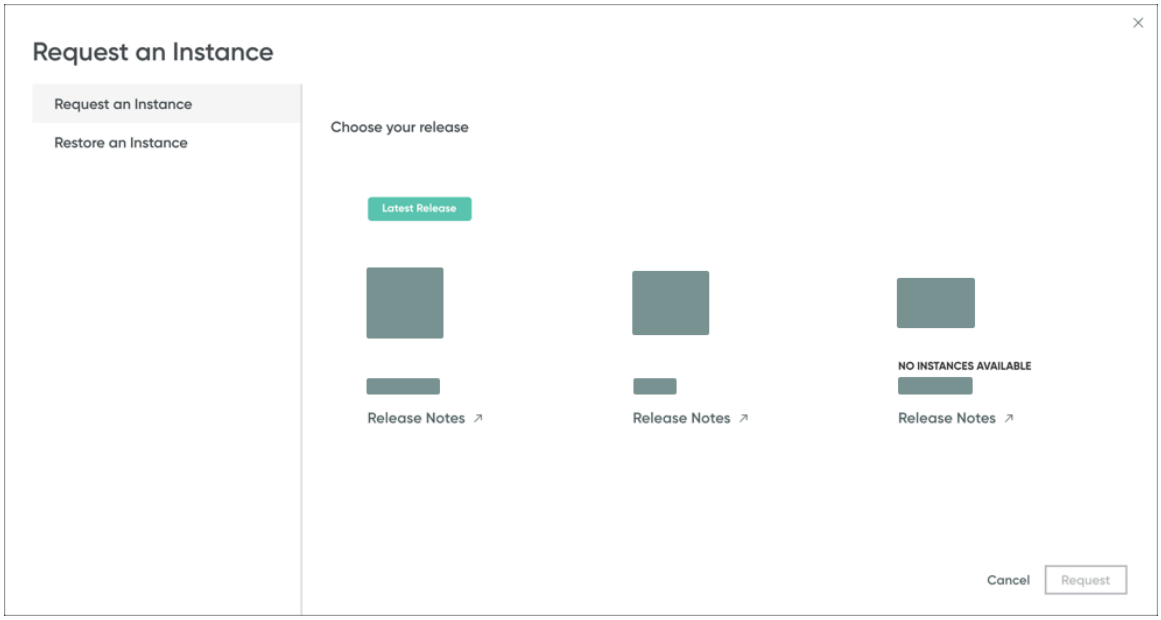
Before you begin
Role required: none

About this task
The first time you sign in to the Developer Site, you are given a PDI running the latest generally available release. You can use this PDI as long as you keep it active. If you want a PDI running an earlier release, release your PDI and request a new instance.

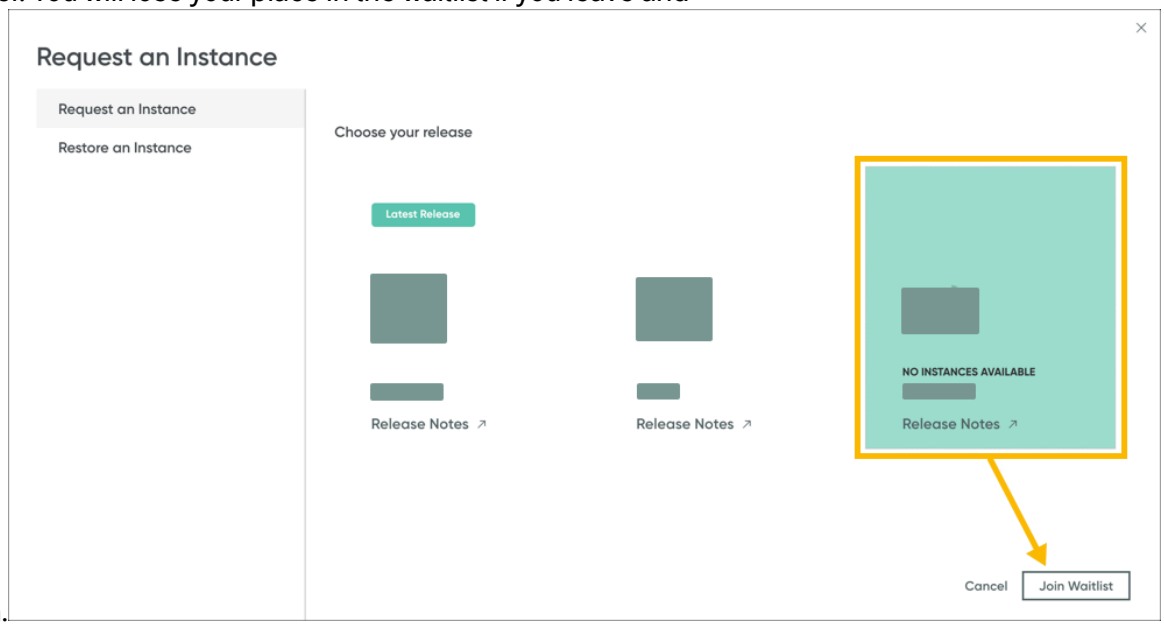
If you do not have a PDI because you released your PDI or went longer than ten days without activity, can request a new instance.

Procedure

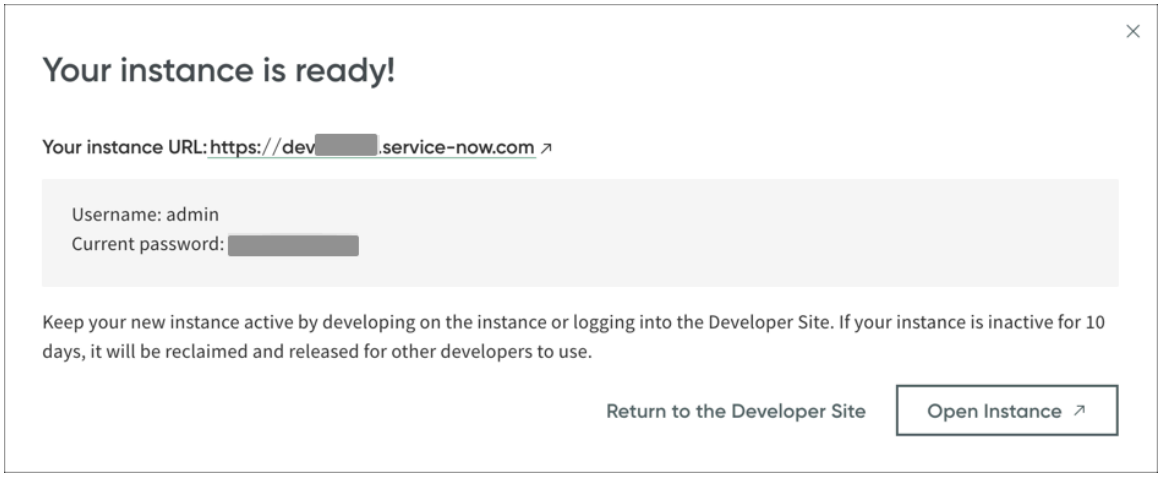
1. Sign in to the Developer Site.
2. Click the **Request Instance** button in the header.
3. In the **Request an Instance** dialog, select a ServiceNow® release for the instance and click the **Request** button.



- PDIs are a limited resource. If no PDIs are available for the desired release, select the release card for the release and click the **Join Waitlist** button. You will be notified when a PDI becomes available.
- To leave the waitlist, click the **Leave Waitlist** link in the header. You will lose your place in the waitlist if you leave and rejoin.



4. When an instance is assigned to you, the **Your instance is ready!** dialog supplies your instance URL and password.



5. Click the **Open Instance** button to open the instance in a new browser tab.

Accessing your PDI

Open your PDI and start building in the instance.

Before you begin

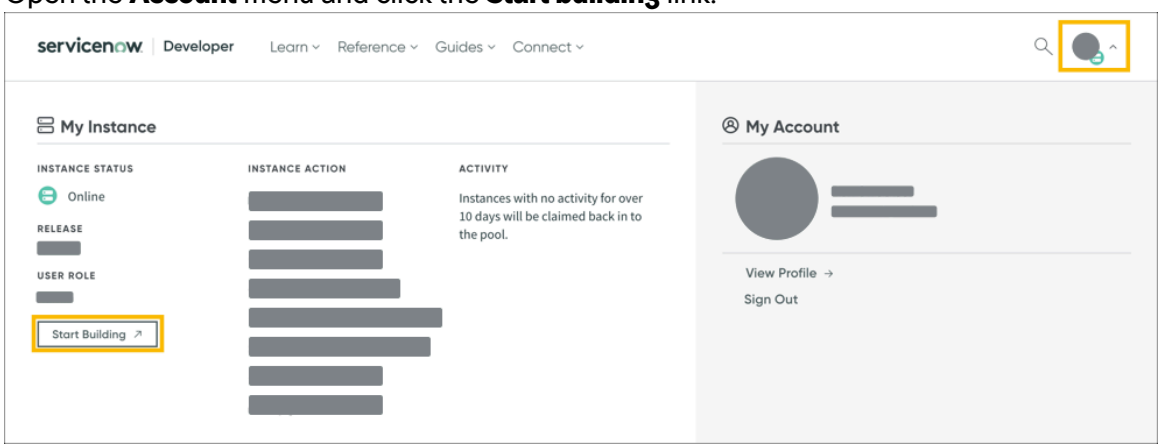
Role required: none

About this task

Once you have obtained a PDI, you can open the instance from **Account** menu.

Procedure

1. Sign in to the Developer Site.
2. Open the **Account** menu and click the **Start building** link.



3. If your PDI is hibernating, wait a couple minutes for your PDI to wake up before you can access it.
   Your PDI wakes from hibernation automatically when you sign in to the Developer site.
4. If your PDI hibernates while you are still signed in to the Developer Site, select the **Refresh Instance** instance action, then click the **Wake Instance** button to wake your PDI from hibernation.
5. If the instance is undergoing maintenance, you will need to wait for maintenance to complete before you can access your PDI.
6. If you have any issues accessing your PDI, review the Getting instance assistance section of this guide for steps you can take.

Activating a plugin from your PDI

One option to get started using an instance is to activate a plugin directly for your PDI.

Before you begin

Role required: none

About this task

You can activate most plugins directly in your PDI. Follow the instructions to Activate a plugin.

Procedure

1. On your PDI, open **All** > **System Definitions** > **Plugins**.
2. In the **Search** field, type the `<name of the plugin to install>`, for example, `app engine studio`.
3. On the plugin's card, click the **Install/Update All** button.
4. *Optional:*In the **Install** dialog, select the **Load demo data** option to install the plugin's sample data.
5. Select the **Install** button.
6. When the installation is complete, click the **Close** button in the **Install** dialog.

Activating a PDI plugin from the developer site

Activate your PDI plugin from the Developer Site to start working on an instance.

Before you begin

Role required: none

About this task

Some plugins require activation by ServiceNow® personnel on company and partner instances. You cannot activate these plugins from the plugins list. You may be able to activate these plugins from the **My Instance** section of the **Account** menu.

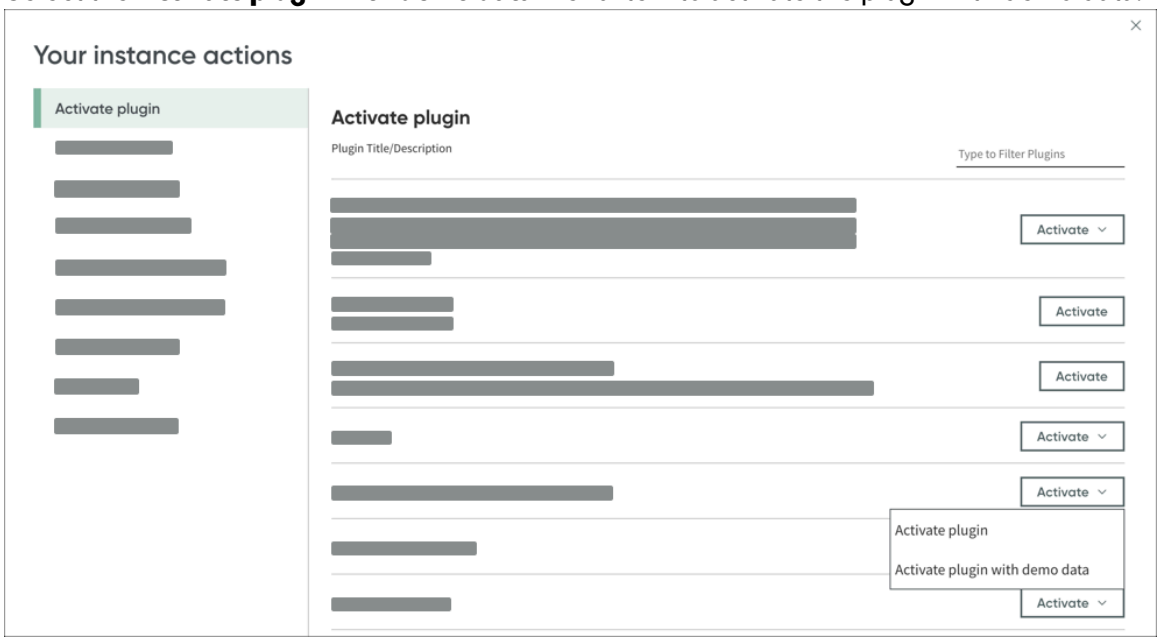
Note:
Many ServiceNow® Store applications cannot be installed to PDIs.

Procedure

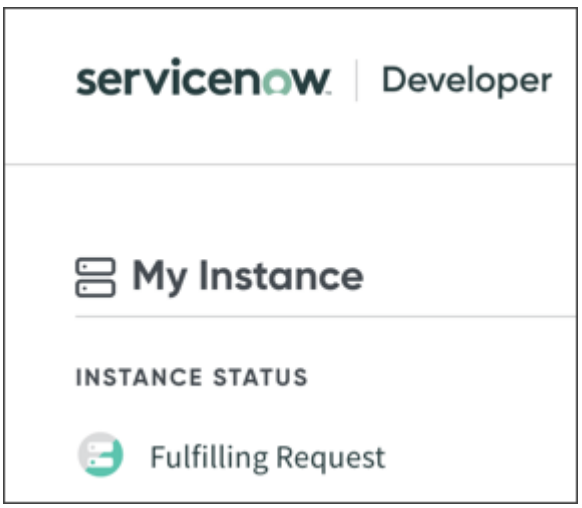
1. Open the **Account** menu to access the **My Instance** section.
2. Click the **Activate plugin** instance action.
3. In the **Your instance actions** dialog, find the plugin and click the **Activate** button.
4. If your plugin has demo data:

a. Select the **Activate plugin** menu item to activate the plugin without demo data.

b. Select the **Activate plugin with demo data** menu item to activate the plugin with demo data.



You cannot perform other instance actions while the plugin is activating. The **Instance Status** is **Fulfilling Request** while the plugin is activated.



Result
You will receive an email from the Developer Site when the plugin activation is complete. Plugins that require activation by ServiceNow® personnel do not appear in the plugins list in your PDI. The **Available Plugins** dialog does not show plugin activation status.

To use the plugin, reload the browser window where you logged in to your PDI.

Managing email properties for your PDI

Configure properties within your PDI to start sending and receiving emails.

Before you begin
Role required: none

About this task

ServiceNow® instances can send and receive email. Configuring email is typically the responsibility of a system administrator, not an application developer. In order to send and receive email with your PDI, email sending and receiving needs to be configured.

Unfortunately, personal developer instances (PDI) have been used both accidentally and maliciously to send email spam. In order to prevent spamming but still allow notifications on PDIs, we have restricted our email servers to sending to a single domain, guerrillamail.com. PDIs can receive email from any domain except guerrillamail.com.

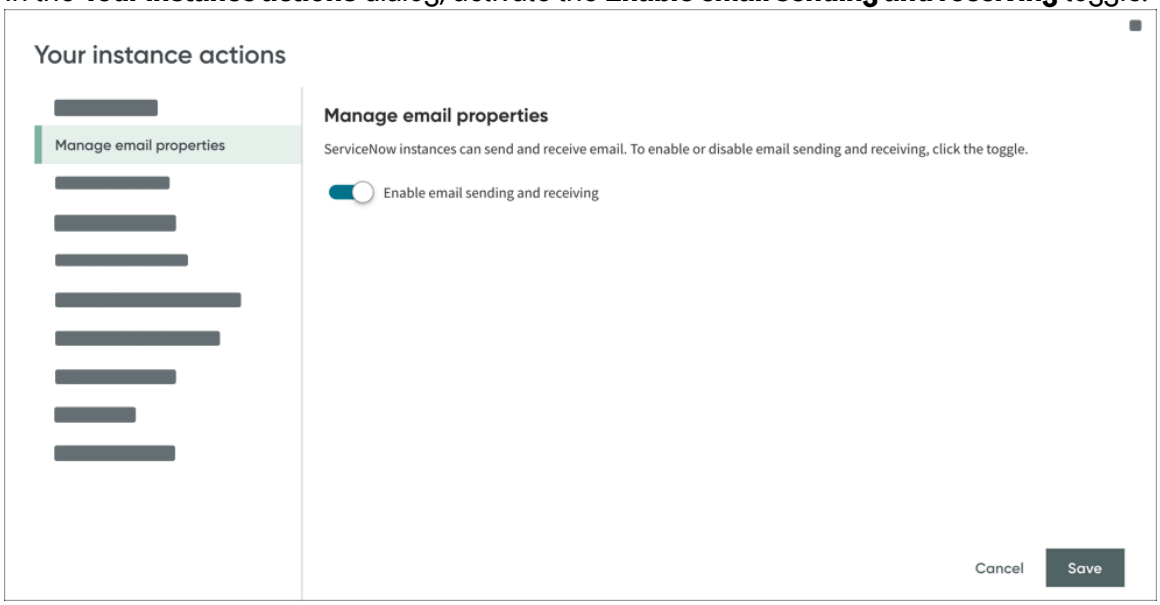
Procedure

1. To enable email sending and receiving from the Developer Site:

   a. Open the **Account** menu to access the **My Instance** section.

   b. Click the **Manage Email Properties** instance action.

   c. In the **Your instance actions** dialog, activate the **Enable email sending and receiving** toggle.



   d. Click the **Save** button.

2. To disable email sending and receiving, deactivate the **Enable email sending and receiving** toggle.

Releasing your PDI

Release your PDI to unassign the instance from you, reset to its original state, and reassign it to another Developer Program member.

Before you begin

Role required: none

About this task

If you no longer need your PDI or you are not going to be able to use it for an extended period of time, save your work and release the instance. When you release your PDI, the instance is unassigned from you, reset to its original state, and reassigned to another Developer Program

member. When you request an instance or restore an instance, you will receive a PDI with a different instance name and URL.

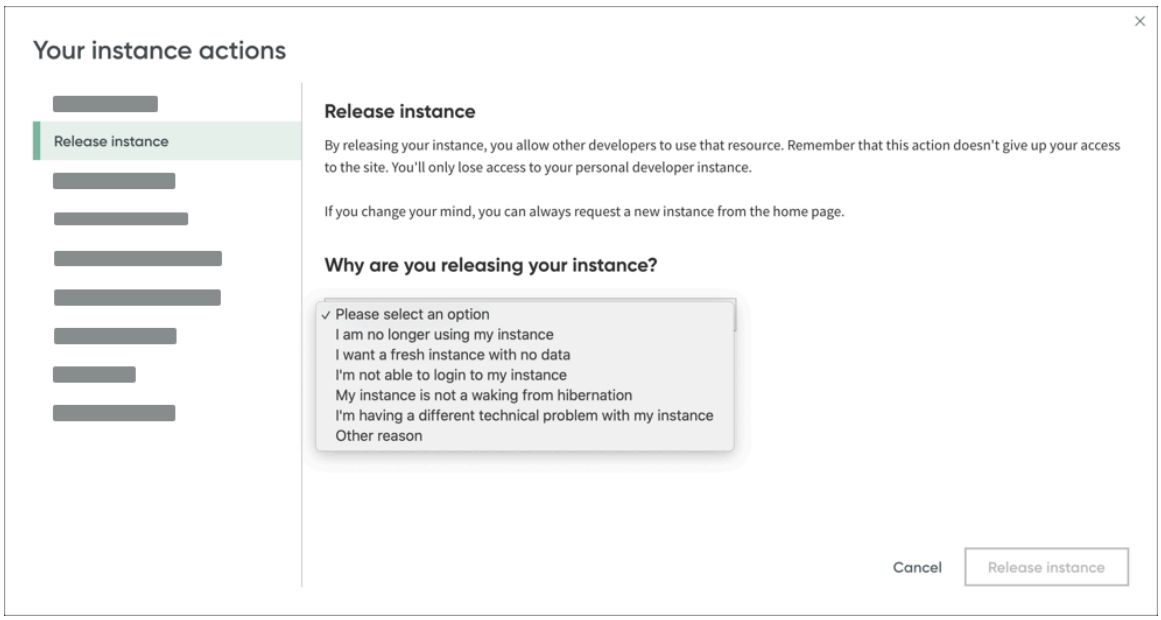
Procedure

To release your PDI:

a. Open the **Account** menu to access the **My Instance** section.

b. Click the **Release instance** action.

c. In the **Your instance actions** dialog, read the details and select a reason for releasing your instance from the choice list.



d. Click the **Release instance** button.

Changing your instance user role

Change your PDI user role to either App Engine Studio Creator or Admin for different views and levels of access.

Before you begin

Role required: none

About this task

Depending on your learning or testing needs, you may need to change the user role for your PDI.

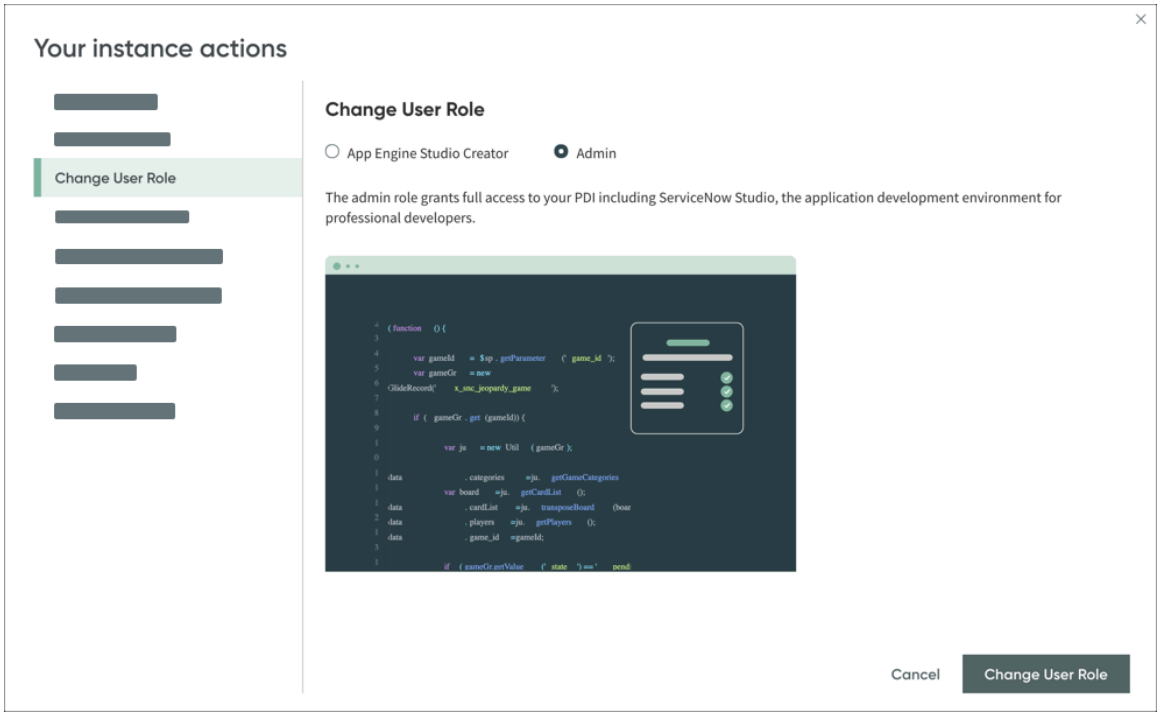
Procedure

1. To change your user role:

a. Open the **Account** menu to access the **My Instance** section.

b. Click the **Change User Role** instance action.

c. In the **Your instance actions** dialog, select **App Engine Studio Creator** or **Admin**.



d. Click the **Change User Role** button.

e. A message indicates when your change user role request is completed successfully.

2. To see the change in your user role:

a. Close any open PDI browser tabs.

b. Relaunch your PDI by clicking the **Start building** button in the **Account** menu.

Removing demo data from your PDI

If you do not need the demo data in your PDI included by default, remove it from your PDI.

Before you begin

Role required: none

About this task

By default, PDIs include demo data. You can remove the demo data from your PDI on the Developer Site.

Procedure

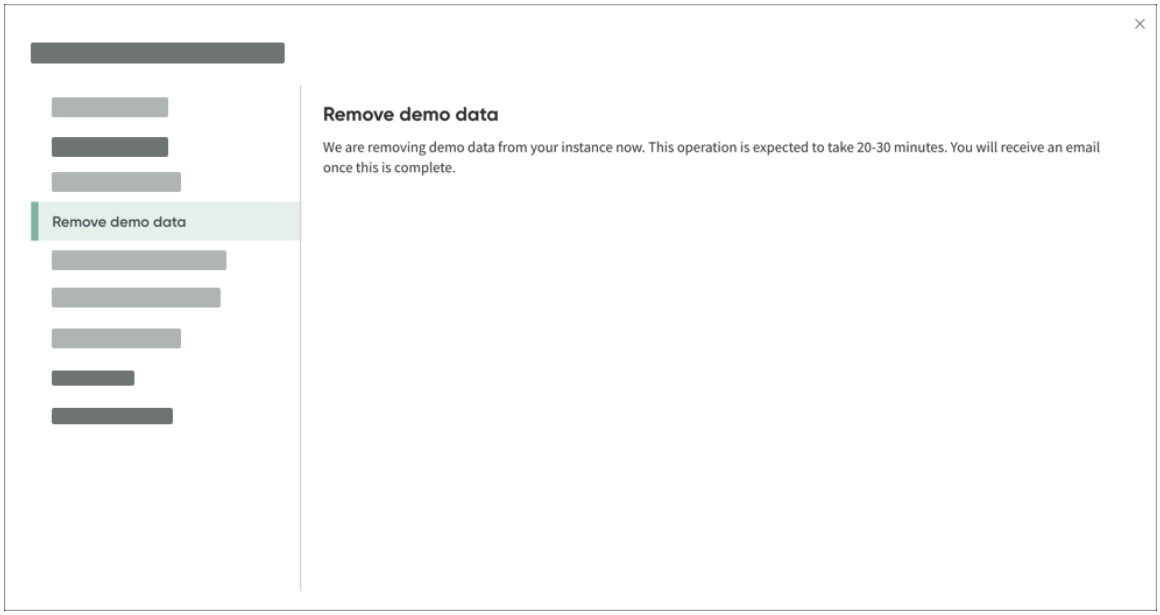
1. Open the **Account** menu to access the **My Instance** section.

2. Click the **Remove demo data** instance action.

3. In the **Your instance actions** dialog, read the description and acknowledge that demo data cannot be restored click the **Remove demo data** button.

Result

Demo data removal takes 20-30 minutes. The Developer Site will send you an email when the process is complete.



Resetting your PDI to its initial state

Reset your PDI to its initial state to start work with a fresh instance.

Before you begin

Role required: none

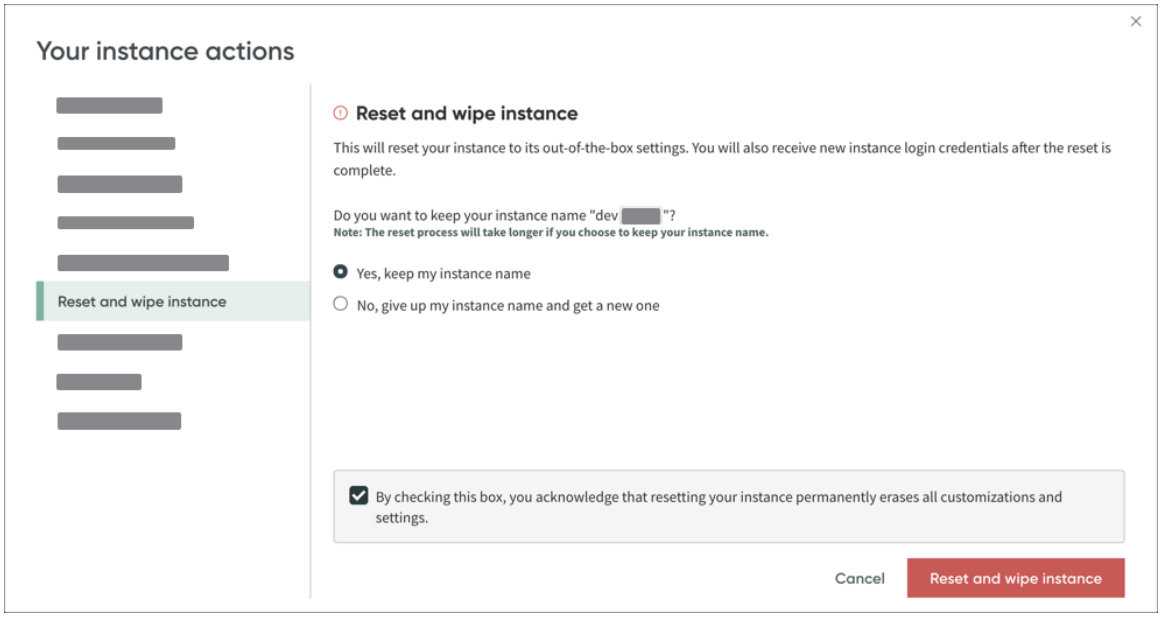
About this task

If you want to start work on a new application or want to work with a fresh instance, you can reset your PDI to its initial state.

> **Note:**
> If you reset and wipe your instance, you will lose your customizations and configurations you made to the PDI. Be sure to back up any work you want to save by committing your work to source control or creating update sets.

Procedure

1. Open the **Account** menu to access the **My Instance** section.
2. Click the **Reset and wipe instance** instance action.
3. In the **Your instance actions** dialog, select how to reset your instance.



- **Yes, keep my instance name**: Reset your instance to its default state. Resetting and wiping your instance may take a couple of hours to complete. You will receive an email when the process is complete.
- **No, give up my instance name and get a new one**: Get a new instance without waiting for the reset process.

4. Select the **By checking this box, you acknowledge that resetting your instance permanently erases all customizations and settings** disclaimer.
5. Click the **Reset and wipe instance** button.

Upgrading your PDI

Apply a patch or new release to your instance to upgrade your PDI.

Before you begin

Role required: none

About this task

When ServiceNow® releases patches and new releases, you can apply the patch or release to your PDI from the Developer Site. You have two options for upgrading a PDI.

Note:
Patches for a release are applied to your PDI automatically up to once a month to maintain compliance with the ServiceNow Patching Program. Sometimes patches may be made available to apply outside the regular patching schedule. Use the **Upgrade instance** option to apply a patch manually.

Procedure

1. Open the **Account** menu to access the **My Instance** section.
2. Click the **Upgrade instance** instance action.
3. In the **Your instance actions** dialog, select the patch or release to apply to your PDI and click the **Upgrade instance** button.

Result
Upgrading your instance may take a couple of hours to complete. Do not run the upgrade until you have time to let the upgrade run.

You will receive an email when the upgrade is complete.

Getting instance assistance

Get assistance for common issues accessing an instance and give feedback.

If you cannot access your PDI or get the **Your instance is currently offline** page, your PDI may be in the process of waking up from hibernation or your PDI just completed patching and needs to restart. Your PDI should be available within five minutes.

If your PDI is not available after five minutes, sign in to the Developer Site, select the arrow next to your avatar, and check the **My Instance** page to see if your PDI is still undergoing maintenance.

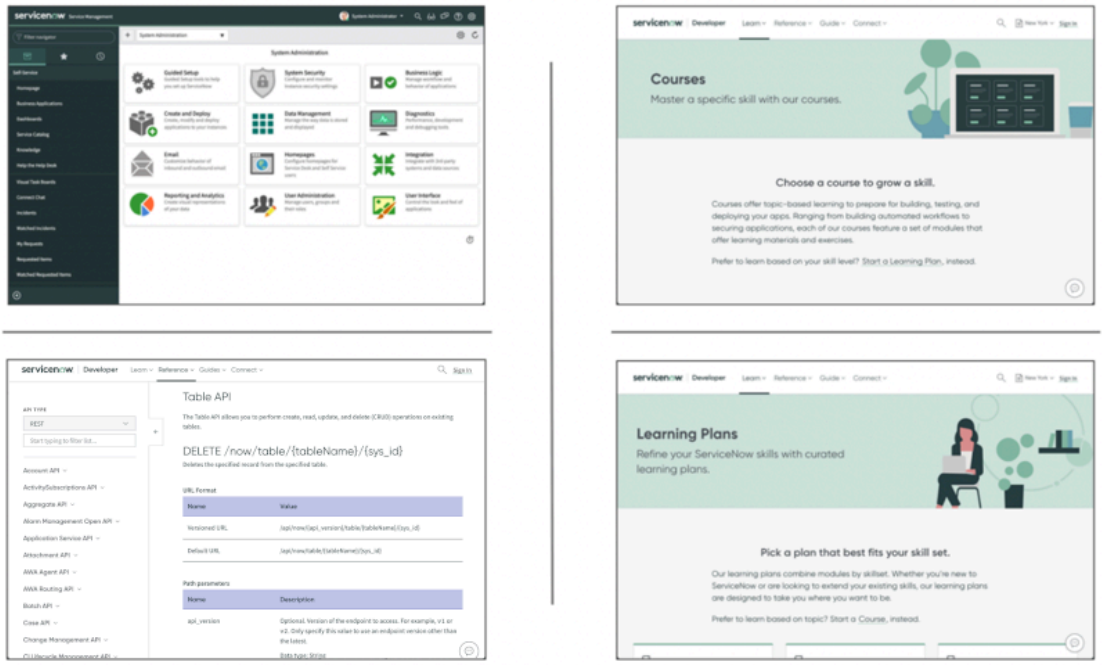
Early Availability guide

Early Availability provides you with exclusive access to the latest release over a month before market launch makes the release available for any customer to use.

The ServiceNow Developer Program provides Early Availability to pre-release versions. You have access to pre-release versions and resources through the ServiceNow Developer Site. As a Developer Program member, Early Availability provides you with exclusive access to the latest ServiceNow release over a month before market launch makes the release available for any customer to use. Early Availability includes:

- Personal developer instances (PDI)
- Learning plans
- Training courses
- APIs
- Documentation

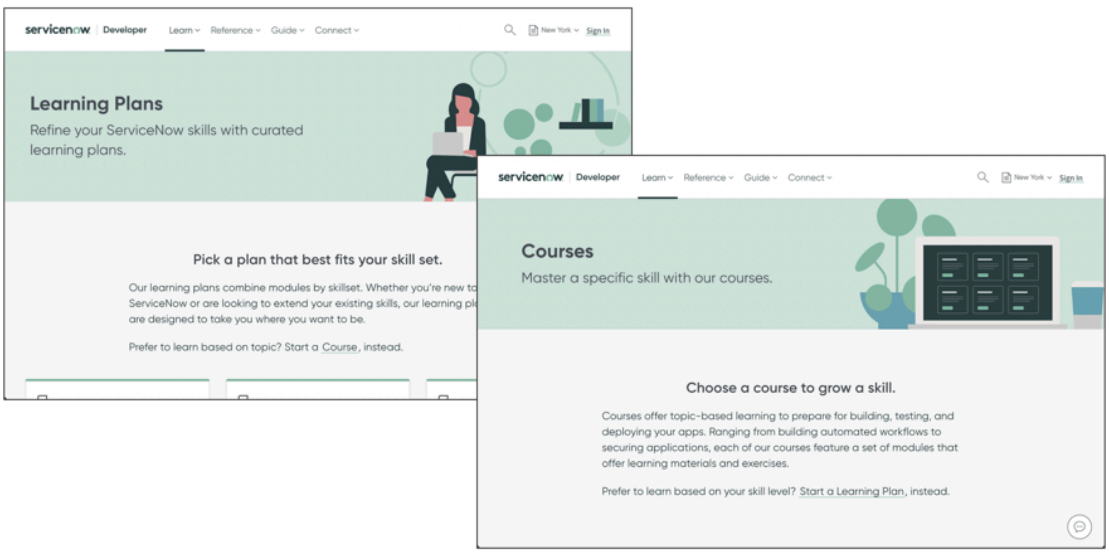
As a Developer Program member, you have the first opportunity to become familiar with the latest ServiceNow release through Early Availability. Early Availability gives you an opportunity to learn skills that will set you apart from other developers in the industry. You can leverage the latest features and upgrades to build new applications.



Early Availability learning plans and training courses

Early Availability gives you access to learning plans and training courses specifically for the latest release.

The Developer Program provides you with the training needed to successfully explore releases at your own pace. Early Availability gives you access to learning plans and training courses specifically for the latest release. Pick from a selection of easy-to-follow training courses or learning plans based on your current knowledge and experience. With either choice, you are guaranteed to build or improve your application development skills on the latest ServiceNow AI



Platform®.

Early availability API documentation

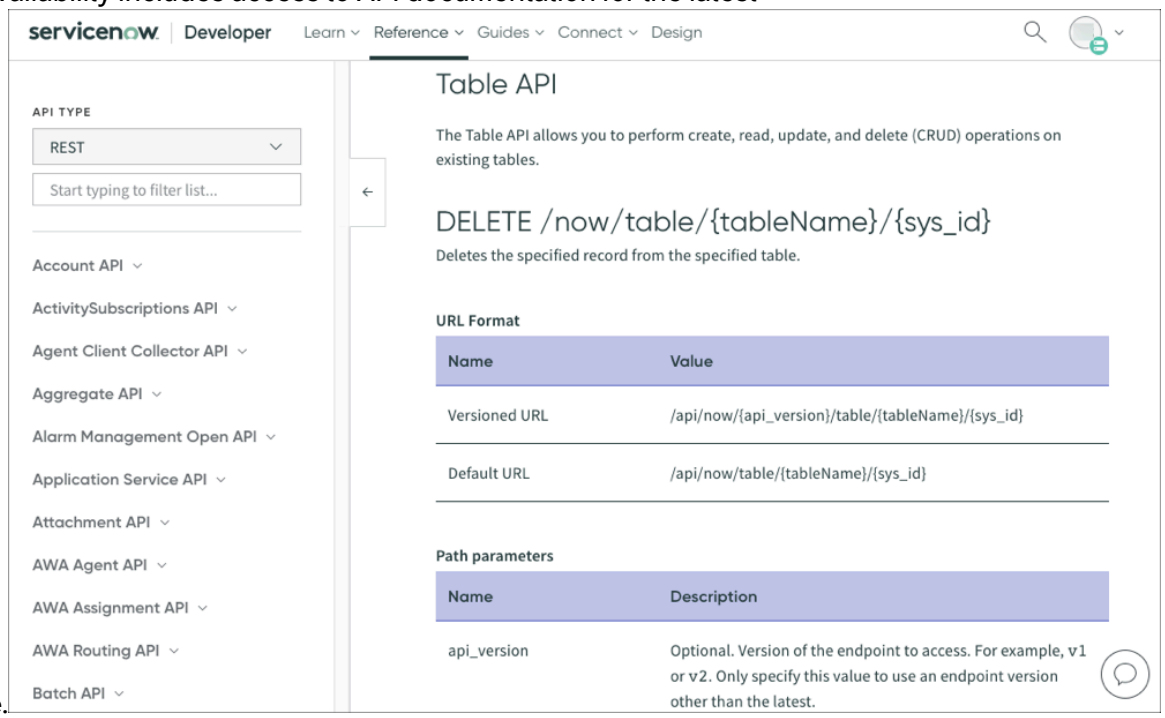
Early Availability includes access to API documentation for the latest release.

The Developer Program also contains API documentation on:

- Server-side APIs (scoped and global)
- Client-side APIs
- REST APIs
- Now Experience UI Framework

Server-side scoped APIs are for use within scoped applications, and may behave differently within the global scope. Server-side legacy APIs are documented for development work in global scope. New applications should be built using scoped APIs.

Early Availability includes access to API documentation for the latest release.



Low-code versus pro-code development

Learn the difference between low-code and pro-code solutions on the ServiceNow AI Platform.

Which builder should I use to create an app?

Types of builders

Want to build an app easily, without code?

Creator Studio specializes in helping you craft request-fulfillment applications without writing code. For example, an application to request office supplies by filling out a form, and someone approves or denies your request. For more information, see Exploring Creator Studio.

Need a more general app but still want low-code options?

App Engine Studio lets you build a broader range of apps than Creator Studio without being a programming pro. For more information, see Exploring App Engine Studio.

Are you a developer who wants more control in a centralized user interface?

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity. For more information, see Exploring ServiceNow Studio.

Are you a developer who wants to use industry-standard development tools and processes?

The ServiceNow IDE and ServiceNow SDK support developing applications in source code with ServiceNow Fluent, creating JavaScript modules, and using third-party libraries. ServiceNow Fluent is a domain-specific programming language for creating application metadata in code.

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. The ServiceNow SDK uses Visual Studio Code Desktop locally. For more information, see Building applications in source code.

What is low-code development

Low-code development is a new approach to app creation that allows users with limited coding experience to create powerful apps. Low-code development platforms rely on graphical interfaces and configuration instead of manual coding. These new low-code development platforms enable more people to create and deploy apps quickly and efficiently.

Benefits of low-code development

Low-code app development streamlines the development process to build more apps faster. Low-code solutions require fewer developers, and allow non-developers to build apps. Pre-built templates provide developers a head start building apps. System administrators can manage app development from a single location and collaborate with other developers. Decrease the time that it takes to deploy apps using predefined workflows in the ServiceNow AI Platform.

ServiceNow no-code and low-code development tools

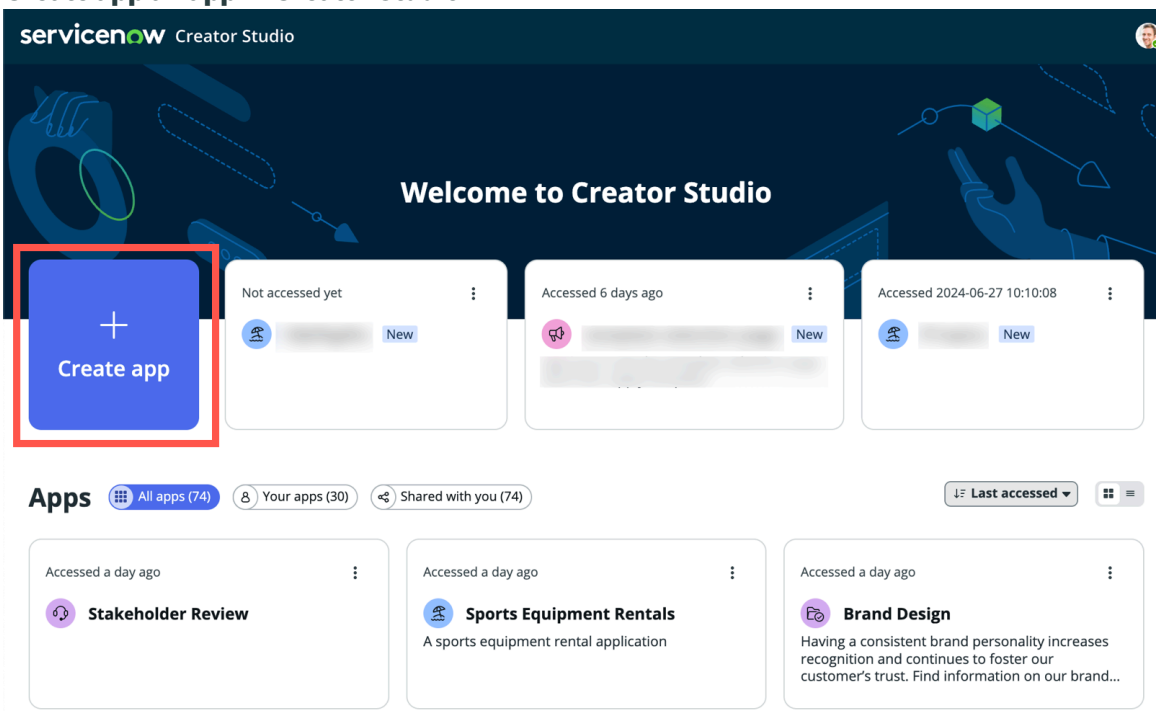
- Creator Studio
- App Engine Studio
- UI Builder
- Guided Application Creator
- Table Builder
- Flows in Workflow Studio
- Workspace Builder
- Exploring decision tables

For more information on low-code development tools, see Building low-code applications.

No-code development tool example

Creator Studio makes creating basic request-fufillment apps easier by dividing their creation into simple steps. You can create forms for users to request catalog items and use form submissions to initiate automated playbooks. Find out more in Creator Studio.

Create app an app in Creator Studio

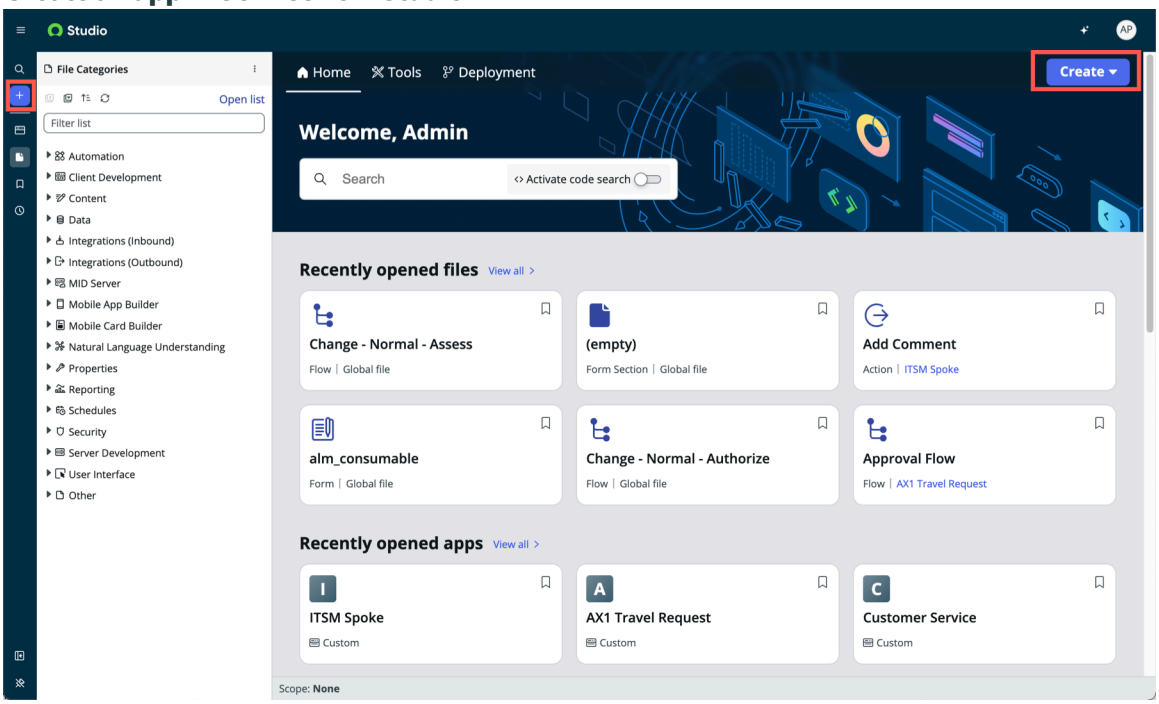


What is platform development

If you're comfortable with the ServiceNow AI Platform and some development tools, use the new ServiceNow Studio to access all of the builders and development tools in one place.

Platform development tool example

Create an app in ServiceNow Studio



What is pro-code development

Pro-code development is used by developers to create complex apps that can't be built with a low-code tool. Traditionally, pro-code development is used to create apps from scratch using custom code to solve a business need. Developers need to have knowledge of coding and how to use programming languages to build apps.

Benefits of pro-code development

The advantage of pro-code development is being able to create custom apps without the limitation of a tool. You can build custom apps unique to your business needs without limits. Developers can create apps with a custom look and feel to match your company's branding.

ServiceNow pro-code development tools

- ServiceNow IDE
- ServiceNow SDK
- Scripting
- REST APIs
- ServiceNow Extensions for Visual Studio Code

For more information on pro-code development tools, see Building pro-code applications.

Low-code versus no-code

The terms low-code and no-code tend to get used interchangeably, but they aren't exactly the same thing. While both low-code and no-code solutions provide tools for simplified app development, the differences are worth considering.

Low-code

Low-code platforms are designed for professional developers and non-technical business users. They require very little training or experience and use visual-based modeling to streamline the development process. They also allow people with coding experience to dive deeper, coding by hand when needed.

No-code

No-code platforms require no development experience, and are designed specifically for citizen developers and business users. No-code solutions open app development up to essentially everyone, but can lead to shadow IT—unsanctioned app development within an organization.

Modifying versus building an application

At a certain point in the lifetime of your applications, it might make sense to create an application to replace the old one. Cost savings, maintenance effort, and features available in new apps can help you determine if it's better to build a new application or modify an existing one.

Consider building a new application for the following reasons:

- The existing application is outdated and doesn't run well on newer hardware.
- Building a new application would produce cost savings over updating and maintaining the existing application.
- The existing application doesn't fundamentally meet your business needs.
- The existing application's features can easily transfer to a new application.
- Updating the existing application would cause significant disruptions to your UI and workflow because the existing application does not support the features you want to add.
- A new application would resonate with more users through an updated UI, more features, and ease of use than the existing application.
- Building a new application would be a quicker and easier process on low-code platforms like App Engine Studio (AES) than maintaining the existing application.

For more information, see the following topics:

- Use App Engine instead of customizations
- Customization vs configuration with ServiceNow Studio

Understand the ServiceNow® UI experiences

You can develop in the following ServiceNow AI Platform user interfaces: the Next Experience UI, the Workspace UI, the Classic Environment, and the Core UI. This topic explains these user interfaces.

Use the information in the following table to select the best UI experience for you and your team.

UI experiences on the ServiceNow AI Platform

UI experience	Benefits
Next Experience UI	**Unified Navigation** • Access app shell items—such as the contextual app pill, menu items, and search—all in one place across the entire

UI experiences on the ServiceNow AI Platform (continued)

UI experience	Benefits
	platform, no matter which application you're using. • Pin and unpin navigation menus to access menu items or free up screen real estate. **Additional menus** • Use the Favorites, History, and Workspaces menu items to navigate to and pin important resources. • Easily switch between Core UI applications and configurable workspaces. • Find your recently viewed items in a single menu. • Create custom menus for your end users. For more information, see Configure custom menus for Unified Navigation.
Configurable Workspace UI	**Improved agent efficiency** Work on multiple issues concurrently in an optimized, intuitive layout. **Speedy resolution** Resolve issues faster with automated suggestions powered by machine learning. **Proactive identification of major incidents** Get notified of potential major incidents based on issue frequency and impact. **Greater visibility for agents** Keep informed of updates and surface important insights with a live activity feed and analytics.
Working in the classic environment	**Work in lists and forms** Work in the classic ServiceNow AI Platform environment, creating lists from records. **Powerful app creation and development**

UI experiences on the ServiceNow AI Platform (continued)

UI experience	Benefits
	Create applications and develop with the full power of the ServiceNow AI Platform.
Working in Core UI	The Core UI interface provides an updated look and usability improvements. Notable features include real-time form updates, user presence, a redesigned application navigator with Favorites and History tabs, and improved activity streams. Core UI is the default user interface for new instances. For upgraded instances, an administrator must activate Core UI.

Support for developers

Here's where ServiceNow developers get support.

Where to find developer support

Resource	Description
ServiceNow Community	• Product hubs - Choose a product area and: ○ Connect to resources and learn about integrations. ○ Get expert and peer advise. ○ Network with peers. • Developer discussions - Connect with developers like you to ask questions, offer solutions, or build together. • Developer blog - See what the developer advocate team has to say.
Developer Site	• Sign up and start building on the ServiceNow AI Platform with a free account. • Join the ServiceNow developer program. • Access learning resources. • Access product and release documentation, APIs, CLI, and Component docs. • Access guides with an overview of the ServiceNow AI Platform, as well as

Where to find developer support (continued)

Resource	Description
	information on how to get started whether you're a new or experienced developer. • Learn ways to connect and collaborate with others working on similar problems.
ServiceNow training	Get started with training and certification.
ServiceNow developer videos	Access the ServiceNow Developer Program YouTube channel.

Application collaboration

People with admin privileges can manage the permissions of developers collaborating on an application.

Collaboration descriptors

You can use the predefined collaboration descriptors that are standard with activation, or create a custom collaboration descriptor. By using a collaboration descriptor, you can assign, manage, and monitor delegated development permissions.

Collaboration descriptors

Descriptors	Description
Owner	Owner of the application that manages other collaborators and can delete the application.
Editor	Standard descriptor to invite collaborators.
Custom	Non-standard (custom) descriptor created by the user.

Development permissions

By setting permissions, you can retain control over the system. You assign permissions to developers (or users who deploy applications) so that they can develop or deploy applications.

Development permissions

Permissions	Description
Application Management	Capabilities to delete the application and version source control access.
File type Access	File types that the user has access to.
Security / Entitlement	Capabilities to manage application security.
Programming Tools	Capabilities for script fields access.
Deployment	Deployment capabilities.

Application collaboration users

You can assign a collaboration descriptor for an individual by creating an app collaboration user. The user can only be assigned one collaboration descriptor for any application.

Application collaboration groups

You can assign a collaboration descriptor for a group of users by creating an app collaboration group. The group can only be assigned one collaboration descriptor for any application.

Install the app collaboration application

Install the app collaboration application so that you can view the collaboration feature in the UI.

Before you begin

Download the app collaboration application so that you can view the collaboration feature.

You must have an App Engine license.

Role required: admin

Procedure

1. Navigate to **System Applications** > **All Available Applications** > **All**.
2. Click the search icon () in the middle of the screen to search for the application collaboration application.
3. Click **Install**.

   Note:
   If app collaboration is already installed, the **Install** button is inactive.

4. Click **Activate**.

Result

The app collaboration feature is active.
After you install and activate the app collaboration plugin, all delegated developers will receive custom collaboration descriptors based on their current permissions and applications. The base system table provides standard Owner and Editor collaboration descriptors which are used in the collaboration feature.

If you install this plugin without App Engine Studio (AES) for an application, you should manage the delegated development permissions with this plugin or use the old Manage Developers screen to manage the permissions. Do not use both the plugin and the Manage Developers screen to manage the delegated development permissions.

To use the old Manage Developers screen, select your application and select Manage Developers in the related links on the form.

Create collaboration descriptors to assign permissions

Create descriptors in the collaboration application so that you can assign permissions to users or groups.

Before you begin

Role required: admin

About this task

Note:
You should create collaboration descriptors in addition to Owner and Editor in the global scope. If you want collaboration descriptors to appear and be used in AES, you should also set them to `standard = TRUE`. AES doesn't support collaboration descriptors that are created in custom scopes, and non-standard collaboration descriptors don't render in AES.

Use the predefined collaboration descriptors that are standard with activation, or create a custom collaboration descriptor. Collaboration descriptors enable you to assign, manage, and monitor delegated development permissions for each application or uniformly across multiple applications. You can assign each collaborator with one collaboration descriptor for an application. However, users can have multiple collaboration descriptors simultaneously if they belong to groups where collaboration descriptors have been assigned.

Procedure

1. Navigate to **All** > **App Engine** > **Collaboration** > **Descriptors**.
2. Click **New**.
3. On the form, fill in the fields.

App Collaboration Descriptor form

Field	Description
Name	Name for the descriptor.
Description	Description of the descriptor that includes permissions capabilities. Examples are as follows: ○ Owner ○ Editor
Application	Scope of the collaboration descriptor. Currently, all collaboration descriptors must be created into the global scope.
Standard	Option for inviting other collaborators with this role.

4. Click **Submit**.

Add permissions to collaboration descriptors

Add permissions to collaboration descriptors to manage your user's capabilities, such as the ability to delete the application or manage collaborators.

Before you begin
Role required: admin

Procedure

1. Navigate to **All** > **App Engine** > **Collaboration** > **Descriptors**.
2. To modify the permissions, select a collaboration descriptor from the related list.
3. To view the permission's options, click **Edit...** in the related list.
4. To add or remove permissions to the descriptor, click the arrows.

For a list of permissions, see Collaboration permissions.

5. Click **Save** and **Update**.

Assign collaboration descriptors to users

Assign collaboration descriptors to your users for a specific application so that you can define specific permissions for these users.

Before you begin

Role required: admin

Procedure

1. In the related list, click **Application collaboration users**.
2. Click **New**.
3. On the form, fill in the fields.

Collaboration users form

Field	Description
Collaboration Application	Application that the user works on.
Collaboration User	User that works on the application.
Collaboration Descriptor	Descriptor for the user on this application.

4. Click **Submit**.

Assign collaboration descriptors to groups

Assign collaboration descriptors to user groups for a specific application so that you can define specific permissions for these user groups.

Before you begin

Role required: admin

Procedure

1. In the related list, click **Application collaboration groups**.
2. Click **New**.
3. On the form, fill in the fields.

Collaboration groups form

Field	Description
Collaboration Application	Application that the group will work on.
Collaboration Group	Name of the collaboration group.
Collaboration Descriptor	Descriptor for the group on this application.

4. Click **Submit**.

Collaboration permissions

Assign permissions to define what collaborators can do.

Collaboration permissions

Permission	Description
Playbooks	Grants access to work with the Playbooks design environment to create processes. Editing activity subflows or actions requires the **Flow Designer** permission.
All Metadata	Grants access to work with all metadata.
Integrations	Grants access to web service APIs, REST APIs, data sources, and Integration Hub - Import.
UI Builder	Grants access to work with UI Builder to build out more complex interfaces.
Mobile builders	Grants access to build mobile experiences, such as with Mobile App Builder.
Security Management	Grants access to security management files, such as Access Control Lists and roles.
Service Portal	Grants access to work with Service Portal editors and tools.
Notifications	Grants access to create automatic email notifications and work with email templates.
Invite collaborators	Grants access to invite developers to collaborate on an app.
Workflow	Grants access to the Workflow Editor and Activity Creator.
Reporting	Grants access to reports and scheduled reports.
Workflow Studio	Grants access to the Workflow Studio design environment to create flows and actions.
Service Catalog	Grants access to work with catalog-related file types such as catalog items, record producers, and variables to add catalog items to apps.
Tables & forms	Grants access to model and layout-related file types such as table columns, form layout, and list layout.
Decision Tables	Grants access to work with Decision Tables to create decision logic based on multiple if-then rules.

Contextual development environment

The platform is a contextual development environment that displays the currently selected application, identifies the scope of every application artifact, and prevents any changes that violate the access settings for an application.

Application developers can use the contextual development environment to:

- Determine the application context
- View and select applications
- Enforce application version standards
- Enforce application resource throttling
- Enforce script protection policies

Application context

When application developers create new records, the system automatically assigns the records to the currently selected application in the application picker.

When application developers attempt to change existing records, the platform checks whether the currently selected application matches the scope of the application artifact. If they match, the application developer can save changes to the artifact. If they differ, the system makes the following changes to the user interface:

- Makes all the fields on the current record read-only.
- Displays a warning message that the application artifact belongs to another scope.

Application access settings

Application access settings determine whether one application can access resources from another application.

Application access settings are similar to access controls (ACLs) in that they allow you to restrict access to certain resources, but instead of restricting tables and records from users they restrict applications resources from other applications. There are several ways to set cross-scope access.

Setting type	Use	Description	Access setting location
Application design and runtime settings	Your application or application script requires access to a cross-scope resource.	Application designers can configure the following application access settings for the entire application. • Specify whether cross-scope tables can be selected during design-time activities. • Specify when scripts can run on cross-scope resources. • Specify what JavaScript standard the application supports.	Access determined by the cross-scope privilege record owned by the calling application.
Table design and runtime settings	A cross-scope application or web service requires access to perform CRUD operations on a table.	Application developers can also configure application access settings for individual tables.	Access determined by settings on the target table.

Setting type	Use	Description	Access setting location
		• Specify whether the table is available to other application scopes. • Specify what runtime operations from other application scopes the table supports. • Specify whether the table can be selected during design-time activities. • Specify whether the table can be accessed from web services.	
Restricted caller access privilege settings	A cross-scope application or script requires access to your application or application resource.	Admin users can configure the following application access settings for an entire application scope or application resource. • Specify whether a cross-scope script can access your application scope or an application resource. • Track cross-scope requests for access to application resources. • Approve or deny cross-scope requests to access application resources.	Access determined by the restricted caller access record owned by the target application.

Application design and runtime settings

The application design and runtime settings determine whether an application can access cross-scope resources.

Design and Runtime fields

Field	Description
JavaScript Mode	The JavaScript standard that the application supports. Select **ECMAScript 2021 (ES12)** to support features in ECMAScript 12th edition or **ES5 Standards Mode** to support features in ECMAScript 5th edition. Select **Compatibility Mode** to support earlier ECMAScript editions. For more information, see Set the JavaScript mode for an application.
Runtime Access Tracking	The application's handling of script access requests to resources in other applications. Select **None** to authorize all access requests to cross-scope resources without logging them. Select **Tracking** to log and authorize all access requests to cross-scope resources. Select **Enforcing** to log access requests to cross-scope resources but require an administrator to authorize each request.
Restrict Table Choices	The availability of cross-scope tables when designing the application. Clear the option to allow the application to see tables from other application scopes. Select the option to restrict design choices to only tables in the same application.

Runtime access tracking

Runtime access tracking allows administrators to manage script access to application resources by creating a list of script operations and targets that the system authorizes to run.

Runtime access tracking provides the following options:

Runtime access tracking options

Option	Tracking done	Authorization done	Results
None	The system does not track runtime access requests.	The system does not require authorization to run access requests.	Application scripts can access resources from other applications as long as the table-level access settings allow it.
Tracking	During development, the system creates a Cross-Scope Privilege record for each runtime access request. After installation, the system no longer tracks new runtime access requests.	The system sets the status of the Cross-Scope Privilege record to **Allowed**.	The system runs the tracked operation as long as the table-level access settings allow it.
Enforcing	During development, the system creates a Cross-Scope Privilege record for each runtime access request. After installation, the system no longer tracks new runtime access requests.	The system sets the status of the Cross-Scope Privilege record to **Requested**.	The system blocks the tracked operation from running until an Administrator manually changes the status to **Allowed** and the table-level access settings allow it.

During development, application designers must run all of an application's script logic to ensure the system tracks and authorizes the access requests to other applications.

Cross-scope privilege record

Runtime access tracking uses cross-scope privilege records to determine which script operations and targets the system allows to run.

The system creates cross-scope privilege records when:

- Runtime access tracking is set to **Tracking** or **Enforcing**.
- A script attempts to access another application.

Each cross-scope privilege record in the Cross scope privileges [sys_scope_privilege] table contains the following information.

Cross-scope privilege fields

Field	Description
Source Scope	The application requesting runtime access to another application's resources.
Target Scope	The application whose resources are being requested.
Target Name	The name of the table, script include, or script object being requested.
Target Type	The type of request: table, script include, or script object.
Operation	The operation the script performs on the target. The target type determines the available operations. Tables support the read, write, create, and delete operations. Script includes and script objects only support the execute API operation.
Status	The authorization for this record: requested, allowed, or denied

Administrators can manually create cross-scope privilege records for application developers in advance to communicate which cross-scope resources they expect developers to access. For example, administrators could create these cross-scope privilege records to permit application developers access to resources from Incident Management.

Sample cross-scope privilege records

Source Scope	Target Scope	Target Name	Operation	Status
My App	Global	incident	Read	Allowed
My App	Global	incident	Write	Allowed
My App	Global	ScopedGlideRecord	Execute API	Allowed

During testing, application developers should run all of their application scripting logic to ensure the system creates any necessary cross-scope privilege records. After application publication, the system only allows runtime requests to run that have a valid cross-scope privilege record.

Note:
Table privilege granting is limited to, at most, the permissions set on the table object (sys_db_object) record. For example, granting a scope privilege to delete for table incident would not be allowed if the table object for incident did not allow Can delete scopes.

Application design access record

Administrators use application design access records to specify which other applications are available to developers during application creation.

When administrators restrict an application's table choices, they can then create application design access records to grant developers access to the tables of selected applications.

Each application design access record contains the following information.

Application design access fields

Field	Description
Source Scope	The application requesting design access to another application's tables.
Target Package	The application whose tables will be available for design time access.
Application	The application to which this record belongs.

Application design access records allow administrators to have oversight of the resources available to application developers. When creating configuration records, application developers can only select tables from another application if there is an application design access record granting them access to the application. For example, administrators could create the following application design access records to grant developers access to tables from Incident Management and Problem Management.

Sample application design access records

Source Scope	Target Package
My App	Incident
My App	Problem Management

After developers create configuration records to other applications, the system displays these applications as dependencies.

Table design and runtime settings

The Application access fields determine whether a table is accessible to other applications during design-time or run-time operations.

Application access fields

Field	Description
Can read	Select the check box to allow script objects from other application scopes to read records stored in this table. This option offers runtime protection. For example, a script in another application can query data on this table. You must first select read access to grant any other API record operation.
Can write	Select the check box to allow script objects from other application scopes to modify records stored in this table. This option offers runtime protection. For example, a script in another application can modify a field value on this table. This option is available only when the **Can read** check box is selected. Clear the check box to prevent script objects from other application scopes from modifying data stored in this table.
Can create	Select the check box to allow script objects from other application scopes to create records in this table. This option offers runtime protection. For example, a script in another application can insert a new record in this table. This option is available only when the **Can read** check box is selected. Clear the check box to prevent script objects from other application scopes from creating records in this table.

Application access fields (continued)

Field	Description
Can delete	Select the check box to allow script objects from other application scopes to delete records from this table. This option offers runtime protection. For example, a script in another application can remove a record from this table. This option is available only when the **Can read** check box is selected. Clear the check box to prevent script objects from other application scopes from deleting records from this table.
Allow access to this table via web services	Select the check box to allow users to make inbound web service queries to this table. This option offers both design-time and runtime protection. The user performing the query must have the correct permissions to access this table, even when this check box is selected. Clear the check box to prevent users from making web service queries to this table.

Runtime access to applications tables

Runtime access determines if an API or web service call can run against an application table.

Access permissions can be set for the following access points.

Runtime access points

Access points	Description
Script API	Any supported object or method call from the scoped system API such as a GlideRecord call.
Web services	Any supported web service call such as a REST, JSON, or SOAP web service.

The system does not prevent you from creating API or web service calls to the application tables, rather it determines if the API or web service call is allowed to run against the application table. API or web service calls that violate the access permissions for an application table produce an error. For example, making a web service call to a protected application table produces a 403 Forbidden HTTP error.

Default runtime access permissions

The default runtime access permissions apply to new application data tables.

By default, new application tables only allow read access from other application scopes.

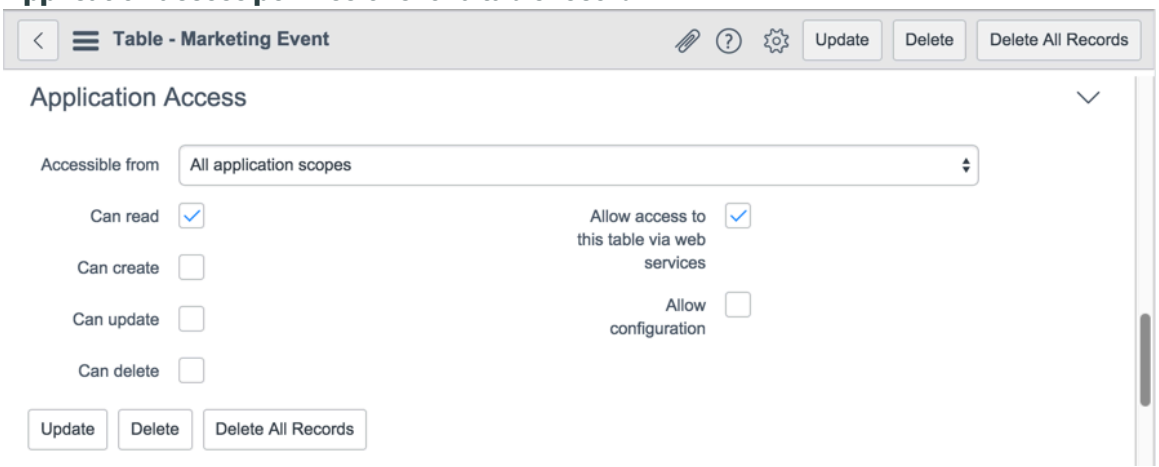
Default runtime access

Field	Value
Accessible from	All application scopes
Can read	Enabled
Can create	Disabled

Default runtime access (continued)

Field	Value
Can update	Disabled
Can delete	Disabled
Allow access to this table via web services	Enabled

Application access permissions for a table record



Set runtime access to application tables

Set these access permissions to protect application tables at runtime.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **System Applications** > **Applications**.
2. Click the button for the application type you want to edit.

Option	Description
Developed	displays applications created on this in stance.
Downloaded	displays applications downloaded on this in stance.

3. Click the application name or the **Edit** button for the application you want to work on.
4. From the **Tables** related list, select the table whose access permission you want to set.
5. From the **Application Access** section, set the runtime access permissions.
6. Click **Update**.

Example denying all runtime access to a table

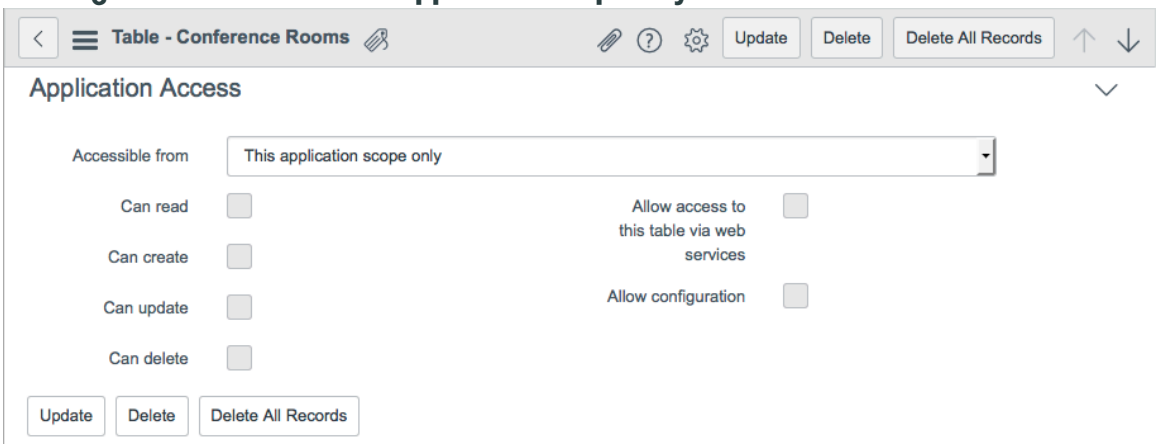
You can prevent script API and web service calls from other application scopes.

Typically, this is to prevent any other application from creating or modifying data in the table. Denying access requires setting the following value in the table record.

Denying all runtime access

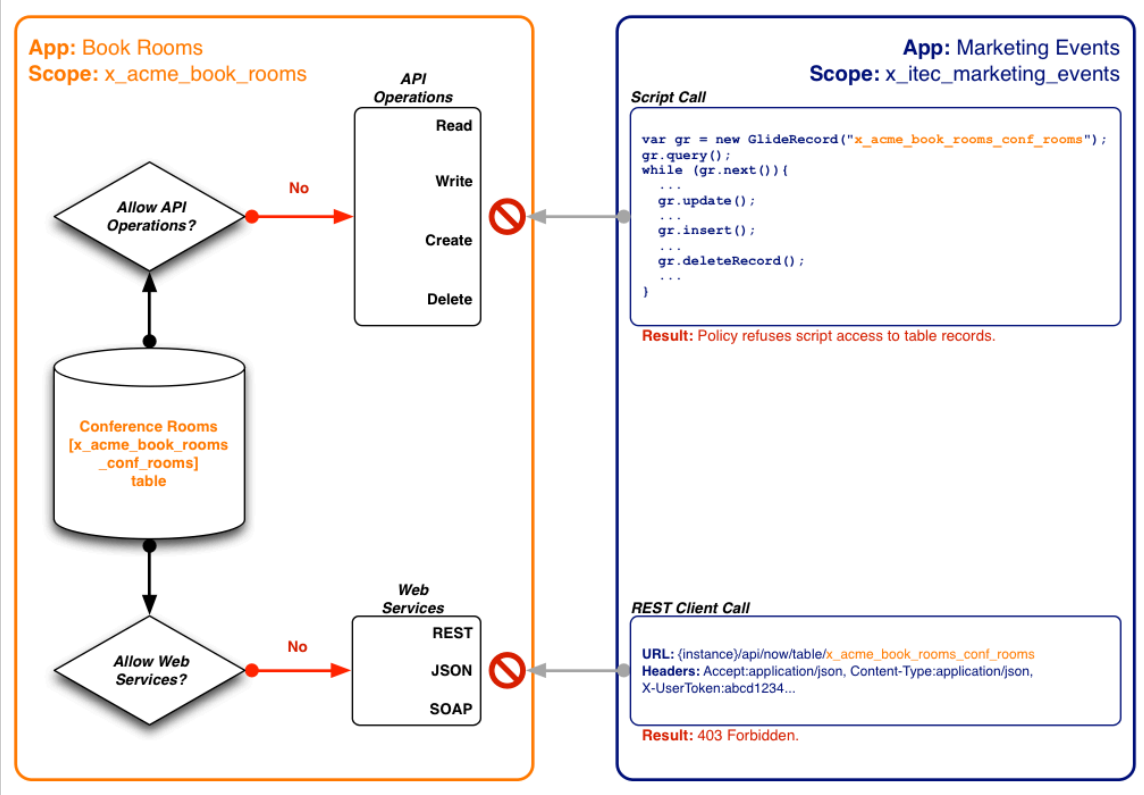
Field	Value
Accessible from	This application scope only
Can read	Disabled
Can create	Disabled
Can update	Disabled
Can delete	Disabled
Allow access to this table via web services	Disabled

Limiting runtime access to this application scope only



The following diagram illustrates the effect of denying other application scopes access to application tables from script API and web service calls.

Deny all runtime access permissions to application tables



Example granting all runtime access to a table

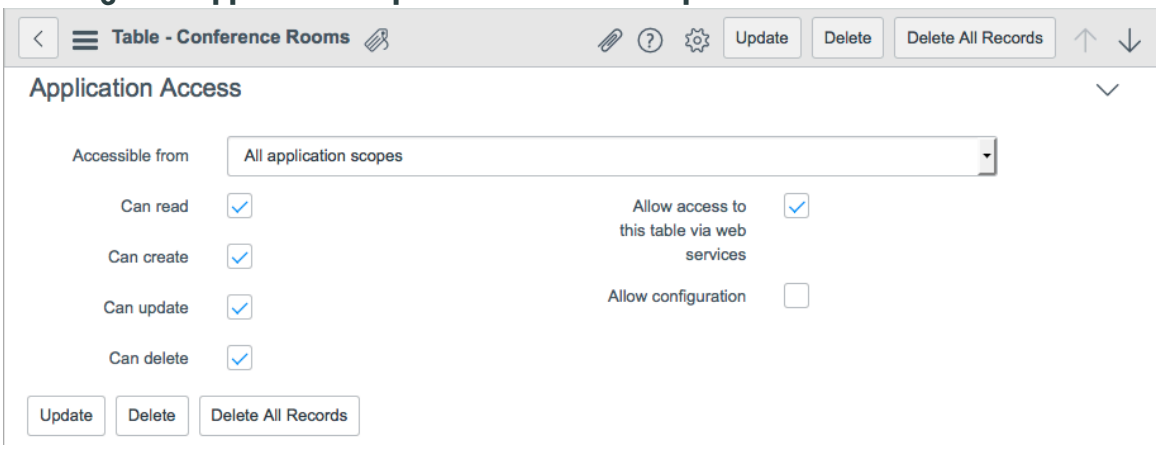
You can permit some or all runtime script API and web service calls from other application scopes.

Granting access requires setting the following values in the table record.

Granting all runtime access

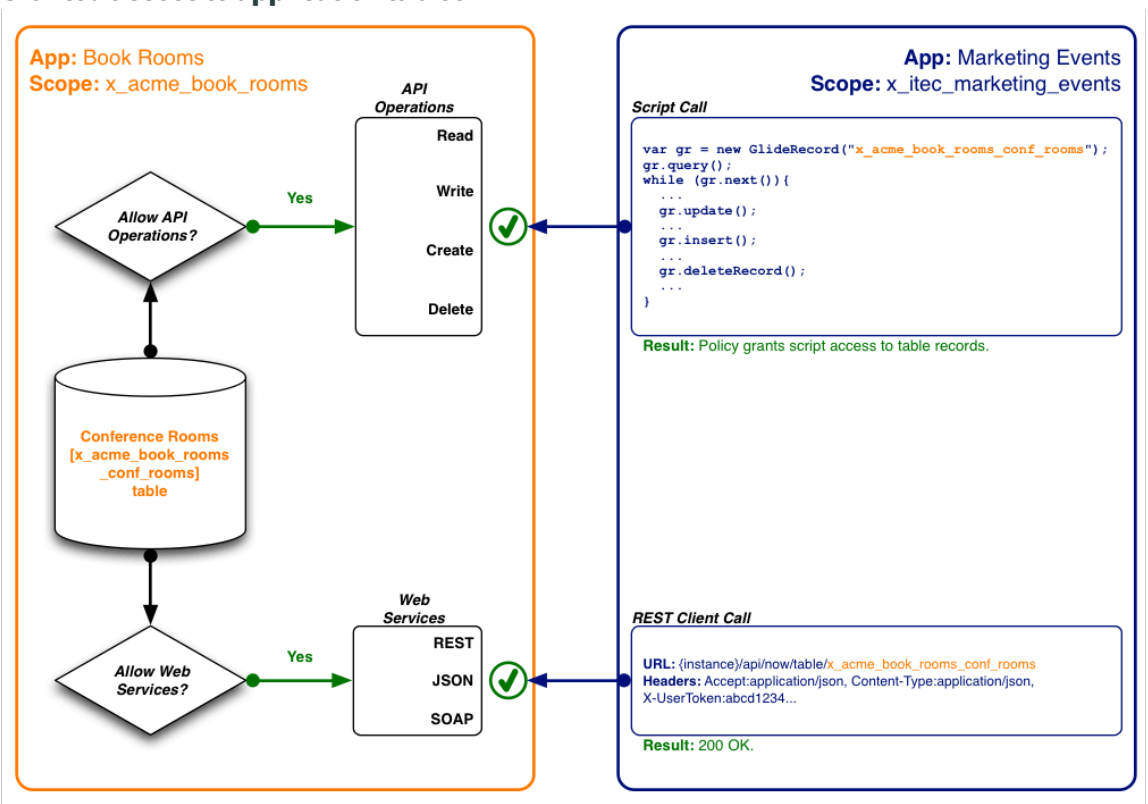
Field	Value
Accessible from	All application scopes
Can read	Enabled
Can create	Enabled
Can update	Enabled
Can delete	Enabled
Allow access to this table via web services	Enabled

Granting other application scopes all runtime access permissions



The following diagram illustrates the effect of granting access to application tables from API calls and web services in other application scopes.

Granted access to application tables



Design-time access to application tables

As the application developer, you can grant or deny other applications the permission to create configuration records, also known as application files, that extend the functionality of an application.

The permission applies to any platform feature that extends the functionality of an application data table such as:

- Business rules
- UI actions
- Client scripts

These access permissions protect the application data table at design-time. The system prevents you from creating configuration records by hiding the application data table as an option in the **Table** field. For example, a protected application table does not appear as an option when you create configuration records such as UI actions and client scripts.

Even when permission is granted to create configuration records, some configuration records have additional restrictions to protect application data from unwanted changes from other application scopes.

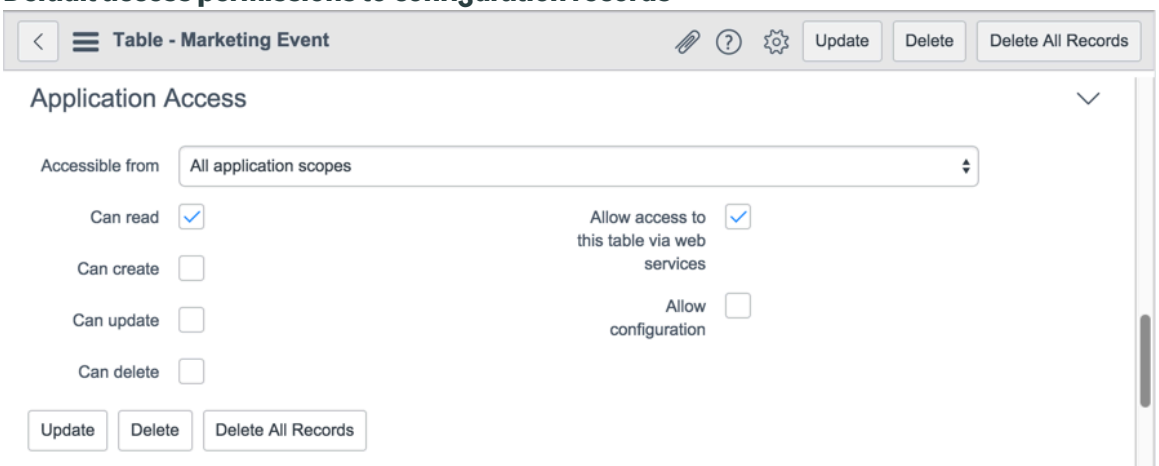
Default design access permissions

By default, new application tables prevent other application scopes from creating configuration records on application data tables. This prevents any other applications from changing the functionality of a table.

Default design-time access

Field	Value
Accessible from	All application scopes
Can read	Enabled
Can create	Disabled
Can update	Disabled
Can delete	Disabled
Allow access to this table via web services	Enabled
Allow configuration	Disabled

Default access permissions to configuration records



Set design-time access to application tables

Set these access permissions to protect application tables at design-time.

Before you begin

Role required: admin

About this task

To set runtime access permissions:

Procedure

1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. Click the button for the application type you want to edit.

Option	Description
Developed	displays applications created on this in stance.
Downloaded	displays applications downloaded on this in stance.

3. Click the application name or the **Edit** button for the application you want to work on.
4. From the Tables related list, select the table whose access permission you want to set.
5. From the Application Access section, set the design-time access permissions.
6. Click **Update**.

Related topics

Example denying all design access to a table

Example allowing configuration records for a table

Example denying all design access to a table

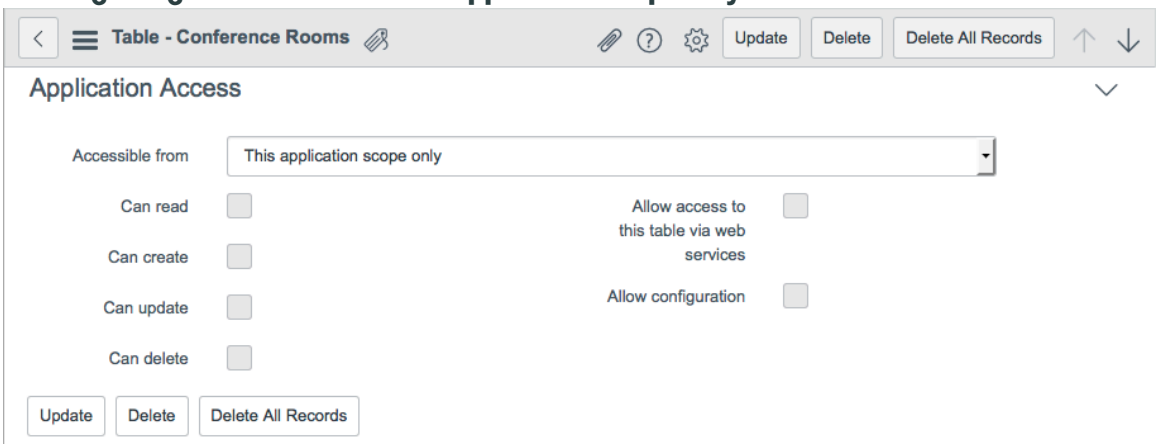
You can prevent other application scopes from creating configuration records on application data tables.

Typically, this is to prevent any other applications from changing the functionality of a table. Denying access requires setting the following value in the table record.

Denying all design-time access

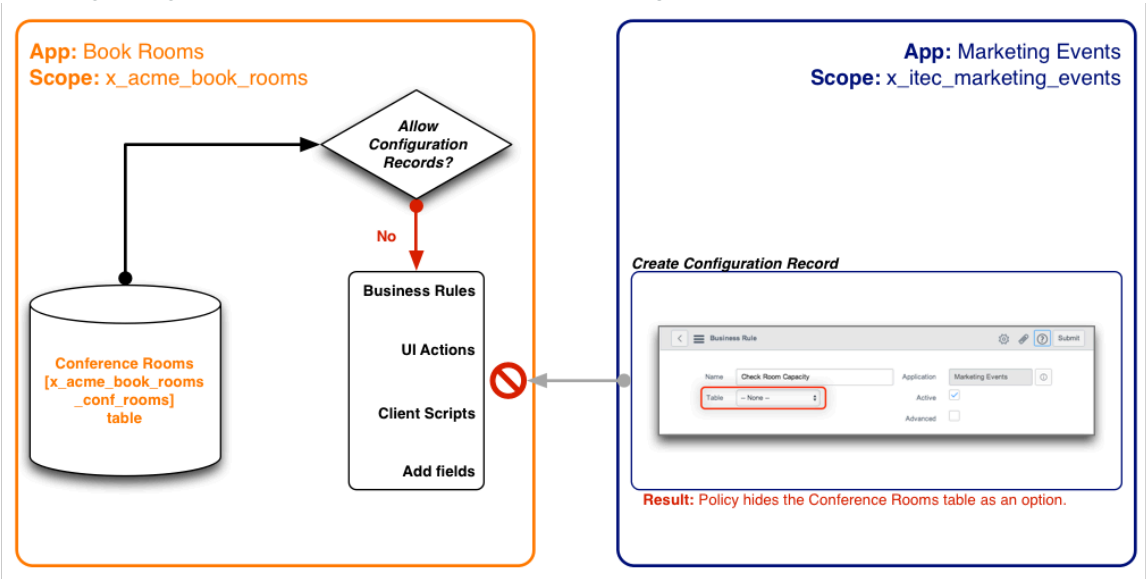
Field	Value
Accessible from	This application scope only
Can read	Disabled
Can create	Disabled
Can update	Disabled
Can delete	Disabled
Allow access to this table via web services	Disabled
Allow configuration	Disabled

Limiting design-time access to this application scope only



The following diagram illustrates the effect of denying other application scopes the ability to create configuration records.

Limiting design access to this application scope only



Example allowing configuration records for a table

You can permit other application scopes to create configuration records on application data tables.

You can grant access to the following configuration records with these settings.

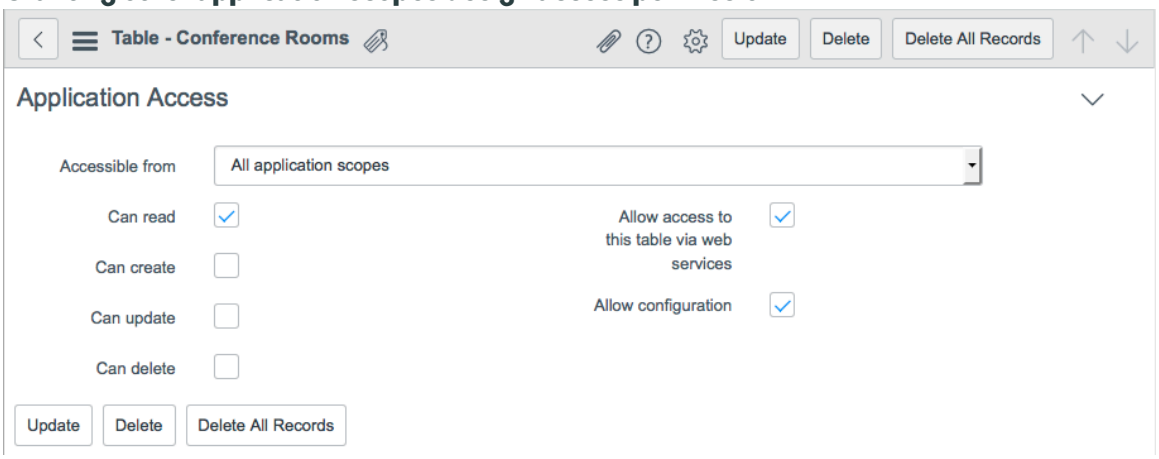
Granting access to configuration records

Configuration record	Setting required to grant access
Access controls	• **Accessible from** set to **All application scopes** • **Can read** is selected
Business rules	

Granting access to configuration records (continued)

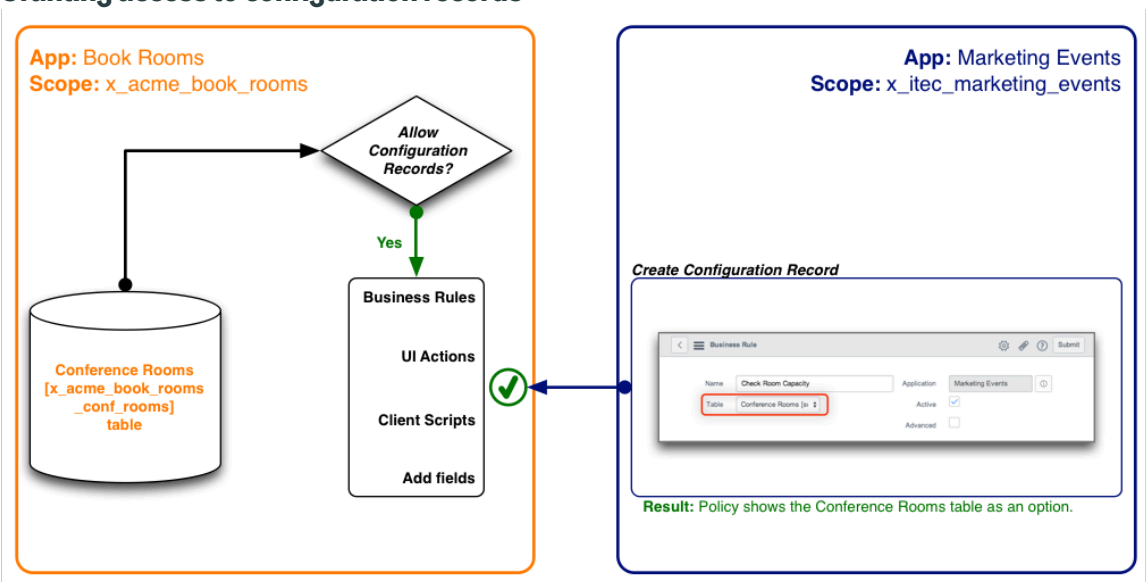
Configuration record	Setting required to grant access
Client scripts	• **Accessible from** set to **All application scopes** • **Can read** is selected • **Allow configuration** is selected
Dictionary entry (new field only)	
UI actions	

Granting other application scopes design access permission



The following diagram illustrates the effect of granting other application scopes the ability to create configuration records.

Granting access to configuration records



Restricted caller access privilege settings

Define cross-scope access to an application, application resource (such as an access control role, a business rule, a UI action, or a script include), or event. You can even use these settings to allow or deny requests for access.

Restricted caller access privilege settings overview

Restricted caller access [sys_restricted_caller_access] records track cross-scope applications or scripts that request access to an application, application resource, or event in the ServiceNow AI Platform. The ServiceNow AI Platform creates sys_restricted_caller_access records when one of these actions occurs:

- Caller access is set to **Caller Restriction** or **Caller Tracking**.
- A cross-scope script attempts to access an application resource or event.

> **Note:**
> A system scope to target scope is an example of a cross-scope.

You can use these records to do these tasks:

- Track cross-scope requests for access to an application resource. You can use access requests to determine which applications need access to resources and data from other application scopes.
- Approve or deny any cross-scope requests for access to application resources or events. For example, you can create a Restricted Caller Access record to allow access for all scope-to-scope requests.

For more information, see Requested restricted caller access (RCA).

Restricted caller access privilege setting combinations

You can define various combinations of privilege settings for restricted caller access and specify whether access is allowed or restricted for each relationship. This process ensures that your application has the privileges to access the correct scope and not one you don't want to access. You can define various combinations of the following settings:

Scope

All application resources in a selected source or target scope. To learn more about application scopes, see Application scope.

Source

A specific application resource in a selected source scope.

Target

A specific application resource in a selected target scope.

These restricted caller access privilege settings combinations include, but are not limited to following combinations:

- Scope-to-scope
- Scope-to-target
- Source-to-scope
- Source-to-target

> **Note:**
> For more information about these access setting combinations and to learn how to create each combination, see Set the application scope, application resource, and event access.

Activating application restricted caller access

You can activate application restricted caller access through one of the following methods:

- Activate the Scoped Application Restricted Caller Access plugin (com.glide.scope.access.restricted_caller).
- Request the HR Service Delivery or Security Incident Response applications. By default, restricted caller access is active in these applications.
- Enable the Restricted Caller Access system property for Workflow Studio.

For more information, see: Activate application restricted caller access.

Activate application restricted caller access

You can activate the Scoped Application Restricted Caller Access plugin (com.glide.scope.access.restricted_caller) if you have the admin role.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. Find the plugin using the filter criteria and search bar.

   You can search for the plugin by its name or ID. If you cannot find a plugin, you might have to request it from ServiceNow personnel.
3. Select **Install** to start the installation process.

   Note:
   When domain separation and delegated admin are enabled in an instance, the administrative user must be in the **global** domain. Otherwise, the following error appears: `Application installation is unavailable because another operation is running: Plugin Activation for <plugin name>.`

   You will see a message after installation is completed. For information about the components installed with a plugin, see Find components installed with an application.

Related topics

List of plugins (madrid)

Define cross-scope access to an application resource

Track cross-scope requests for access to an application resource and approve or deny requests.

Before you begin

If you enable application administration for the target application, only application administrators of the target application can set access to an application. If application administration is not enabled, an admin user can set access to an application.

Role required: admin or application admin

Note:
To learn about application-specific administrator roles and delegated development, see Access control rules in application administration apps and Delegated development and deployment.

Procedure

1. To define access to an application resource, navigate to the application resource record.
   Available application resources include these options.
   - Table
   - Script Include
2. Set the **Accessible from** field to **All application scopes**.
   If set to **This application scope only**, no other application scopes can access the resource.
3. Select the appropriate access level in the **Caller Access** field.

Option	Description
None	Cross-scope calls to the resource are approved or denied based on the value of the **Accessible from** field.
Caller Restriction	Calls to the resource must be manually approved. Access requests are tracked in the Restricted Caller Access table with a status of Re quested.
Caller Tracking	Calls to the resource are automatically approved. Calls are tracked in the Restricted Caller Access table with a status of Allowed.

4. Allow or deny an access request from a calling application.
   When a cross-scope application attempts to access a resource set to Caller Restriction, the system denies access to the resource and creates a record in the Restricted Caller Access table with a status of Requested. An admin user or application administrator must allow or deny the request. When access is allowed, all future access attempts gain access to the restricted resource.

   If a calling resource changes (such as when a business rule's script changes), the restricted caller access record status changes to Invalidated. An admin user or application administrator must update the status to Allowed or Denied.

   a. In the application record, navigate to the **Restricted Caller Access Privileges** tab.

   b. Review records that have a Status of Requested.
      Each Restricted Caller Access Privilege record lists the operation performed, information about the calling source, and information about the target resource requested.

   c. In the **Status** column, set the value from **Requested** to **Allowed** or **Denied**.

   Once a calling source is allowed, all subsequent calls are allowed.

Set the application scope, application resource, and event access

Create a record in the Restricted Caller Access Privileges [sys_restricted_caller_access] table to set cross-scope resource access requests. Approve or deny requests from a source scope or source scope application resources to a target scope or to target scope application resources.

Before you begin

If you enable application administration for the target application, only application administrators of the target application can set access to an application. If application administration is not enabled, an admin user can set access to an application.

Role required: application admin or admin

Note:
To learn about application-specific administrator roles and delegated development, see Access control rules in application administration apps and Delegated development and deployment.

About this task

You can set the following restricted caller access privilege settings combinations:

- Scope-to-Scope
- Scope-to-Target
- Source-to-Scope
- Source-to-Target

Note:
In the Rome release, we have enforced that an RCA privilege record must be present in the target application to grant access to a resource. This means that the target scope must match the application scope.

Procedure

1. Navigate to **All** > **System Applications** > **Application Restricted Caller Access**.
2. On the form, fill in the fields.

Restricted Caller Access fields

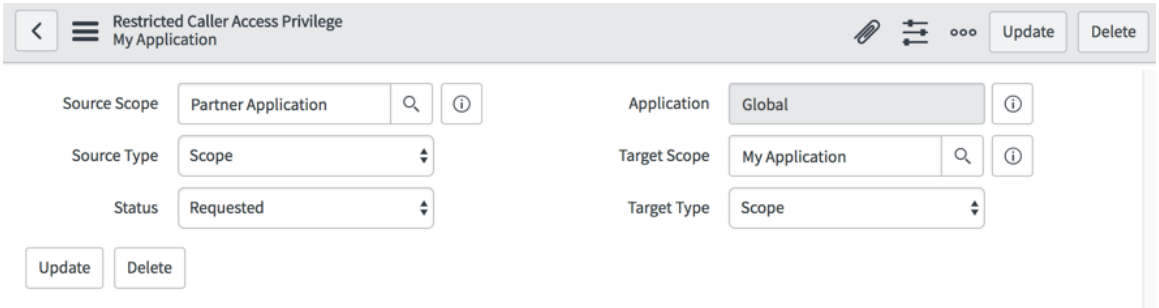
Field	Description
Operation	Operation performed on the target resource. ○ Read ○ Write ○ Create ○ Delete ○ Execute API
Source	Cross-scope record that is accessing a restricted application resource.
Source Scope	Scope of the calling application.
Source Table	Table that contains the Source record.
Source Type	Type of record that is calling the application resource: ○ ACL ○ Business Rule ○ Document Title ○ Flow ○ Flow Action ○ GlideScopedEvaluator ○ Inbound Email Script ○ Orchestration RunScript Activity ○ Record Producer Script

Field	Description
	○ Service Portal Widget ○ Scheduled Script ○ Scope ○ Script Include ○ UI Action ○ UI Macro ○ UI Page ○ Workflow Activity For example, to allow access from an entire application, select **Scope**.
Status	Status of the access request: ○ Requested ○ Denied ○ Allowed ○ Invalidated **Note:** If a calling resource changes, the restricted caller access record status changes to Invalidated. If you enable application administration, only application administrators of the target application can update the status of a request.
Target	Record of the requested resource.
Target Scope	Scope of the requested resource.
Target Table	Table that contains the Target record.
Target Type	Type of requested resource. ○ Event **Note:** An event is a special type of target for restricted caller access. By selecting an event in a target scope, you give a source application permission to queue an event that is registered as part of a target application. However, if you set the caller access on the event registry to None, it prevents cross-scope access calls to an event. This setting combination is a one-to-one relationship. To learn more about events, and their function, see Events. If you set caller access to None on the event registry, the cross-scope access calls to an event are denied. ○ Scope

Field	Description
	○ Table ○ Script Include For example, to allow access to an entire application, select **Scope**.

Scope-to-scope settings

Allow or deny access of all application resources in a source scope to all application resources in a target scope. This setting combination is a many-to-many relationship.



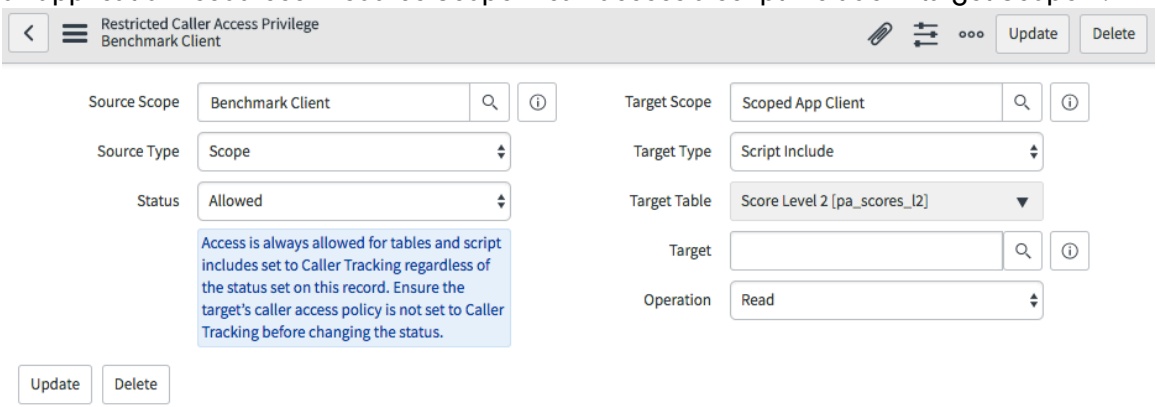
Enter the following field settings for Scope-to-Scope restricted caller access.

Field	Entries
Source Scope	Scope of the calling application that contains the source application resources.
Source Type	Type of record calling the application resource. To allow access from an entire application, select **Scope**.
Status	Status of the access request. Select **Allowed** to allow access, or select **Denied** to restrict access for this source-target resource relationship.
Target Scope	Scope of the requested resource that contains the target application resources that the source application resource requests access to.
Target Type	Type of requested resource. Select **Scope** to include all application resources in the target scope.

Scope-to-target settings

Allow or deny access of all application resources in a source scope to a specific application resource (business rule, table, script include, or event) in a target scope.

This setting combination is a many-to-one relationship. For example, you can specify that all application resources in source Scope A can access a script include in target Scope B.



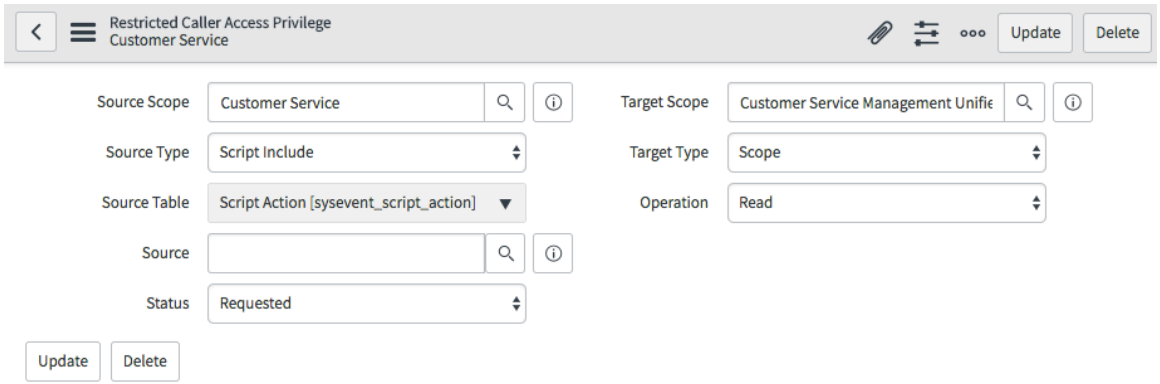
Enter the following field settings for Scope-to-Target restricted caller access.

Field	Entries
Source Scope	Scope of the calling application that contains the source application resources.
Source Type	Type of record calling the application resource. Select **Scope** to include all application resources in the source scope.
Status	Status of the access request. Select **Allowed** to allow access, or select **Denied** to restrict access for this source-target resource relationship.
Target Scope	Scope of the requested resource that contains the target application resources that the source application resource requests access to.
Target Type	Type of requested resource. Select the specific application resource (for example, business rule, script include, UI page, event) the source application resource requests access to.
Operation	Type of operation (for example, Read, Write) in the target application resource the source application resource requests access to.

Source-to-scope settings

Allow or deny access of a specific application resource in a source scope to all application resources in a target scope.

This setting combination is a one-to-many relationship. For example, you can specify that a particular business rule in source Scope A can access all application resources in target Scope B.



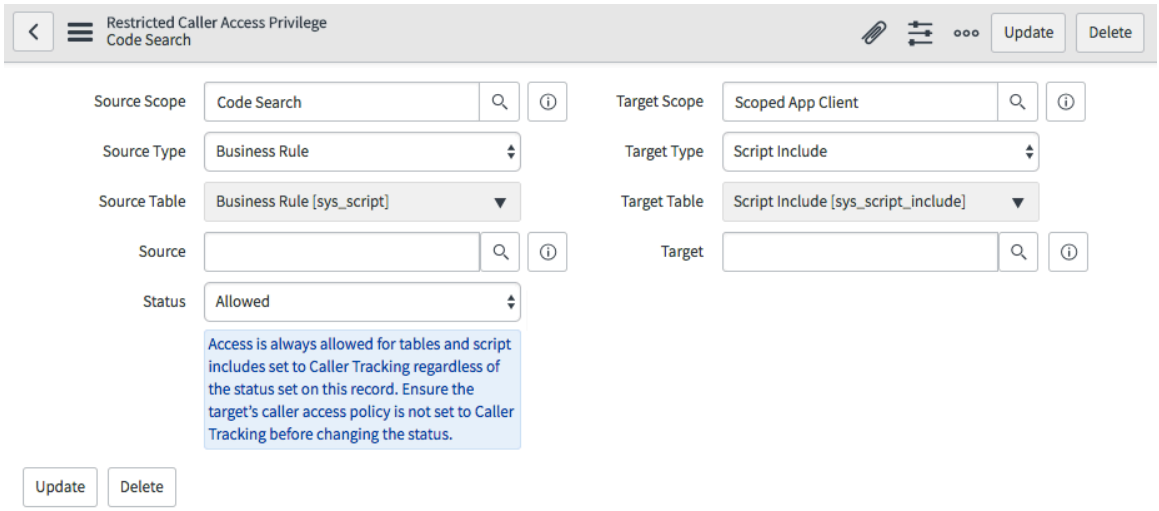
Enter the following field settings for Source-to-Scope restricted caller access.

Field	Entries
Source Scope	Scope of the calling application that contains the source application resource requesting access to the target scope application resource.
Source Type	Type of record calling the application resource. Select the specific application resource (for example, business rule or script include) requesting access to the specified target scope application resource.
Status	Status of the access request. Select **Allowed** to allow access, or select **Denied** to restrict access for this source-target resource relationship.
Target Scope	Scope of the requested resource that contains the target application resources that the source application resource requests access to.
Target Type	Type of requested resource. Select **Target** to include all application resources in the selected target scope.
Operation	Type of operation (for example, Read, Write) in the target application resource the source application resource requests access to.

Source-to-target settings

Allow or deny access of a specific application resource in a source scope to a specific application resource in a target scope.

This setting combination is a one-to-one relationship. For example, you can specify that a specific business rule in source Scope A can access a specific application resource, such as a business rule, table, script include or event, in a target scope.



Enter the following field settings for Source-to-Target restricted caller access.

Field	Entries
Source Scope	Scope of the calling application that contains the source application resources.
Source Type	Type of record calling the application resource. Select the specific application resource (for example, business rule, script include, or UI page) requesting access to the specified target scope application resource.
Status	Status of the access request. Select **Allowed** to allow access, or select **Denied** to restrict access for this source-target resource relationship.
Target Scope	Scope of the requested resource that contains the target application resources that the source application resource requests access to.
Target Type	Type of requested resource. Select the specific application resource (for example, business rule, Script Include, event) the source application resource requests access to.

Requested restricted caller access (RCA)

You can use a requested RCA to grant store apps access to protected resources in the ServiceNow AI Platform without the need to wait for the next family release. If you have the system admin or application admin role, you can review requested RCAs and approve and deny them.

RCAs are classified into two categories:

- Real RCA: sys_scope==target_scope
- Requested RCA: sys_scope!=target_scope

For example: A real RCA record is where the application scope and target scope match. A requested RCA is a record that is still awaiting approval for access to the target scope.

When you install an application, your scheduled jobs generate RCA records with the status of Requested in the target application for each requested RCA record that is packaged in the source application.

Note:
The jobs are generated once Upgrade Summary has run.

Example of how a store app accesses a table

Let's say that a store app called HR Integrations Framework wants to access an HR Core Case table. The table is in the business rule called Find Case in the Integration Service table.

To request access, the HR Integrations Framework app requires that an RCA privilege is packaged in its own scope as follows:

- sys_scope = HR Integrations Framework
- target = HR Core Case
- status = Allowed
- target_scope = Human Resource: Core
- source = Find Case

App development example for developers

When you are developing an application, real RCAs are generated with the status of Requested when the target has a caller restriction. If the target has caller tracking, the status becomes Allowed. The developer can review and finalize all the real RCA records that are required for the application to work. For example, those RCAs with a status of Allowed.

A developer can click the **Generate RCA Privileges in Current App** in the related links to generate requested RCAs that are packaged in the current application. Requested RCAs are synchronized with real RCAs, which means that if a real RCA is updated or deleted, a requested RCA is updated or deleted too.

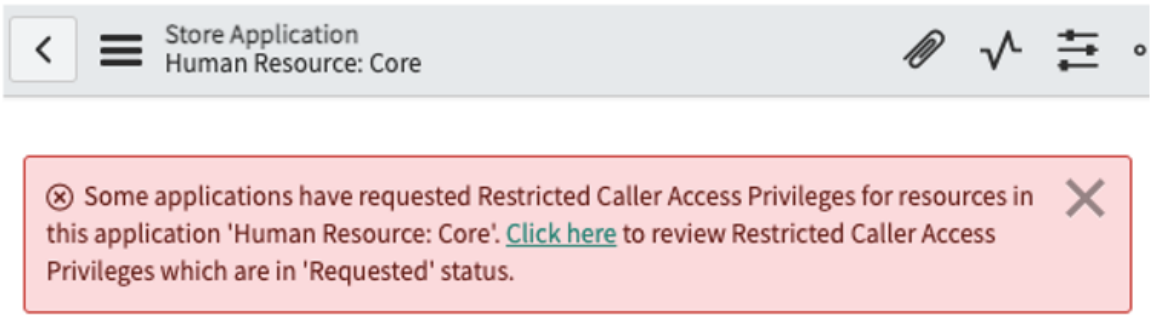
Now, the HR Integration Framework application can be packaged and installed on a customer instance.

App installation example for administrators

When you are installing an app on a customer's instance, real RCAs are generated in the target application. A real RCA would have the Human Resource: Core with a status of Requested. This process is done asynchronously in a scheduled job, where some lag time can occur.

To notify the target app admin about an RCA's pending review, messages have been added to application pages. An example is as follows:

RCA pending review message



Store App backward compatibility

If a store app is compatible and can be installed on an instance that is pre-Rome, then you must package the RCA records in their own scope with the status of Allowed.

Note:
This process ensures that the store app works on all versions.

When upgrading to Rome, you can configure a one-time fix script to move RCAs in the source scope to the target scope. In Rome, if the target app already has the necessary RCA records, no RCA records are generated for the RCAs that are packaged by the source app.

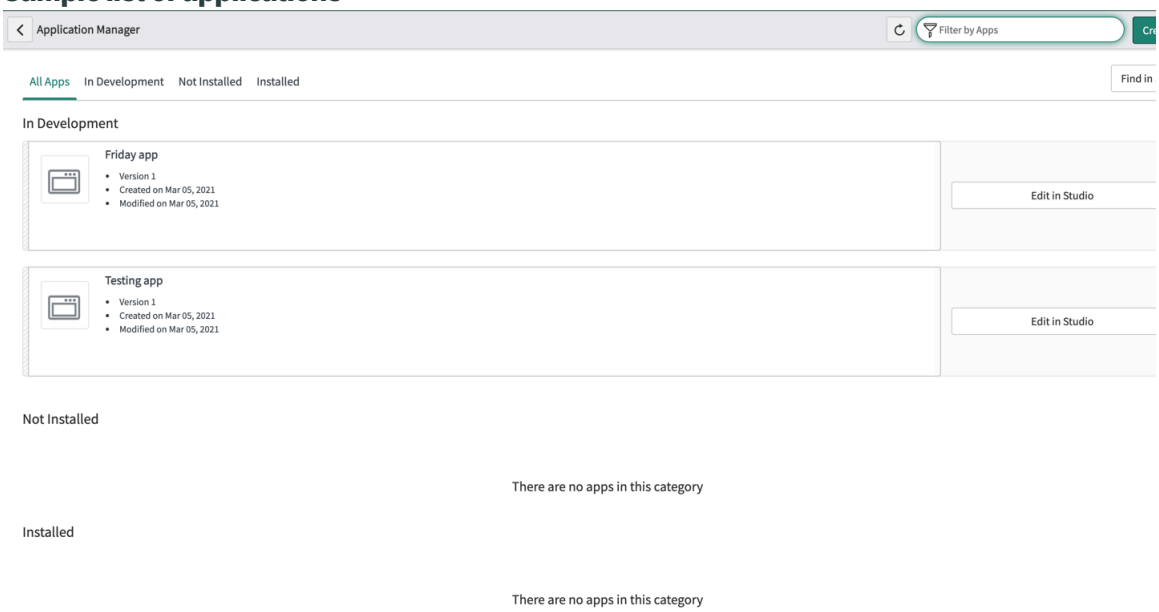
Application list

The applications list allows application developers to view and select applications.

Application developers can use the applications list to open a custom application record. If the contextual development environment detects that you are editing an application artifact in another application scope, it displays a warning message you can use to switch to another application.

Administrators have the following options from the applications list.

Sample list of applications



Application list options

Tab name	Options available	Application source
Develop	• Create new application • Edit existing application • Share application • Delete application	Applications developed on this instance

Application list options (continued)

Tab name	Options available	Application source
Downloads	• Install • View installed files • Edit installed files • Uninstall	• Applications shared on the company application repository • Applications shared on the ServiceNow Store
Updates	• Update • View version information	• Applications shared on the company application repository • Applications shared on the ServiceNow Store

The application list displays the following information.

Applications list

Type	Description
Developed	Display applications created on this instance.
Downloaded	Display applications downloaded on this instance.
Updates	Display available updates for downloaded applications on this instance.

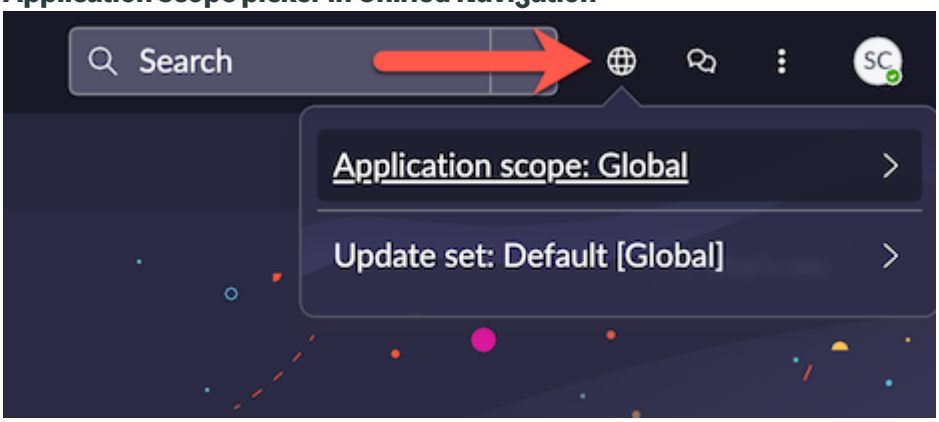
Application picker

The application picker enables application developers to view and select the application scope where their changes apply.

The application picker is available in the Unified Navigation menu. Select the picker icon ( ), then select **Application scope**, to open a list of application scopes that you have access to.

When your application scope is set to a non-Global scope, the picker icon displays with a red ring( ).

Application scope picker in Unified Navigation



Resources

The application picker is part of the Next Experience picker. For more information, see Exploring Next Experience pickers.

For more information about application scoping, see Application scope.

Lists and forms in scoped applications

The current application context determines what customization and form design options are available when working with lists and forms in scoped applications.

The user interface uses visual indicators to identify a list or form in the same or different application scope.

Available layout and design actions

The system allows the following layout and design actions when working on lists or forms in custom applications.

Available layout and design actions

Action	Access granted	Notes
Create a list view	Always allowed	This action is always available to users with access to customization.
Create a form view	Always allowed	This action is always available to users with access to customization.
Create a form section	Always allowed	This action is always available to users with access to customization.
Select fields to display in a view	Only allowed for sections in the view that match the current scope.	This restriction is independent of a user's role. Administrators cannot bypass this restriction.
Change the order of sections in a view	Only allowed for views that match the current scope.	This restriction is independent of a user's role. Administrators cannot bypass this restriction.
Select fields to display in a section	Only allowed for sections that match the current scope.	This restriction is independent of a user's role. Administrators cannot bypass this restriction.
Add or remove section columns	Only allowed for sections that match the current scope.	This restriction is independent of a user's role. Administrators cannot bypass this restriction.
Delete a form section	Only allowed for sections that match the current scope.	This restriction is independent of a user's role. Administrators cannot bypass this restriction.
Create new fields	Only allowed for sections that match the current scope and when the **Allow configuration** option is enabled.	This restriction is independent of a user's role. Administrators cannot bypass this restriction.

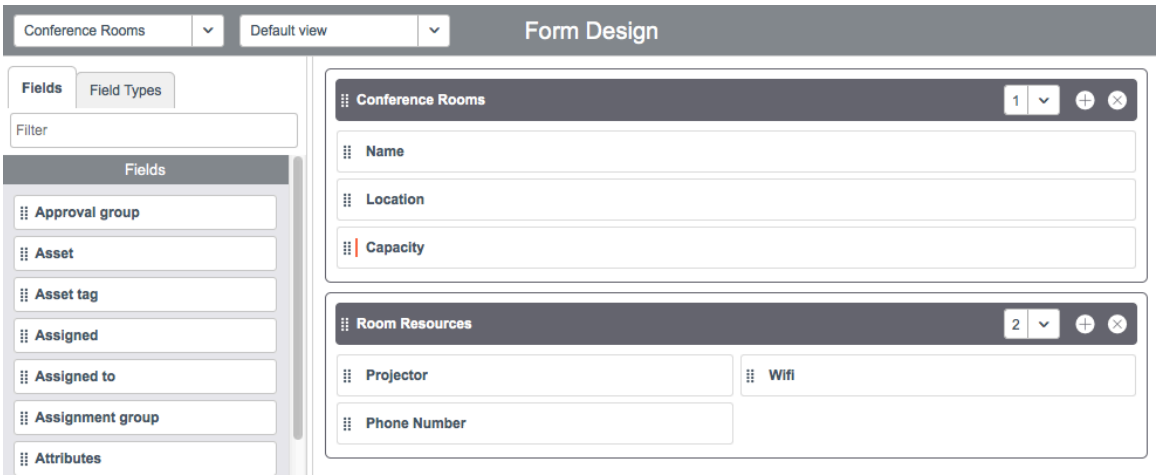
Form design visual indicators

The UI displays the following visual indicators when designing forms in custom applications.

You can only edit views and sections when you are in the same application scope as the form. Editable sections display:

- Section headings with a solid color background.
- A solid line around the section.
- A control to set the number of columns.
- A **Delete this section** button.
- Grip icons beside section headings.
- Grip icons beside fields.

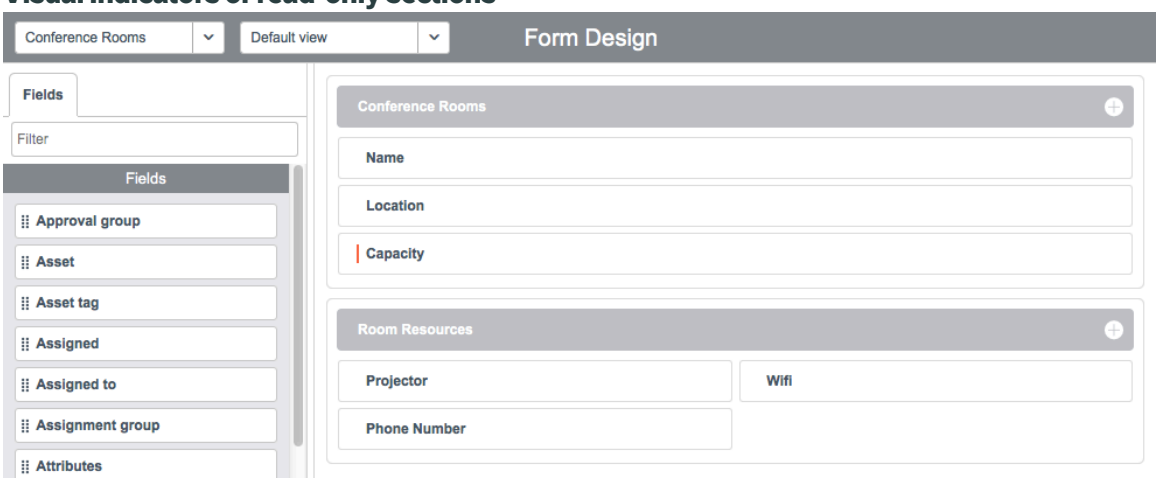
Visual indicators of editable sections



Views and sections in another application scope display as read only. Read-only sections have:

- Section headings with a gray background.
- A gray line around the section.
- No control to set the number of columns.
- No **Delete this section** button.
- No grip icons beside section headings.
- No grip icons beside fields.

Visual indicators of read-only sections



Default form design permissions

By default, new application data tables have the following form design permissions.

Default form design permissions

Form design action	Permission setting
Create new sections in tables belonging to another application scope	Allowed
Create new fields in sections belonging to the same application scope	By default, denied. Requires application designer to set **Allow configuration** for application table.
Add or remove fields from sections belonging to the same application scope	Allowed
Change the order of fields in sections belonging to the same application scope	Allowed
Change the order of sections belonging to the same application scope	Allowed
Add or remove fields from sections belonging to another application scope	Denied
Change the order of fields in sections belonging to another application scope	Denied
Change the order of sections belonging to another application scope	Denied
Create new fields in sections belonging to another application scope	Denied

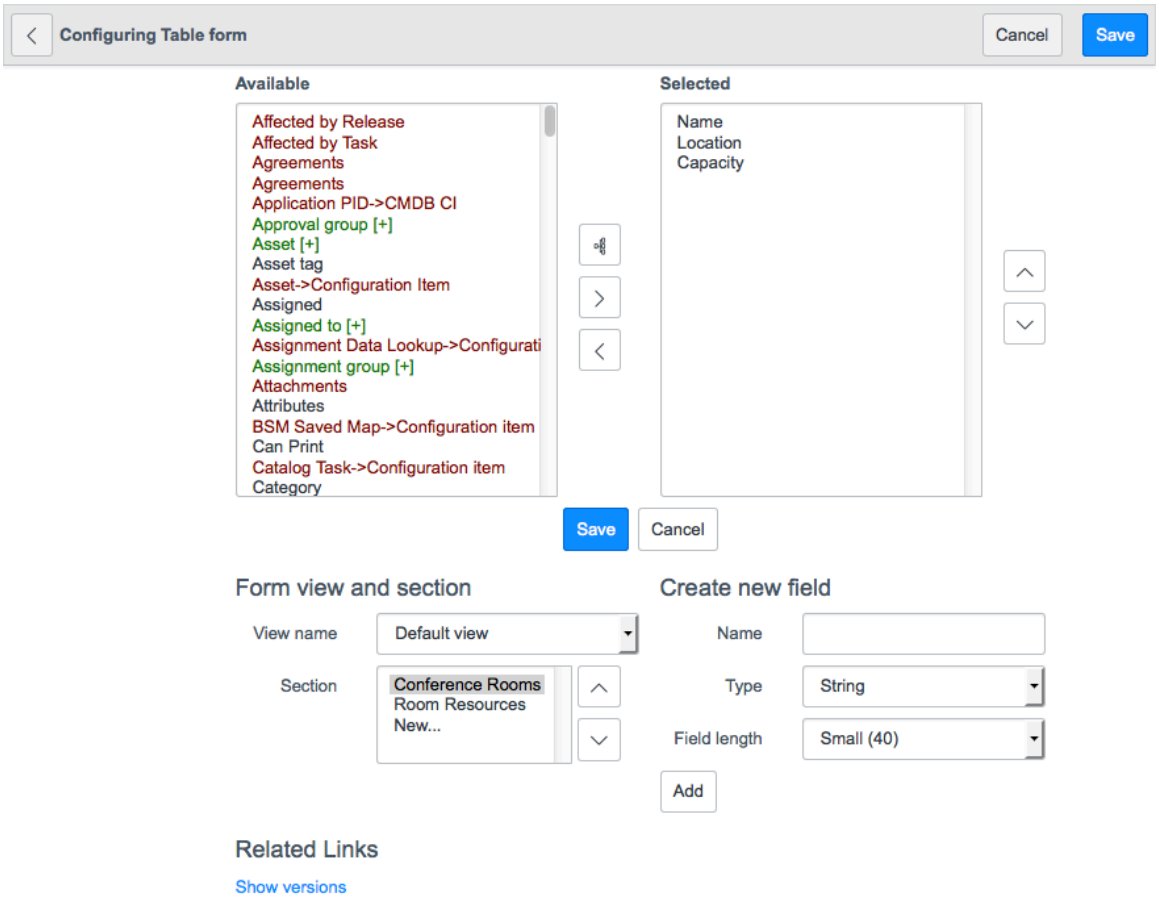
Layout visual indicators

The UI displays the following visual indicators when configuring the layout of a custom application's list or form.

You can only edit fields, views, and sections when you are in the same application scope as the form. Editable sections display:

- Field selections with a white background.
- Buttons to add or remove fields.
- Save button with a blue background.
- Editable fields in the **Create new field** section.

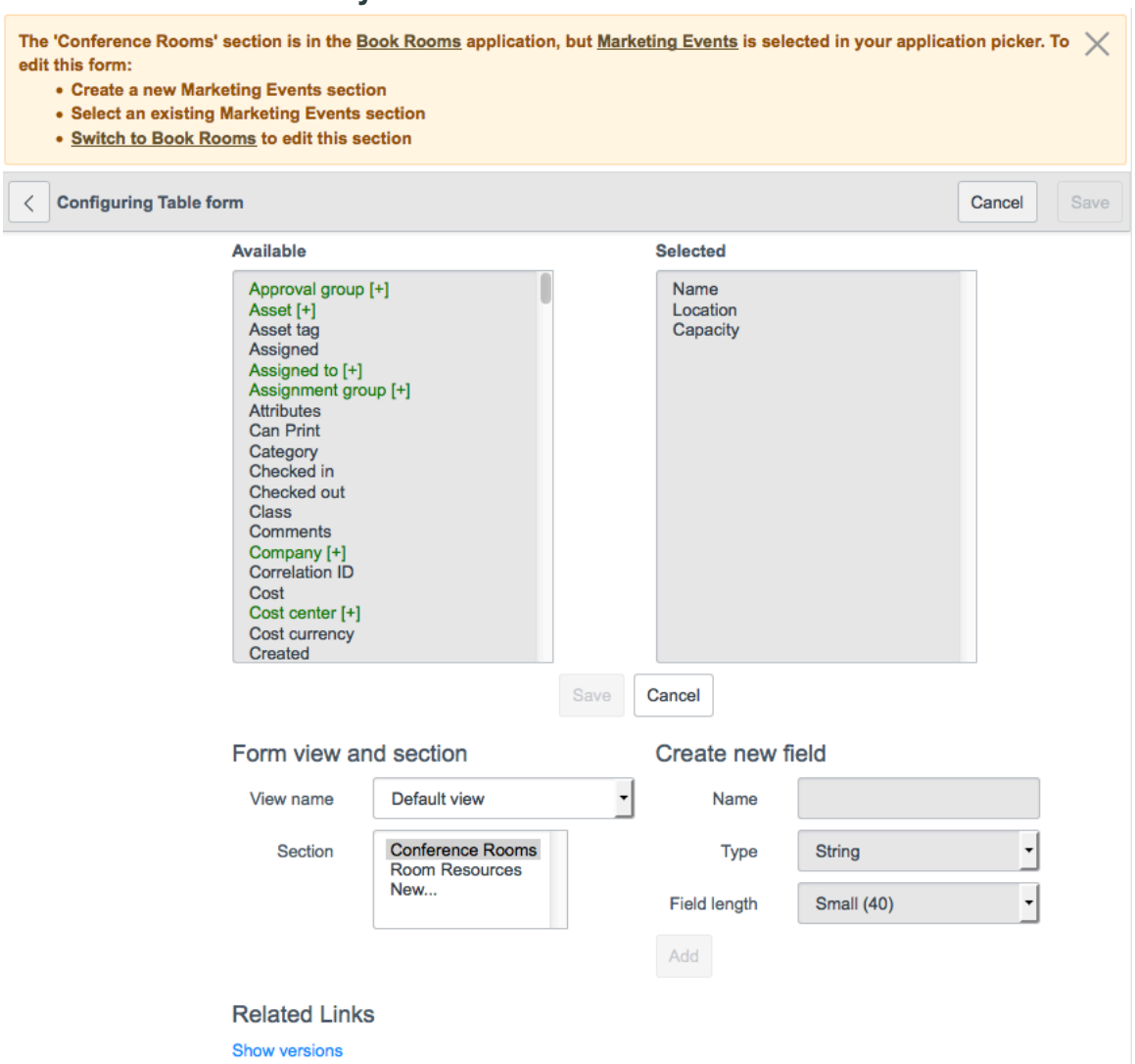
Visual indicators of editable sections



Views and sections in another application scope display as read only. Read-only sections have:

- A warning message about the currently selected application scope and the scope of the form.
- Field selections with a gray background.
- No buttons to add or remove fields.
- Save button with a gray background.
- Read only fields in the **Create new field** section.

Visual indicators of read-only sections



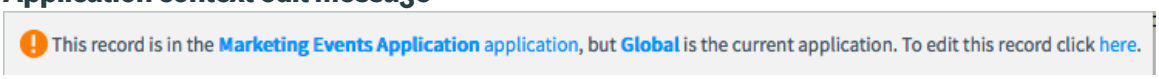
Related topics

- Lists and forms in scoped applications
- Available layout and design actions
- Form design visual indicators

Contextual development edit messages

The platform displays a message if you attempt to edit a Store Application record when you're in a different application scope.

Application context edit message



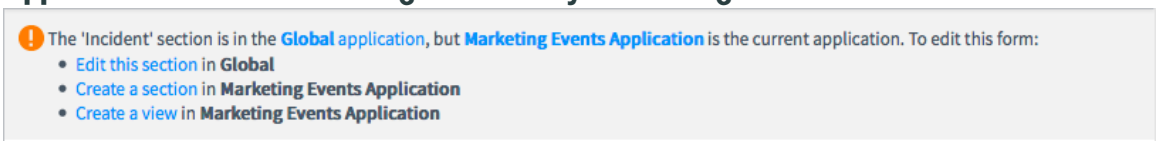
This message can be used to:

- Open the application to which the current configuration record belongs.
- Open the application of the currently selected application in the application picker.
- Temporarily switch to the application to which the current configuration record belongs and edit it.

Note:
The system returns you to the current application context after you save or cancel out of the record.

The system also displays a message when a user attempts to configure a list or form layout while working from another application scope.

Application context edit message for form layout or design



The message provides a list of valid options:

- Edit the current section by temporarily switching to the application that owns it.
- Create a new section in the current application context.
- Create a new view in the current application context.

Note:
The system returns you to the current application context after you save or cancel out of the record.

Script protection policy

Application developers can set a protection policy for script includes published as part of a custom application. The policy determines whether someone can view or edit the script include after the application is installed on their instance.

Application developers have these options to protect their custom application script includes:

Script include protection policy Options

Protection policy	Description
None	Select this option to allow other application developers to customize your script include.
Read-only	Select this option to allow other application developers to see your script logic, but not to change it.
Protected	Select this option to prevent other application developers from changing your intellectual property.

Application administration

Protect sensitive application data by using application administration to restrict how users acquire application-specific roles.

Functions of application administration

Use application administration to:

- Help prevent unauthorized users from accessing sensitive data via forms, lists and UI, such as financial records or personally identifiable information.
- Determine who can assign scope-protected roles such as the administrator and designated developers for the application.
- Help prevent users with the system-level admin role from:
  - Assigning themselves a protected application role.
  - Assigning themselves to a group containing a protected application role.
  - Bypassing existing access controls to a protected application by creating access controls.
  - Impersonating a user who has a protected application administration role, unless the developer or administrator also has that role.
  - Inheriting a protected application role.
  - Overriding existing access controls to a protected application.

Roles in application administration

You can make any role an application-specific administrator by selecting the **Application Administrator** check box in Role Configuration. To learn more, see Restrict access to an application. By convention, create the following roles:

Application administration roles

Role name	Description
Application-specific admin	Users with this role can assign other users to an application-specific role for that application. For example, you can create a role named my_application.admin. It should include the name of the restricted application, with a suffix of "admin" to indicate that it is the admin role for the application.
Application-specific developer	Users with this role can access the restricted application. For example, you can create a role named my_application.developer. It should include the name of the restricted application, with a suffix of "developer" to indicate that it is the developer role for the application. The developer role needs both application administrator and delegated development permissions to modify the application files. To learn more, see Delegated development and deployment and Delegate development and deployment permissions to personnel.

Application-specific admin role

The application-specific admin role enables a user to access a specific application, but does not grant the user any other admin rights. Assign the system-level admin role to a user before that user can do these tasks:

- Configure form and list layouts.
- Change application tables and fields.
- Assign the application-specific admin role to new users.

If you do not want a user with the application-specific admin role to have the system-level admin role:

- Do not assign the system-level admin role to the user. Assign only the application-specific admin role.
- Have the user assign themselves the application-specific developer role.

As an application-specific developer, the user can perform a subset of administrative tasks without having the system-level admin role.

Note:
Assign the application-specific admin role to more than one user. Then if a user with the application-specific admin role leaves the company, you are not prevented from changing the application.

Enabling application administration and assigning application-specific roles

You can enable application administration for an application from the application record and restrict the assignment of application-specific roles from the user role record.

Note:
Enable application administration and assign application-specific roles after completing development of the application, but before adding application records. This practice protects sensitive data in the application records from access by unauthorized users.

The target instance must have at least one authorized user with the application-specific admin role.

- If you enable application administration for an application but do not assign the application-specific roles, no user can access the application.
- If you assign only one application-specific role, you cannot delete that role.

A warning appears if you enable application administration for an application, but no users have the application-specific admin role required to assign roles for the application. The warning reminds you to assign the application-specific roles for administrators and developers of the application.

Set the *[scoped_app_name].min_admin_count property* to require that more than one user must have the application-specific admin role. Assigning the application-specific admin role to multiple users reduces the risk of getting locked out of the scoped application. The *[scoped_app_name].min_admin_count* property has the following limitations:

- If you specify an invalid value for the property, the default requirement for assigning at least one application-specific admin is enforced.
- If you specify a valid value for the property, you can't delete any application-specific admins unless you exceed the specified value. For example, if you specify a value of two and you have three application-specific admins, you can delete only one of those roles.
- You can specify a value higher than the actual number of assigned application-specific admins. However, you can't delete any application-specific admins until you exceed the specified value. For example, if you specify a value of six, but have only three application-specific admins, you can't delete any of those roles.

For procedures, see Restrict access to an application.

Deploying applications with application administration

You must have the system-level admin role in both your developer and production instances to deploy an application protected by application administration. The process is outlined in the following steps.

1. Develop the application on a development instance.
2. Create the application-specific admin role.
3. Grant the application-specific admin role to all system-level admin users.
4. Update the application record to enable application administration and restrict access to the application.
5. Publish the application to the application repository.
6. From a production instance, install the application from the application repository.
7. As a system-level admin on the production instance, grant the application-specific admin role to the appropriate users.
8. Remove the application-specific admin role from all users with the system-level admin role.

For procedures to enable application administration and restrict the assignment of application-specific roles, see Restrict access to an application.

Training

The ServiceNow® Developer Site has training for Securing Applications.

Related topics

Delegated development and deployment

Restrict access to an application

Restrict management of an application and access to that application to prevent unauthorized users from assigning administrative rights to the application or accessing sensitive information in the application records.

Before you begin

- Records required:
  - Role record to designate a role as the application-specific admin role
  - User record to assign an application-specific admin role to a user
  - Application record to enable application administration for a specific application
- Role required: admin

If an application-specific admin role does not already exist, create it before beginning this procedure. For example, you can create a role named my_application.admin that includes the name of the restricted application with the suffix "admin" to indicate that it is the admin role for the application.

Procedure

1. Navigate to **All** > **System Security** > **Users and Groups** > **Roles** or **User Administration** > **Roles**.
2. Open the role record for the application-specific admin role.
3. Configure the form to add the **Application Administrator** field.

   Note:
   The **Application Administrator** check box replaces the **Assignable by** field. By default, when you upgrade from a Jakarta or earlier release to a Kingston or later release, any role that was in the **Assignable by** field is defined as the application-specific admin role, and the **Application Administrator** check box is selected.

4. In the role record, select the **Application Administrator** check box, and then click **Update**.
5. Navigate to **System Applications** > **My Company Applications**.
6. Open the user record for the admin user.
7. On the Roles tab, add the application-specific role.
   Only users with the application-specific admin role can enable application administration for an application.

   Note:
   Assign the application-specific admin role to more than one user. Then, if a user with the application-specific admin role leaves the company, you are not prevented from performing changes to the application.

8. Click **Update**.
9. Log out and then log in with the application-specific admin role.
10. Navigate to **System Applications** > **Applications**.
11. Select the application for which you want to enable application administration.
12. In the application record, select **Application administration**.
13. Click **Update**.
    The system validates that the following requirements have been met:
    - The application has an application-specific admin role (there is at least one role with **Application Administrator** selected).
    - The current user has the application-specific admin role.

    If the validation passes, the system updates the application record. Otherwise, the system displays this error message and does not update the application record:

    ```
    Application Administration uses the 'Application Administrator'
     role to define what users are application administrators. None
     of the roles defined by this application have 'Application
     Administrator' enabled.
    ```

14. **Optional:** From the Related Links, you can select one of the following options:

Related Link	Description
Manage Developers	Modal that enables the application-specific admin to manage these tasks:

Related Link	Description
	○ Designate developers for the application. ○ Make themselves a delegated developer. After the application-specific admin becomes a delegated developer, the application-specific admin can perform a subset of administration tasks without having the system-level admin role. Learn more: Delegated development and deployment
Grant application administration to all admins	Modal that creates a Contained Role [sys_user_role_contains] record for the system-level admin role. This adds the application-specific admin role as a contained role of the system-level admin role. **Note:** When you publish the application with this record, users with the application-specific admin role can access the application after installing it.

Access control rules in application administration apps

By default, when application administration is enabled for a scoped application, ACL rules for the scoped application are applied. If no ACL rules for the scoped application are found, global ACL rules can apply.

This behavior applies to configuration records created in tables that extend the Application File [sys_metadata] table only. You can also change the default behavior.

When no access control (ACL) rules for an application administration app are defined, global ACL rules can apply to the configuration records of the application administration apps. See Application files for more information.

Note:
Not all applications utilize global ACL rules. If you are unable to perform an action as expected, verify that the application allows global ACL rules. If it does not, you may need to create specific ACL rules for some applications.

To allow a table in an application administration app to inherit global ACL rules, check that the system property is true and follow the instructions detailed in Configure a table in an application administration app to inherit global ACL rules.

- `glide.security.scoped_administration.honor_global_acl` system property: If no scoped ACL rules are defined, application administration apps can inherit global ACL rules. By default, this property is enabled for new and upgraded instances.
- Application Administration ACL Inheritances [sys_scoped_admin_acl_inheritance] table: If no ACL rules for the application administration app are found, tables added to this list inherit global ACL rules. Only the administrator for the application administration app can add, remove, read the records owned by the application administration app in this configuration table.

Configure a table in an application administration app to inherit global ACL rules

To avoid duplicating global access control rules (ACLs) in your applications, you can configure application file tables in application administration apps to inherit global ACLs when no ACL rules for the scoped application are found.

Before you begin
Role required: admin role for the application

Procedure

1. Enter `sys_scoped_admin_acl_inheritance.list` in the navigation filter.
   The Application Administration ACL Inheritance table [sys_scoped_admin_acl_inheritance] opens.
2. Click **New**.
3. In the **Table** field, select an application file table (a table that extends the Application File [sys_metadata] table) from the application administration app.
   Only the administrator for the application administration app can add/remove/read records owned by the application administration app in this configuration table.
4. Click **Submit**.
   If no ACL rules from the application administration app are found, tables added to this list inherit global ACL rules.

Access enforcement for ServiceNow Store apps

All production instances monitor and generate reports on usage patterns for ServiceNow Store apps. When subscription enforcement is enabled, users who are not subscribed to the app are blocked from performing fulfiller actions in the app.

Overview
The following actions are required to enable a production instance to enforce entitled usage of your ServiceNow Store App:

1. The usage admin at your organization uses the Subscription Management application to allocate fulfiller users to the subscription.
2. You decide on the enforcement mode, either:
   - Monitor and report usage with no enforcement (default)
   - In addition to monitoring and reporting usage, enforce that all usage must be by subscribed fulfiller users.
3. To enforce usage only by subscribed users, you configure the tables where only record owners or subscribed fulfiller users can make updates as fulfillment tables.

Licensing

Understand the different types of software licenses and subscriptions available with ServiceNow applications.

Licenses

Note:
Discuss licensing with your ServiceNow account representative for information specific to your contract.

You need a license to get entitlement for an application. If you have entitlement to a particular app, you can activate it on the ServiceNow Store.

All software licenses outline exactly how the software may be used without infringing on copyright law. There are many different subcategories of software licenses. These categories are not necessarily exclusive; different kinds of software will often overlap.

For more information on different types of software licenses, see What is a software license?

Subscriptions

A subscription enables you to upgrade from one family release to the next without dealing with the ServiceNow Store. For example, if you're on Tokyo and have a subscription to an app, you're automatically upgraded to Xanadu as soon as it's in general release.

The type of subscription determines the allocation of users, access to applications, and custom application and table entitlements. For more information on types of subscriptions available for ServiceNow applications, see Types of subscriptions in Subscription Management.

Creation restrictions across application scopes

The system restricts the creation of some configuration records when the current application scope does not match the application scope of the configuration record's target table.

Configuration record creation restrictions prevent one application from making unwanted changes to another application's data tables. These restrictions only apply when you create a configuration record whose target table belongs to another application. Configuration records that belong to the same application scope do not have these restrictions.

The system always enforces the following creation restrictions when a developer adds a configuration record belonging to another application scope.

Configuration record creation restrictions

Configuration record type	Creation restrictions when target table is in another application scope
Access controls	• You can only create field-level access controls with a role-based requirement. • You cannot create table-level access controls for a table in another application scope. • You cannot create field-level access controls that apply to all fields. • You cannot create access controls that use conditions. • You cannot create access controls that use a script-based condition.
Business rules	• You can create a rule where **When** is *async* with any of the following options: ○ *Insert*, *Update*, and *Delete* database operations. You cannot select *Query*. ○ **Set field values** actions and scripts (the **Script** field). • You can create a rule where **When** is *before* with any of the following options: ○ *Insert*, *Update*, and *Delete* database operations. You cannot select *Query*. ○ **Set field values** actions only. You cannot write scripts and you cannot abort the database transaction.
Calculated fields	You cannot create calculated fields for tables in another application scope.

Configuration record creation restrictions (continued)

Configuration record type	Creation restrictions when target table is in another application scope
Data policies	• You cannot create data policy rules for fields in another application scope. • You cannot make a field mandatory.
Field styles	You cannot create field styles for fields in another application scope.
Form sections	• You cannot modify existing form sections created in another application scope. • You can create new form sections.
Record producers	You must have create access to the application table to create records from a record producer.
UI policies	• You cannot create UI policy rules for fields in another application scope. • You cannot make a field mandatory.
UI script	You cannot create a global UI script from a scoped application.
Views	• You can create new views. • You cannot modify existing views created in another application scope.

Planning your application

The application development process starts with planning. Consider how the application will work, who will use it, and how it improves user experience.

Planning on the Next Experience

App intake

With your carefully planned idea in hand, you're ready to submit it through Application Intake for approval and development in App Engine Studio (AES).

Define tables and fields for application records

Applications use tables and records to manage data and processes, such as Incident, Problem, and CMDB.

Security: App Engine Studio roles and permissions

Administrators assign roles to give team members permission to configure or use AES.

Planning on UI16

Delegated development

Delegated development enables designated users without a system admin role to develop or deploy applications on the ServiceNow AI Platform.

Schema map for tables

The schema map displays the details of tables and their relationships in a visual way, enabling administrators to view and easily access different parts of the database schema.

Table Administration

Administrators can use Table Administration to view and modify the database structure.

Team Development

Team Development supports parallel development on multiple, non-production ServiceNow instances.

Related applications and features

Security: Access control

By default, when application administration is enabled for a scoped application, access control (ACL) rules for the scoped application are applied. If no ACL rules for the scoped application are found, global ACL rules can apply.

Security: Edge Encryption

ServiceNow® Edge Encryption encrypts sensitive data on your company premises before sending it over the internet to your ServiceNow instance (encrypted in flight), where it remains encrypted at rest.

Security: Roles

Set up the security and roles for your UI Builder instance. Security and roles in UI Builder are controlled through your application's scope, domain separation, and protection policy settings.

Security: UI policies

Create a UI policy to define custom process flows for tasks.

Security: Users and groups

Create users and groups using the ServiceNow AI Platform user administration feature.

ServiceNow application repository

After you develop and test a custom application, you can make the application available to company instances by publishing it to the ServiceNow application repository.

Source Control integration

Enable application developers to integrate with a Git Source Control repository. Save and manage multiple versions of an application from a non-production instance.

Submit your idea for app development

With your carefully-planned idea in hand, you are ready to submit it through Application Intake for approval and development in App Engine Studio (AES).

Before you begin

If you have not already activated the Apply for Citizen Development standard catalog item, you must do so before you can perform this procedure.

Role required: none

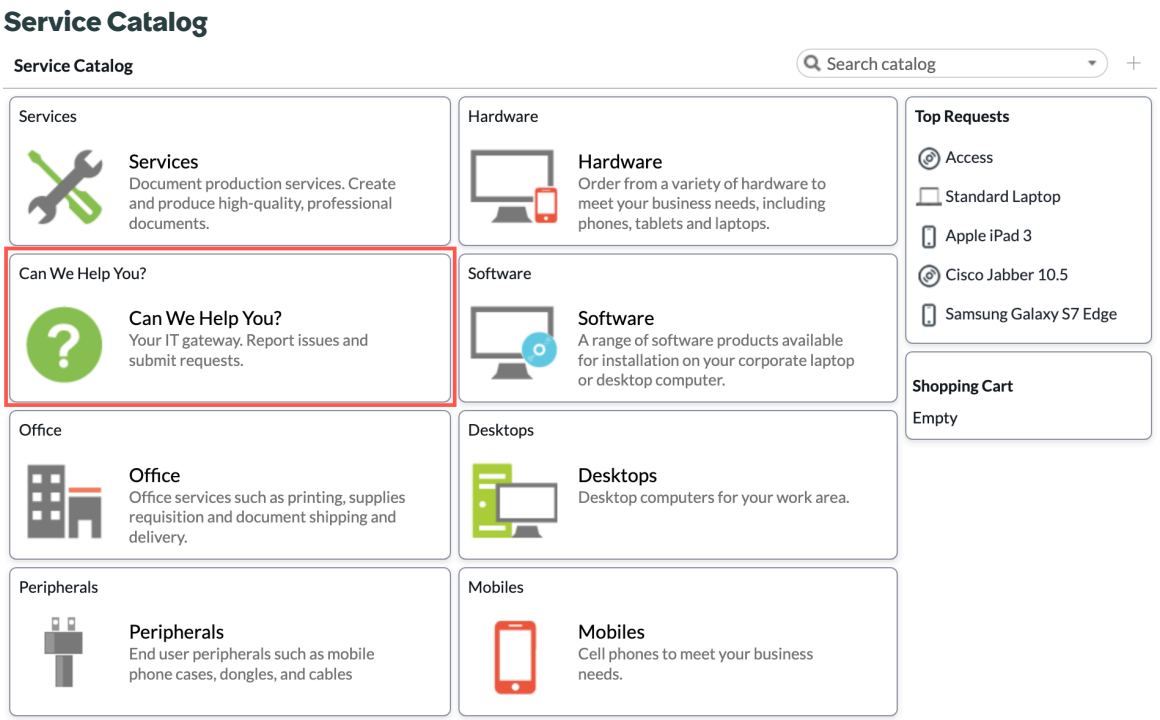
About this task

The intake request process streamlines submitting ideas for app development. Using Application Intake, the idea is shared with your App Engine Studio admin for review. The admin either approves or denies the request based on how well you have sold the business value of the idea.

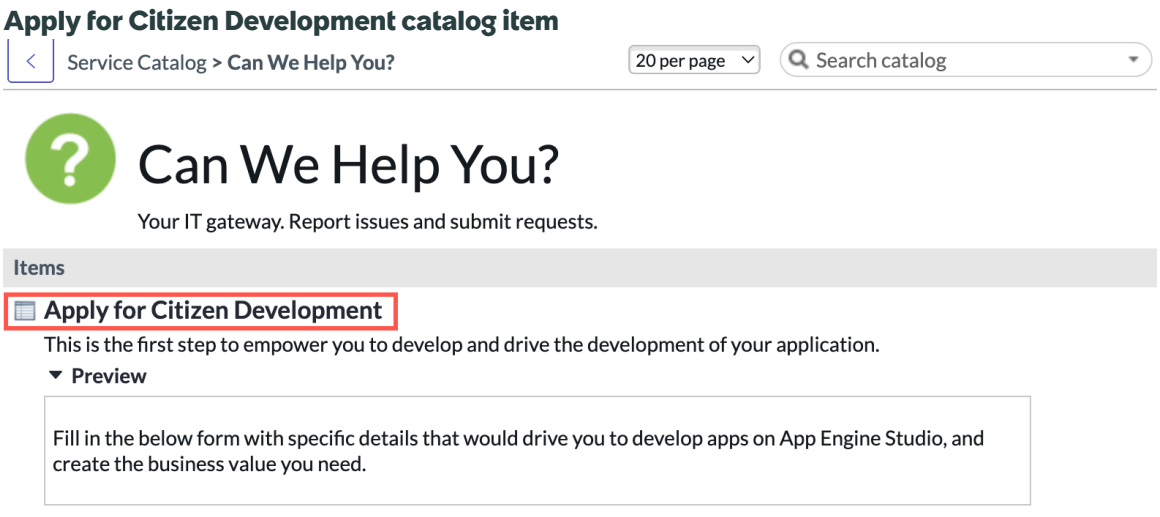
If the idea is approved, developers can build the app with the guided, intuitive app development environment provided in App Engine Studio.

Procedure

1. Navigate to **All** > **Self-Service** > **Service Catalog**.



2. In the Service Catalog, select the **Can We Help You?** category.



3. Select the **Apply for Citizen Development** catalog item to open the Application Intake form.
4. On the form, fill in the fields.

Fields on the Apply for Citizen Development form

Field	Description
Application Name	Unique name for the app.
Describe your idea in a few sentences	Simple description of the application.
Is your process repeatable?	Option to indicate whether the app idea could be used in other areas of the business.
Do you have an email or excel-based process?	Option to indicate whether an existing process is in place that this application would replace.
How many users are involved in this?	Number of users who would use the resulting application.
Does this involve any sensitive/PII data?	Option to indicate if the application would use any of the users' health, financial, or personally identifying information.
Do you need data from other departments?	Option to indicate whether other departments must supply data for the application.
Who are the users that will have access to the data in this application?	Users or groups who need access to the application.

5. Select **Order Now**.

Result

If the admin approves your request, you can start building your app in App Engine Studio. You should receive a confirmation email with access to build the app.

If you are an admin who approves requests, learn more about managing AEMC requests.

Specify data for your application

In its most basic form, building applications on the ServiceNow AI Platform means you're storing and using data in some way. Defining your application's data model is up to you and is based on the type of application you want to build.

You can use any combination of the following three methods to define your application's data model.

- **Reuse existing tables and fields to store record data.** For example, use the existing Task table to work on existing record types such as incidents and problems. Typically, this option is only available for tables in the global scope or tables that make their data available to all application scopes.
- **Extend existing tables to store record data.** For example, extend the Task table to create VIP incidents. The extended table inherits the fields from the parent table, but you can add new fields as needed.
- **Create new tables and fields to store record data.** You can use a variety of available tools to create new tables. If you have access to App Engine Studio, use Table Builder to create new tables and fields as you build applications. Alternatively, you can create tables and fields separately from application development using basic Table administration in the ServiceNow AI Platform.

More resources

These concepts are explained in further detail in the App Engine Studio, Table Builder, and Table administration documentation.

Create a data model for your application

Add data to apps that you build in App Engine Studio.

Table administration

Table administration currently contains more comprehensive functionality and options for managing tables.

Table Builder

Table Builder offers an intuitive way to visually build tables and forms in the ServiceNow AI Platform.

Delegated Development

Delegated development allows designated users without a system admin role to develop or deploy applications on the ServiceNow AI Platform.

Get started

Explore	Configure	Using
		
Learn about Delegated Development	Configure Delegated Development	Using Delegated Development

Troubleshoot and get help

- Ask questions and explore other resources for Delegated Development in the ServiceNow Community
- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

Exploring Delegated Development

Delegated development allows designated users without a system admin role to develop or deploy applications on the ServiceNow AI Platform.

Delegated Development overview

If you have the application-specific admin role or the system-level admin role, you can delegate application development to designated developers at the application level.

Delegated Development users

Users

User	Description
Admin	Admin user can delegate developers.
Application Administrator	If Application administration is enabled, only an application administrator of the target application can delegate developers to an application. If application administration is not enabled, an admin user can delegate developers.

What to explore next

To learn more about configuring and using Delegated Development, see:

- Configuring Delegated Development
- Using Delegated Development

Delegated development and deployment

Delegated development allows designated users without a system admin role to develop or deploy applications on the ServiceNow AI Platform.

If you have the application-specific admin role or the system-level admin role, you can delegate application development to designated developers at the application level.

Delegated deployment tasks

You can also delegate deployment tasks (application publishing, first-time installation, or update) to developers or non-admin users, such as Change Management personnel. You delegate deployment tasks to specific users at the application level, or through assignment of specific user roles at the instance level.

Assignment Method	Applies to	Available options
Setting deployment permissions in Manage Developers. See Delegate development and deployment permissions to personnel	Specific applications	Publishing and upgrades of specific applications. Publishing options include the application repository, ServiceNow Store, and update sets.
Assignment of deployment user roles to specific persons. See Instance-specific deployment user roles.	Local non-production instance (for example, Development or QA)	First-time installations and upgrades of all applications that contain the same company as the current instance. For example, applications for ABC Company and XYZ Company display on the Application Client page. A user with this role can only install XYZ Company applications when logged in to a XYZ Company

Assignment Method	Applies to	Available options
		instance. The user cannot install applications for ABC Company.

Application-specific permissions

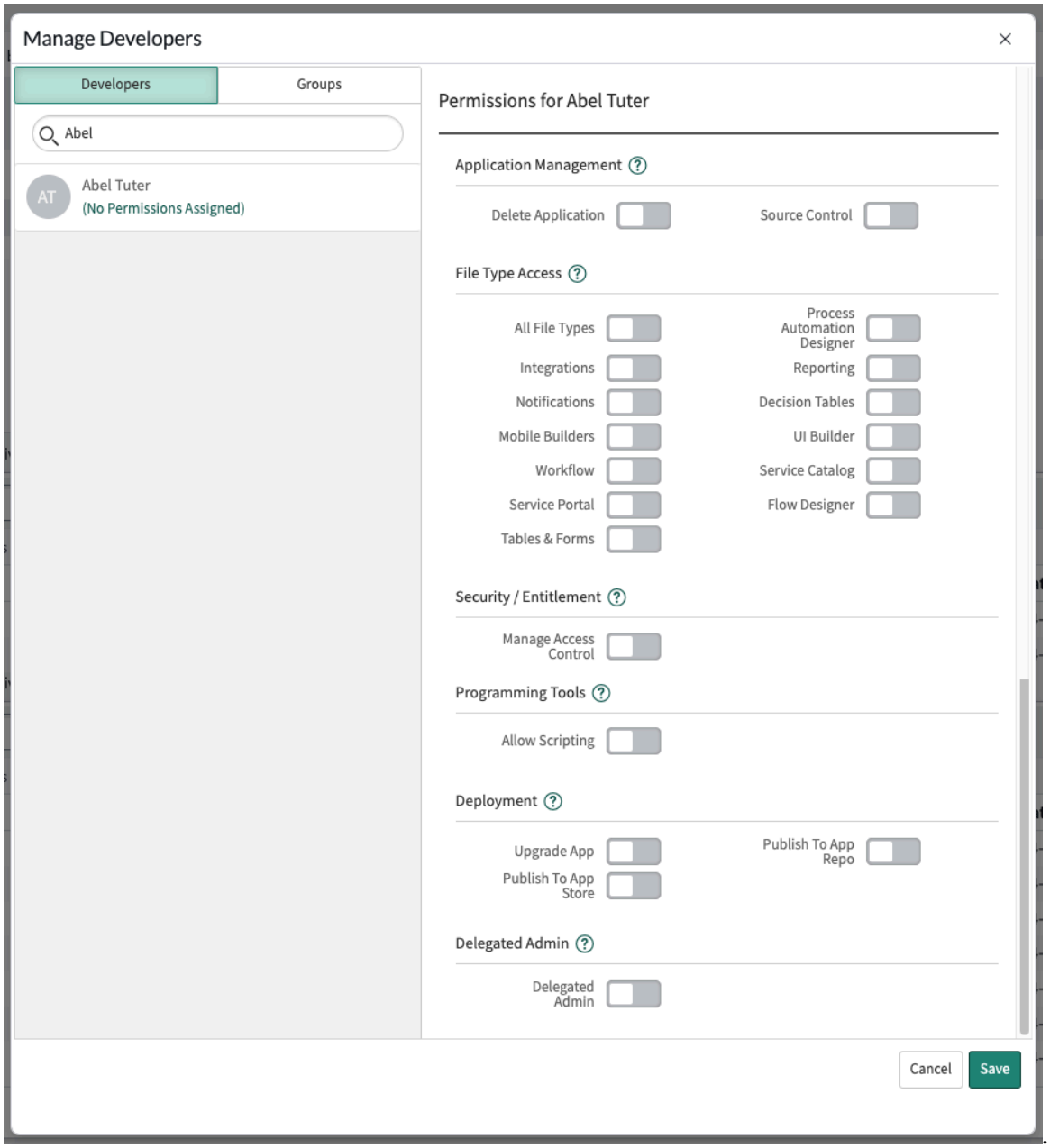
Developer and deployment permissions are application-specific. For example, a developer who has permission to access all file types for one application does not necessarily have any developer permissions for another application. Administrators must set developer (and optionally deployment) permissions for each application. Administrators must be familiar with application files and the system table structure to set developer permissions. For example, a developer expected to create advanced business rules needs both the **All File Types** and **Allow Scripting** developer permissions.

Important:
If Application administration is enabled, only an application administrator for the target application can delegate developers for an application. Application administrators do not have system admin privileges. To enable a delegated developer to perform the functions granted in the developer permissions, the delegated developer must also be given the application administrator role.

Setting each permission grants one or more system-managed delegated development roles, allowing system admins to retain control over the system. System admins no longer have to elevate developers (or users who deploy applications) to the system admin role to enable them to develop or deploy applications.

Developer and deployment permissions example

As a system administrator, you want to assign Abel Tuter certain developer and deployment permissions for a specific application in your development instance. For more details on developer and deployment permissions, see Delegate development and deployment permissions to personnel.



Related topics

- Application files
- Install a ServiceNow Store application
- Install an update to a ServiceNow Store application
- Application sharing

Domain separation and Delegated Development

Domain separation is unsupported in the Delegated Development feature. Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Support level: No support

- The domain field may exist on data tables but there is no business logic to manage the data.
- This level is not considered domain-separated.

For more information on support levels, see Application support for domain separation.

Related topics

Domain separation for service providers

Configuring Delegated Development

Configure Delegated Development to assign, delete, display or hide permissions.

Configuration overview

- Assign source control permissions

  Ability to assign full access to source control for a particular scope to a delegated developer. The Source Control menu is only visible if you have the correct permissions for the application that you are working in. Before you begin.

- Assign delete permissions

  Assign the ability to delete an application to a delegated developer.

- Display or hide update set deployment permissions

  Display or hide deployment permissions for update sets from the Manage Developers dialog.

- Instance-specific deployment user roles

  Assign roles that enable non-admin users install or upgrade all applications in specific instances. You delegate these tasks by manually assigning specific user roles per instance.

Display or hide update set deployment permissions

Display or hide deployment permissions for update sets from the Manage Developers dialog.

Before you begin
Role required: admin

About this task
System properties control the visibility of the **Publish To Update Set** and **Manage Update Sets** permissions. By default, both deployment permissions for update sets are hidden.

Procedure

1. In the Navigation filter, enter `sys_properties.list`.
   The entire list of properties in the System Properties [sys_properties] table appears.

   Note:
   For details on how to add and enable system properties, see Add a system property.

2. Enable or disable specific deployment permissions by adding or updating the following system properties.

Deployment permissions

System property	Description
com.snc.dd.manage_update_set_enabled	Enables or disables display of the **Manage Update Set** permission. The default value of this system property is **false**. To enable the display of this permission, set this value to **true**.
com.snc.dd.publish_to_app_repo_enabled	Enables or disables display of the **Publish To App Repo** permission. The default value for this system property is **true**. To disable display of this permission, set this value to **false**.
com.snc.dd.publish_to_app_store_enabled	Enables or disables display of the **Publish To App Store** permission. The default value for this system property is **true**. To disable display of this permission, set this value to **false**.
com.snc.dd.publish_to_update_set_enabled	Enables or disables display of the **Publish To Update Set** permission. The default value for this system property is **false** by default, which disables display of this permission. To enable the display of this permission, set this value to **true**.
com.snc.dd.upgrade_app_enabled	Enables or disables display of the **Upgrade App** permission. The default value for this system property is **true**. To disable display of this permission, set this value to **false**.

Instance-specific deployment user roles

Assign roles that enable non-admin users install or upgrade all applications in specific instances. You delegate these tasks by manually assigning specific user roles per instance.

For example, you can assign user roles to Change Management personnel that allow them to perform application installations in non-production (development or QA) instances.

Important:
Give careful thought to whom, and in what instances you assign installation and upgrade user roles. You may only want to assign these roles to non-system administrators in non-production instances (for example, Developer or QA), but decline to do so in production instances. Given the potential impact of installing and upgrading applications, you may want to leave the responsibility for installations and upgrades in production instances to a system administrator.

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

Add deployment user roles

A system administrator can assign user roles to specific personnel that allow them to perform first application installations only, or install and upgrade applications in a local instance.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **User Administration** > **Users** and then open a user record.

   Note:
   For details on how to assign a role to a user, refer to Assign a user role.

2. In the **Roles** related list, select **Edit**.
3. In the **Collection** list, select the desired deployment roles, and then select **Add**.

   Delegated deployment roles

User Role	Description
sn_appclient.app_client_company_installer	Allows a designated person to perform first-time installations of applications displayed on the Application Client page that contain the same company as the current instance. A user with this role cannot install an application for another company. For example, applications for ABC Company and XYZ Company display on the Application Client page. A user with this role can only install XYZ Company applications when logged in to a XYZ Company instance. The user cannot install applications for ABC Company. **Note:** While this role does not grant a user the ability to upgrade applications, the system auto-adds a delegated developer upgrade role after the user does a first-time installation. The user can then perform upgrades for the application in the local instance. For more details, see System-managed developer and deployment roles.
sn_appclient.app_client_user	Allows a designated person to install and upgrade all applications displayed on the Application Client page.

4. Select **Save**.

System-managed developer and deployment roles

Although system admins can still manually assign and remove the user roles, they are encouraged to let the system manage the following delegated developer roles.

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

Role	Description
delegated_developer	User has one or more developer permissions.
Roles that start with an sn_dd prefix (for example, sn_dd_<app_name>_upgrade_app)	User has an application-specific developer permission. The role name indicates the application scope to which i applies. For example, after a user with a sn_appclient.app_client_company_installer role installs a company application, the system automatically grants a sn_dd_<app_name>_upgrade_app delegated deployme role. This role allows the user to upgrade the application when future updates are published to the Application Client page.
glide.security.add_admin_contained_roles_to_system	Default Value: true if the property is true, then all the roles, directly or indirectly contained by the admin role, are added to the system user, including the scoped-admin roles. **Note:** Setting the property to false results in the old behavior, where the system user has the admin role, but not any of the scoped-admin roles contained by the admin role. Example: The admin role contains the *sn_templated_snip.template_snippet_adm* role Old behavior: The system user does not have the *sn_templated_snip.template_snippet_adm* role. New behavior: The system user has the *sn_templated_snip.template_snippet_adm* role and other scoped roles that it contains.

Note:
Users with delegated developer roles cannot add or remove the system admin role.

Assign source control permissions

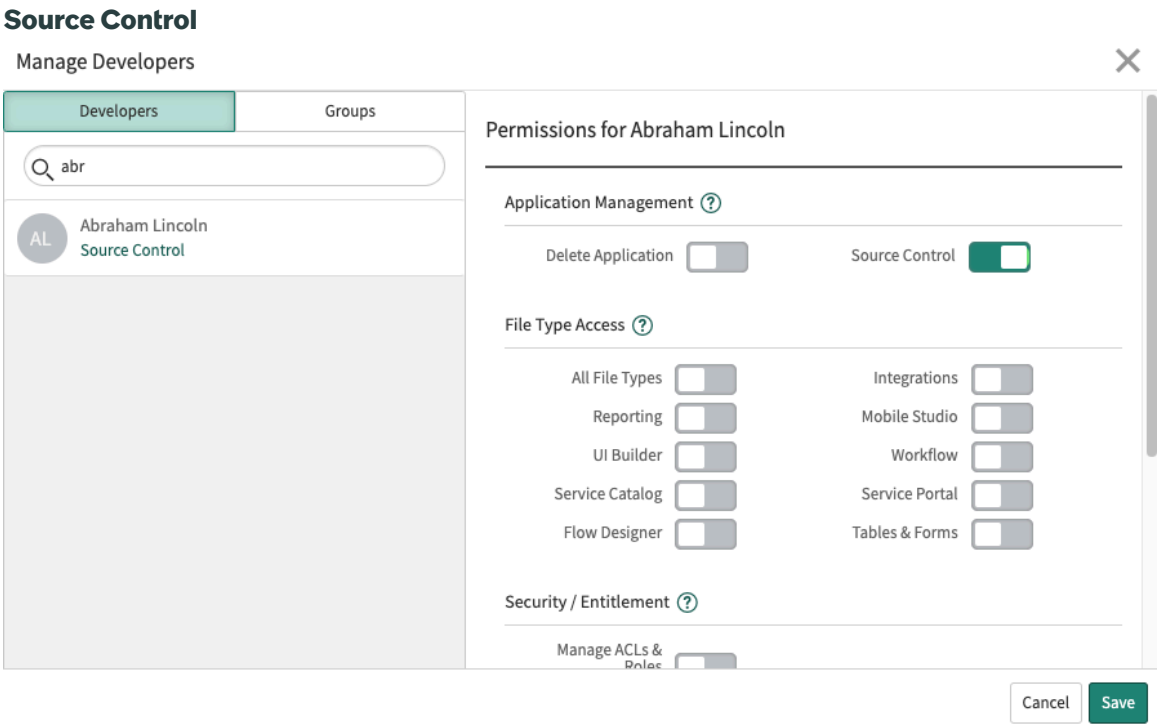
Ability to assign full access to source control for a particular scope to a delegated developer. The Source Control menu is only visible if you have the correct permissions for the application that you are working in.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. Open an application for which you want to assign delegated developer permissions.
3. As a System Admin, select **File** > **Manager Developers**.
4. Choose a developer to apply delegated developer permissions.
5. Toggle on **Source Control**, and then select **Save**.



6. The delegated developer can now access the Source Control menu options.
   For more information on source control options, see Legacy - Available source control operations.

Assign delete permissions

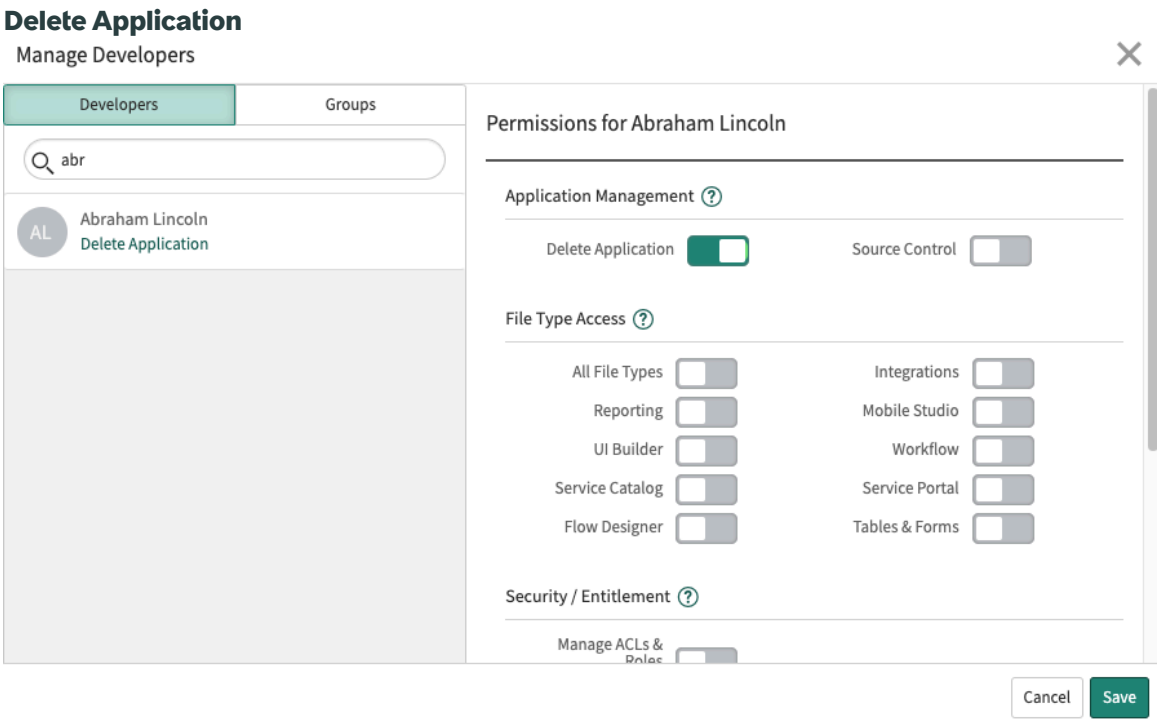
Ability to assign the ability to delete an application to a delegated developer.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. Open an application for which you want to assign delegated developer permissions.
3. As a System Admin, select **File** > **Manager Developers**.
4. Choose a developer to apply delegated developer permissions.
5. Toggle on **Delete Application**, and then select **Save**.



6. The delegated developer can now delete the application.

Using Delegated Development

Use Delegated Development to develop or deploy applications on the ServiceNow AI Platform.

Overview of Delegated Development

- Delegate development and deployment permissions to personnel

  A system administrator can assign a non-administrator user or group as a developer or deployment resource for a specific application.

- Developer and deployment permissions

  Using Manage Developers, administrators can assign one or more developer and deployment permissions to a group or user for a specific application.

- Remove a developer

  Removing a user as a developer prevents the user from developing, changing, or deploying the application in the current instance.

Delegate development and deployment permissions to personnel

A system administrator can assign a non-administrator user or group as a developer or deployment resource for a specific application. You can set permissions that designate what specific actions the assigned user can perform in the current instance, or make the user a Delegated Admin to grant access to all permissions at once.

Before you begin

- Role required: admin or application administrator

  If Application administration is enabled, only an application administrator of the target application can delegate developers to an application. If application administration is not enabled, an admin user can delegate developers.

- Records required:
  - Application
  - User
  - Group

Procedure

1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. Select the name of the application to which you want to add developers.
   The system opens the application record.
3. Select **Manage Developers**.

   Note:
   If the App Collaboration Component is installed, the link appears as **Manage Collaborators**.

   The system displays the Developer Permissions window.

   Note:
   Developer permissions are available only for scoped apps, not global apps.

4. Select the type of developer (or user without system admin roles) you want to assign.
5. In **Developer Name** or **Group Name**, enter the name of the user or group you want to grant developer or deployment permissions.
6. Select specific developer and deployment permissions for the application, or select **Delegated Admin** to quickly grant access to all permissions for the application.
   For details, see Developer and deployment permissions.
7. Select **Save**.
   The system assigns the designated developer and deployment permissions for the application.

Developer and deployment permissions

Using Manage Developers, administrators can assign one or more developer and deployment permissions to a group or user for a specific application. These permissions designate the specific actions the assigned user can perform for the application.

For example, you might grant permissions that enable a user to upgrade the application, publish to the application repository and ServiceNow Store, but prevent publishing to an update set.

Developer permissions

Note:
If the App Collaboration Component is installed, the link appears as **Manage Collaborators**.

Permission	Description
Delete Application	Grants the assigned developer within a scoped app rights to delete the application.
Source Control	Grants the assigned developer full access to source control.
All File Types	Grants the assigned developer access to all application file types, including some not granted by the other options. This permission is equivalent to granting the user the admin role but with some limitations. Specifically, it provides access to all file types that are configured in your application per the Manage Developers task in the Application Creator. For an example of such file types, see the permissions example in Delegated development and deployment.
Playbooks	Grants the assigned developer access to the Playbooks design environment to create processes. Editing activity subflows or actions requires the **Flow Designer** permission.
Integrations	Grants the assigned developer access to web service APIs, REST APIs, data sources, and Integration Hub - Import.
Reporting	Grants the assigned developer access to reports and scheduled reports.
Notifications	Grants the assigned developer access to work with automatic email notifications.
Decision Tables	Grants the assigned developer access to Decision Tables to create decision logic based on multiple if-then rules.
Mobile Builders	Grants the assigned developer access to mobile builders, such as Mobile App Builder.
UI Builder	Grants the assigned developer access to UI Builder to create pages for experiences.
Workflow	Grants the assigned developer access to the Workflow Editor and Activity Creator.
Service Catalog	Grants the assigned developer access to catalog-related file types such as catalog items, record producers, and variables.
Service Portal	Grants the assigned developer access to Service Portal editors and tools.
Workflow Studio	Grants the assigned developer access to the Workflow Studio design environment to create flows and actions. Script action steps require the **Allow Scripting** permission.
Tables & Forms	Grants the assigned developer access to model and layout related file types such as table columns, form layout, and list layout.
Manage ACLs & Roles	Grants the assigned developer access to security-related file types such as access controls and user roles.
Allow Scripting	Grants the assigned developer write access to script fields such as those in business rules, client scripts, and Workflow Studio script action steps.
Manage Collaborators	Grants the assigned developer the ability to invite and manage users and groups. This permission allows the delegated developer to further invite and manage other developers to the application.

Permission	Description
Invite Collaborators	Grants the assigned developer the ability to invite users and groups. This permission allows the delegated developer to further invite other developers to the application.
Delegated Admin	Grants access to all Delegated Development permissions so that permissions do not need to be granted individually.

Deployment permissions

The update set deployment permissions are hidden by default and require a system administrator to enable them with system properties. See Display or hide update set deployment permissions for more information.

The Submit for Deployment, Manage Collaborators, and Invite Collaborators delegated development permission sets are only available with the Developer Collaborator feature. They will not be shown in Manage Developers.

Permission	Description
Upgrade App	Grants a user with an assigned delegated developer role permission to upgrade the associated application after it has been installed in the current instance.
Publish To Update Set	Grants a user with an assigned delegated developer role permission to publish the associated application to an update set in the current instance. **Note:** Users with this permission cannot also have the **Manage Update Set** permission.
Publish To App Store	Grants a user with an assigned delegated developer role permission to publish associated application to the ServiceNow Store in the current instance. **Note:** The **Upgrade App**, **Publish To App Repo**, and **Publish To App Store** permissions display by default. The **Publish To Update Set** permission only displays if manually enabled by a system administrator. For more details, see Display or hide update set deployment permissions.
Manage Update Set	Grants a user with an assigned delegated developer role permission to manage local and retrieved update sets. This permission allows users to create, update, and delete local update sets as well as preview, resolve conflicts, and commit retrieved update sets. **Note:** Users with this permission cannot also have the **Publish To Update Set** permission.
Publish To App Repo	Grants a user with an assigned delegated developer role permission to publish the associated application to the application repository in the current instance.
Submit for Deployment	Grants a user with assigned delegated developer role permission to submit the associated application for review and deployment.

Remove a developer

Removing a user as a developer prevents the user from developing, changing, or deploying the application in the current instance.

Before you begin

Role required: admin or application administrator

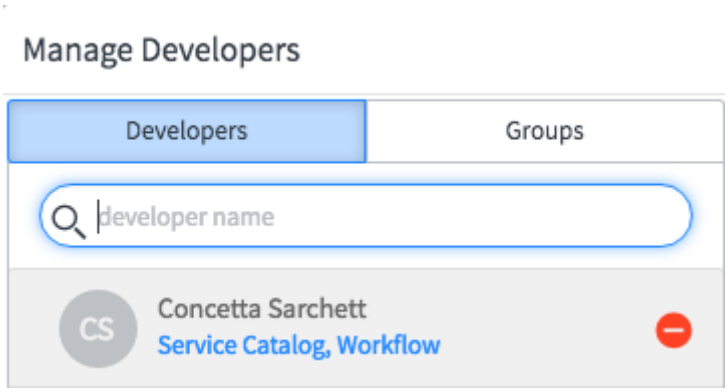
If Application administration is enabled, only an application administrator of the target application can delegate developers to an application. If application administration is not enabled, an admin user can delegate developers.

Procedure

1. Navigate to **All** > **System Applications** > **Applications**.
2. Select the application name of the application from which you are removing developers.
   The system opens the application record.
3. Select **Manage Developers**.

   Note:
   If the App Collaboration Component is installed, the link appears as **Manage Collaborators**.

   The system displays the Developer Permissions window.
4. Point to the developer you want to remove.
   The system displays a minus icon next to the developer name.



5. Select the minus icon next to the developer name.
   The system removes the developer and any associated application roles.

Team Development

Team Development supports parallel development on multiple, non-production ServiceNow instances.

Team Development provides the following features:

- Branching operations, including pushing and pulling record versions between instances.
- The ability to compare a development instance to other development instances.
- A central dashboard for all Team Development activities.

Team Development overview

Team Development allows developers to work on separate development instances while sharing code and resolving collisions throughout the development process.

After setting up the instance hierarchy, you can develop changes on your local development instance. Use the team dashboard to manage Team Development activities, such as:

- Tracking local changes and determining which changes to promote to the parent development instance.
- Pulling changes from the parent instance and resolving any collisions with local changes.
- Comparing your instance with other development instances and resolving any collisions with other development projects.
- Pushing changes when a feature is tested and ready to promote to the parent development instance.

Developers with admin access to their development instance and the parent instance can use team development. For alternative access settings, see Granting access rights to developers.

Related topics

Set up an instance hierarchy

When to use Team Development

Team Development allows multiple developers to work on applications.

Deployment option	Good for	Future considerations
Update Sets	Storing changes to a base system or installed application. Storing and applying a particular version of an application. Producing a file for export.	You can manually create update sets to store a particular application version. Use update sets to deploy patches or changes to installed applications. **Note:** Don't use update sets to install applications. Instead, use the application repository or the ServiceNow Store to install applications.
Application Repository	Installing and updating applications on all company instances. Automatically managing application update sets. Restricting access to applications to the same company. Deploying completed applications to end users.	Consider uploading an application to the ServiceNow Store to share it with other users. Allows installation of and update to the latest application version only. Use update sets to store prior application versions. **Note:** If used with team development, publish applications only from a parent instance.

Local changes

The Local Changes table tracks which customized records have current versions that exist on the development instance but not on the parent instance.

Use local changes to collect changes in preparation for a push.

You queue local changes that are ready to push. Each development instance maintains a single queue, regardless of who develops or queues the changes. You ignore local changes that you do not want to push. For example, you may want to ignore changes to the color scheme that visually distinguish a development instance from the production instance. You can remove a change from the queue or stop ignoring a change.

Changing the parent instance or reconciling recreates the list of local changes that have not been queued or ignored. If you had previously queued or ignored a local change, that designation is maintained.

Local change lists

On the team dashboard, the *Local Changes* list shows the local changes that have not been queued for the next push or ignored for all pushes.

The *Ready to Push* list shows the changes that are queued, and the *Ignored* list shows the changes that are ignored. Use any of these methods to navigate a list of local changes.

local changes list

Action	Option
Click the reference icon beside the row	To open the local change record itself.
Click the link in the first column	To open the customized record.
Right-click the row and select **Show Changes Since Last Pull**. An error message appears if a previous version does not exist (for example, in the case of a newly created record). If a previous version is available, you can revert to that version from the comparison window	To view a comparison between the current local version and the version most recently pulled from or pushed to the parent.
Right-click the row and select **Show Application File**	To open the application file for the customized record.
Right-click the row and select **Show Version**	To open the current version record.

Pull exceptions

Pulling ignores versions when certain conditions occur.

Pull exceptions table

Issue	Description
Matched an exclusion policy	An exclusion policy prevents pulling changes for records matching the policy conditions. The pull identifies the changes but does not include versions for these records.

Pull exceptions table (continued)

Issue	Description
Private properties	A private property is excluded from all Update Sets and pulls.
Collisions	A collision is detected when the pulled version and the current local version both include modifications to the same record. You must resolve all collisions before you can pull.
Previously resolved collisions	When you resolved a collision by accepting either the pulled version or local version of a record, the pull remembers your decision and accepts the version that you indicated as a "previously resolved collision".
Skipped	Pulls skip versions where there is a problem with the version record such as a corrupt or missing version.

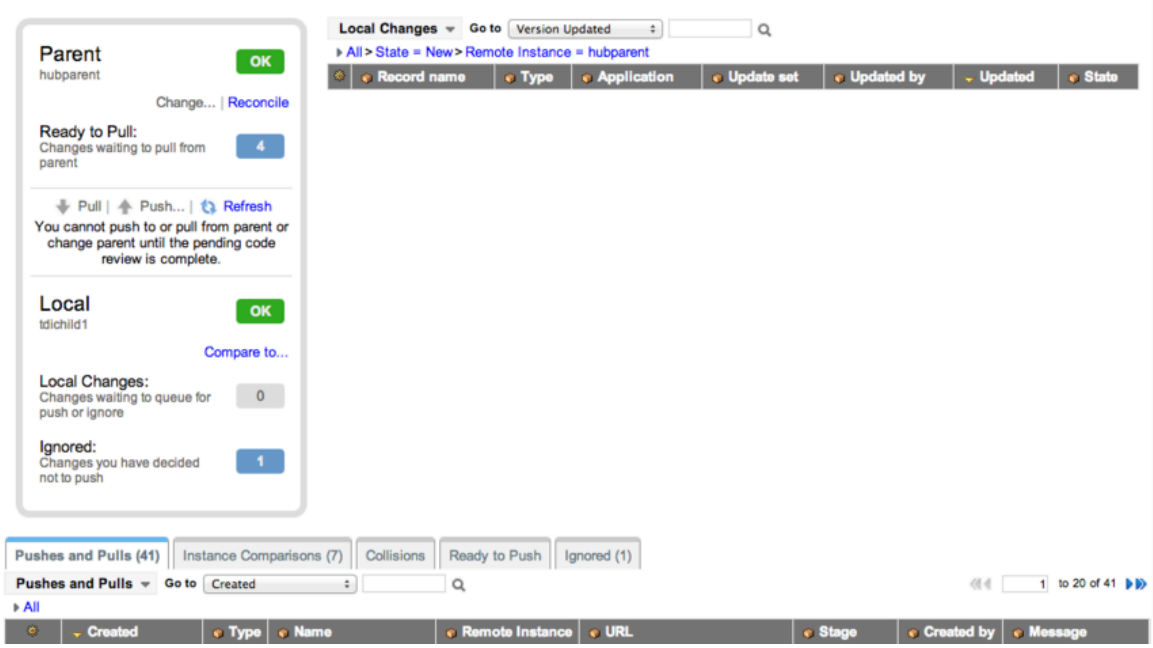
Team dashboard

The team dashboard provides a central place to manage all Team Development activities on your development instance.

You can track local changes, pull and push changes between the local and parent instances, compare the local instance to other development instances, and resolve any collisions. You can also reconcile with the current parent instance or change the parent instance.

To access the dashboard, navigate to **Team Development** > **Team Dashboard**.

Team dashboard



The control panel in the top left provides status indicators and Team Development actions.

- Parent: indicates the status of the connection to the parent instance. If a problem or warning is detected, point to the indicator to view the error messages, or click the indicator to open the remote instance record.
- **Change**: changes the parent instance. See Changing the Parent Instance.
- **Reconcile**: compares the development instance to the parent instance. See Reconciling.

- Ready to Pull: indicates the number of changes on the parent that have not been pulled to the local instance.
- **Pull**: initiates a pull. See Pulling Versions.
- **Push**: opens a page that allows you to review the changes before a push. See Pushing Versions.
- **Refresh**: updates the status indicators on the control panel. The dashboard updates only when you reload or refresh the page.
- Local: indicates the status of the most recent comparison with another instance. If collisions are detected, click the indicator to open the list and resolve the collisions. See Resolve a collision in Team Development.
- Collisions: appears only if any local changes collide with versions pulled from the parent and indicates the number of collisions. Click the indicator to open the list and resolve the collisions. See Resolve a collision in Team Development.
- **Compare to**: allows you to select another development instance to compare with the local instance. See Comparing to Peer Instances.
- Ready to Push: indicates the number of local changes that are queued for the next push. See Queuing and Ignoring Local Changes.
- Local changes: indicates the number of local changes that have not been queued or ignored. Click the indicator to open a list of these changes.
- Ignored: appears only if any local changes are ignored and indicates the number of ignored changes. Click the indicator to open a list of these changes.

The team dashboard includes lists for tracking local changes and viewing the history of Team Development activities.

- Local changes: lists the local changes that have not been queued or ignored.
- Pushes and Pulls: provides a history of pushes and pulls. Expand a row to see the customized records for which versions were transferred as part of the push or pull.
- Instance Comparisons: provides a history of comparisons with other development instances.
- Collisions: lists the collisions that must be resolved before the next pull or push. You can right-click a row and select **Resolve Collision**. See Resolving Collisions.
- Ready to Push: lists the local changes that have been queued for the next push.
- Ignored: lists the local changes that are ignored for all pushes.

Related topics

Team Development overview

Pull a version

Pull exceptions

Resolve a collision in Team Development

Resolve multiple collisions

Local changes

Local change lists

Queue a local change for a push

Ignore a local change

Back out a local change

Approve or reject a push

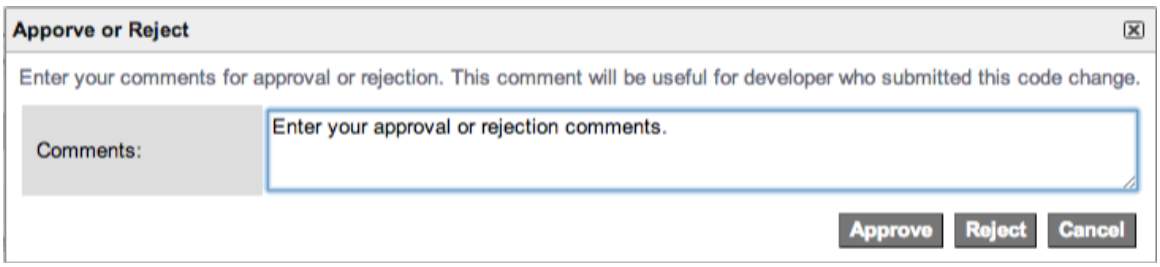
Code reviewers must approve or reject a push from the Team Development application.

About this task

Although reviewers can see the individual versions within a push, they must approve or reject the push as a whole.

Procedure

1. Log in to the parent instance that requires code review.
2. Navigate to **Team Development** > **Code Review Requests**.
3. Select a change in the *Awaiting Code Review* stage.
4. Review the changes in the *Push or Pull Versions* related list.
5. Click **Approve or Reject**.
6. **Optional:** Enter review comments in *Comments*.
   These comments are visible to anyone who can see the Pushes and Pulls history.
7. Click either **Approve** or **Reject**, as appropriate.



Note:
The *URL* and *Remote Instance* fields list the address and name of the instance where the change originated.

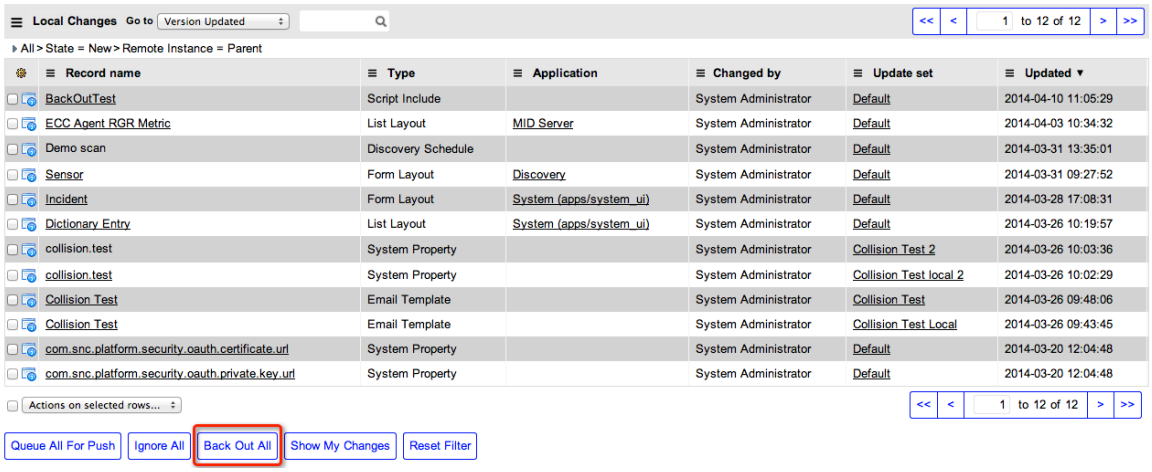
Back out a local change

Back out all local changes and restore the last version reconciled with the parent instance.

Procedure

1. Define a parent instance.
2. Pull changes from the parent instance.
3. Navigate to **Team Development** > **Team Dashboard**.
4. Filter the *Local Changes* list to show only the changes that you want to back out.
5. Do one of the following actions:

- Click **Back Out All**.
- Right-click the local change you want to back out, and then click **Back Out**.



Cancel a code review request

Developers can cancel any push they submitted that is in the *Awaiting Code Review* stage.

About this task

Canceling a request sets the push to the **Code Review Request Cancelled** stage on the submitting instance. The submitting instance retains a version history of the push but the parent instance does not.

Procedure

1. Log in to the instance that pushed the changes.
2. Navigate to **Team Development** > **Pushes and Pulls**.
3. Filter for the push you want to cancel.

> **Note:**
> You cannot cancel a push that has been approved or rejected.

4. Select the Push or Pull record.
5. Click **Cancel Code Review**.

Change the parent instance

If it becomes necessary to modify the instance hierarchy, you can change the parent for a development instance.

About this task

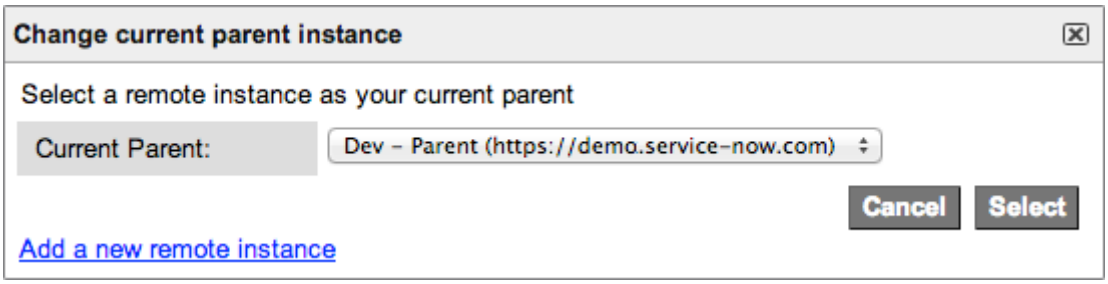
Changing the parent initiates a complete comparison between the development instance and the new parent instance. To optimize comparison speed and reduce the number of collisions and local changes that need review afterwards, ensure that the new parent instance was cloned recently from an appropriate instance (for example, the production instance). Before you change the parent instance, ensure that the change does not conflict with your change management process or other development efforts.

To change the parent for a development instance:

Procedure

1. On the development instance, navigate to **Team Development** > **Team Dashboard**.
2. In the control panel, click **Change**.
3. Select the remote instance you want to use as the parent and click **Select**.

   Alternatively, click the link to define a new remote instance. Then, repeat steps 1–3 and select the remote instance you defined.



   The system initiates a reconcile, which compares the local instance to the parent, and then generates the list of local changes and calculates the number of changes that are ready to pull from the parent.

4. On the completion page, click **Team Dashboard**.
5. Pull versions from the parent instance and resolve any collisions.
6. Review the local changes list and queue or ignore changes, as appropriate.

Check the review status of a pushed change

If the parent instance requires pushed changes to undergo code review, changes are placed in the *Awaiting Code Review* stage.

About this task

If you configure the parent instance to send notifications, it sends the submitting developer a notification when the pushed changes are approved or rejected. Developers can also manually check the status of their pushed changes from the *Pushes and Pulls* module on the submitting instance.

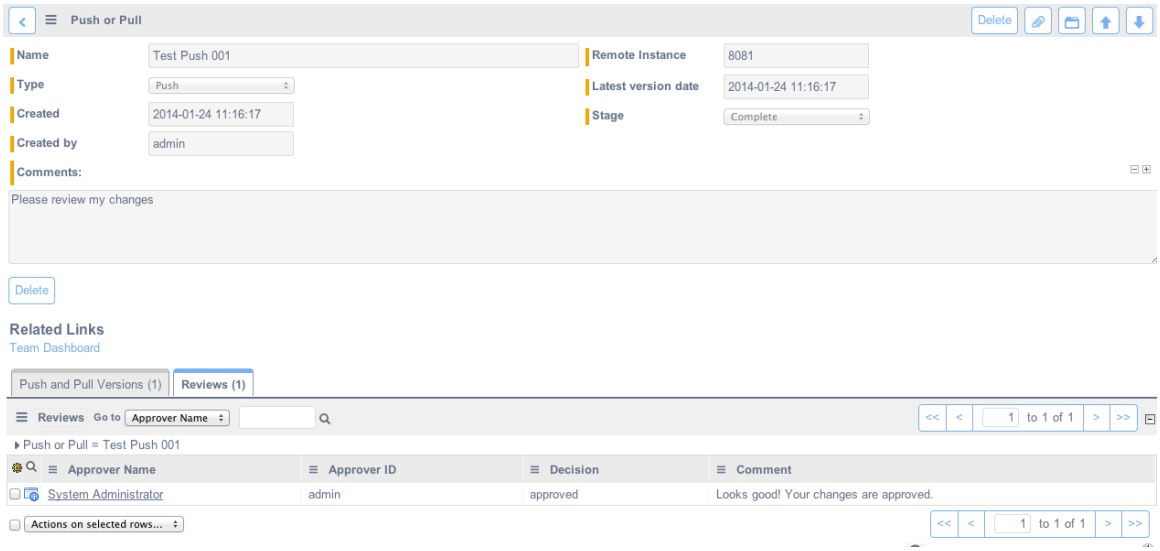
Procedure

1. Log in to the instance that submitted code for review.
2. Navigate to **Team Development** > **Pushes and Pulls**.
3. Filter for the push you want to review.

   - Pushes in the *Complete* stage are approved and applied to the parent instance.
   - Pushes in the *Collided* stage are rejected because of a collision.
   - Pushes in the *Awaiting Code Review* stage are awaiting review.
   - Pushes in the *Code Changes Rejected* stage are rejected by a reviewer.
   - Pushes in the *Code Review Request Canceled* stage are canceled by the submitting developer.

4. Click the **Reviews** related list to see the following information.

- Who submitted a review decision.
- What the decision was: either approved or rejected
- What comments if any the reviewer provided.



Compare a pushed version to a local version

Code reviewers can compare the pushed versions to the local versions to see the potential effect of incoming changes.

Procedure

1. Log in to the instance requiring code review.
2. Navigate to **Team Development** > **Code Review Requests**.
3. Select a change in the *Awaiting Code Review* stage.
4. Review the changes in the *Push or Pull Versions* related list.
5. Right-click a row in the list and click **Compare to Current**.
   A comparison of the differences between the pushed and local versions appears.

Related topics

Merge tool

Compare to the current version

Compare two versions of an article

Resolve conflicts for an individual record

Resolve a collision in Team Development

Revert a change

View customizations and compare with current version

Compare to peer instances

You can compare the local instance to any other remote instance and commit any current versions from the remote instance on your development instance.

About this task

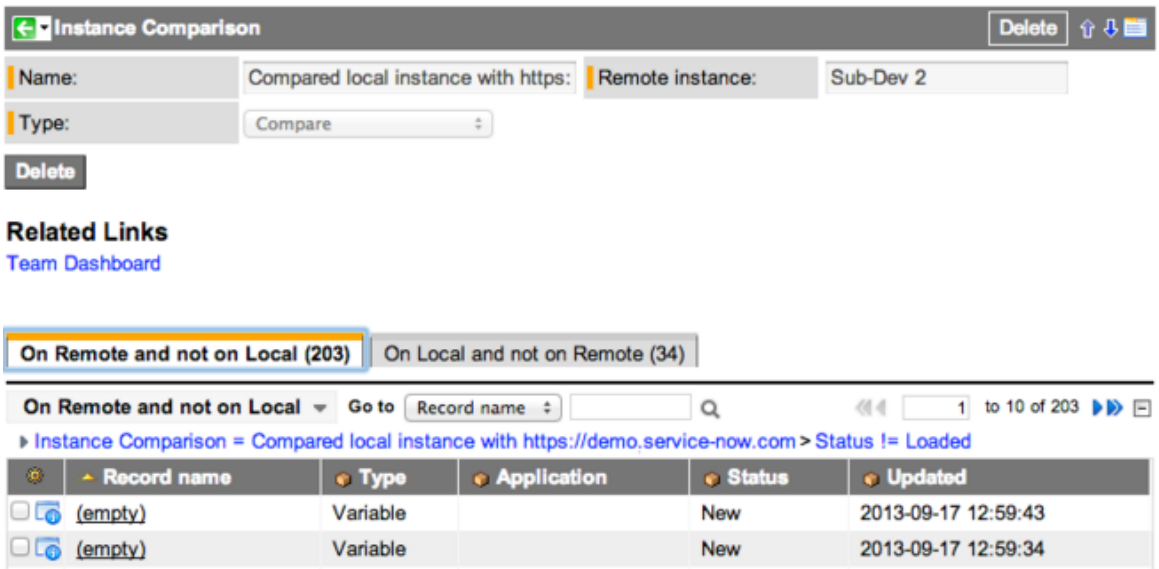
Comparing allows you to share code between instances without pushing to a common parent.

Comparing instances does not automatically commit any versions on the local instance. It initiates a full comparison of all changes on the remote instance and all changes on the local instance, and then reports which customized records have different current versions. You can selectively commit a version from the remote instance or compare it with the version on your local instance. You can delete the instance comparison record when you finish evaluating the differences.

To compare the local instance to a peer instance:

Procedure

1. Ensure that the peer instance is defined as a remote instance.
2. Navigate to **Team Development** > **Team Dashboard**.
3. In the control panel, click **Compare to**.
4. Select the peer instance you want to compare to the local instance and click **Compare**.
5. On the completion page, click **Show Results**.
   The instance comparison record opens.



6. Review the *On Remote and not Local* related list, which shows the customized records where the current version on the peer instance is not on the local instance.

   For each customized record, you can:

   - Compare the current remote version to the current local version by right-clicking a row and selecting **Compare to Current**.
   - Load the current remote version as the current local version by right-clicking a row and selecting **Load This Change**.

Ignore a local change

Ignoring a local change prevents updates to a record from generating new versions in the Local Changes list.

About this task

An ignored local change always points to the current version for the record. You cannot push ignored records to another instance.

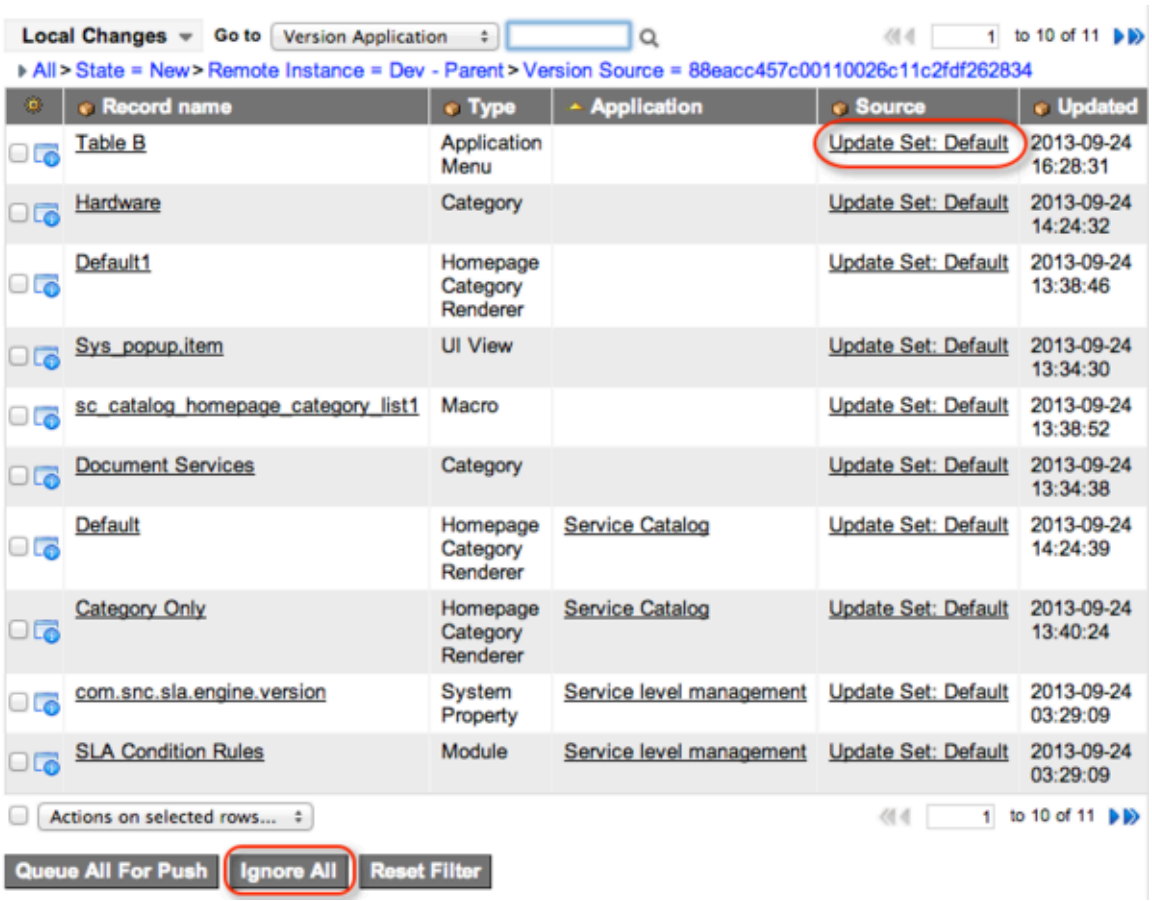
Local change action list

Action	Result
Ignore a record that has a version queued for push	The queued change is deleted
Ignore a record that has a version queued for code review	The queued change is deleted
Pull changes for an ignored record	Collision
Resolve a collision by taking the parent version	There is no longer a local change to ignore
Resolve a collision by keeping the local version	The ignored change remains on the local instance

Procedure

1. Navigate to **All** > **Team Development** > **Team Dashboard**.
2. Filter the *Local Changes* list to show only the changes that you want to ignore.

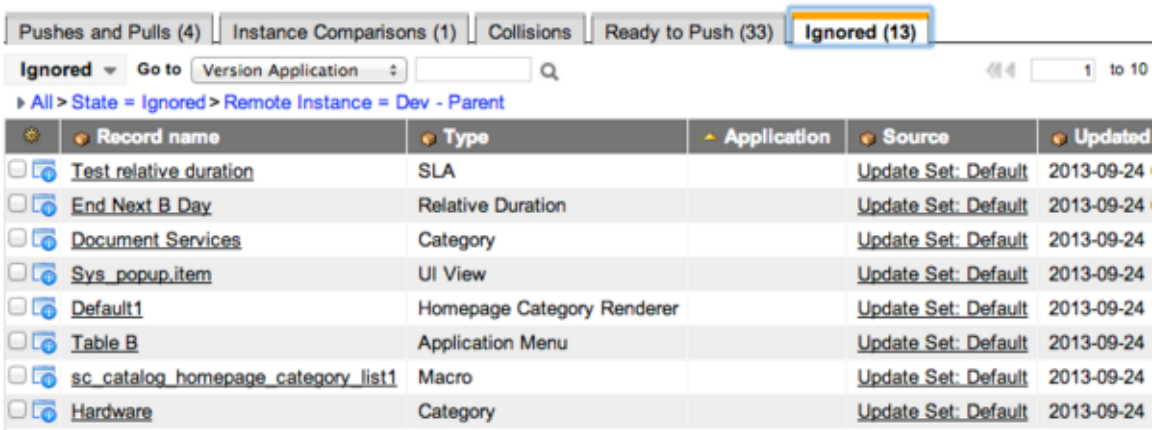
   For example, filter the list to show all changes in the *Default* Update Set.



3. Click **Ignore All**.

4. Review the *Ignored* list to ensure that the correct changes are ignored.
   This step is a recommended best practice.

Option	Description
To stop ignoring changes	Select the check boxes beside the rows and select **Do Not Ignore** from the *Actions* choice list.
To stop ignoring changes and add them to the queue instead	Select the check boxes beside the rows and select **Queue for Push** from the *Actions* choice list.



Pull a version

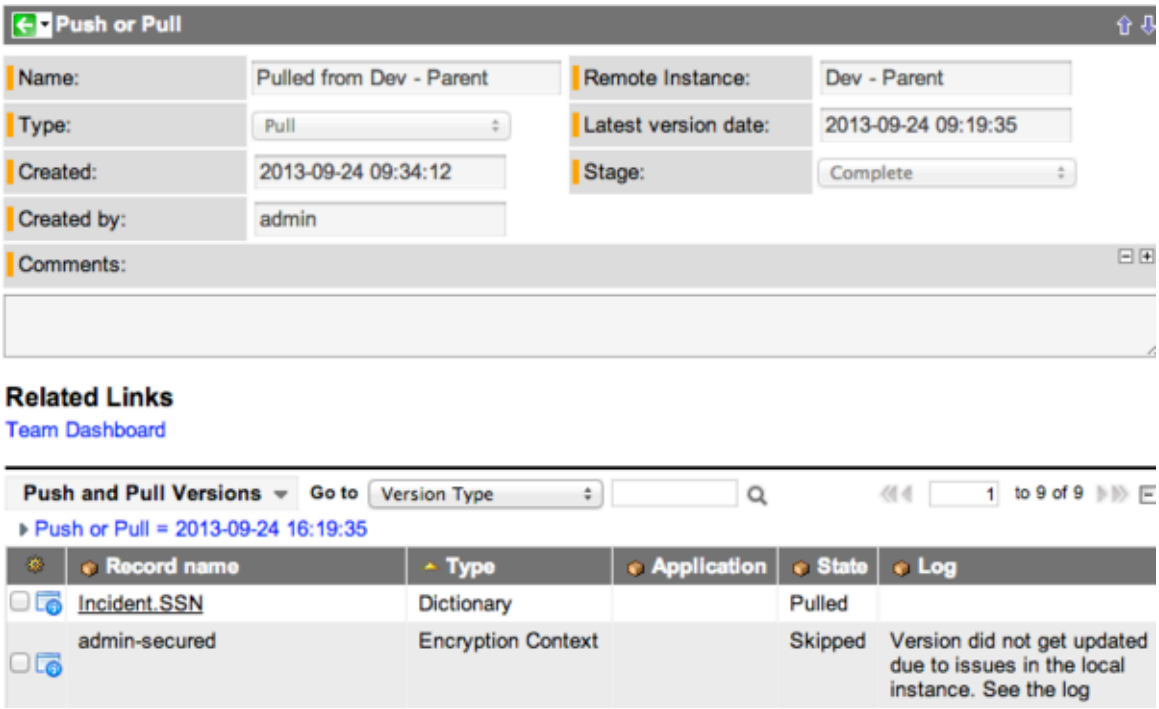
Pulling retrieves versions of customized records from the parent instance and adds them on the development instance. Pulling does not retrieve any versions for changes made by system upgrades, but it retrieves all versions for changes made by users, not just the current version.

About this task

Pulling retrieves all versions for changes made by users that have not already be pulled onto the development instance, and you cannot choose which versions to pull. The first time you pull from a parent instance, the pull retrieves all versions for changes made by users. Subsequent pulls retrieve the new versions since your last pull. Each pull is recorded in the Push or Pull `[sys_sync_history]` table on the development instance. Historical versions are saved with a state of *History*.

Procedure

1. Navigate to **All** > **Team Development** > **Team Dashboard**.
2. In the control panel, click **Pull**.
3. On the completion page, click **Show Results**.
   The *Push and Pull Versions* related list for the Push or Pull form shows the customized records for which versions were retrieved and indicates if any pull exceptions exist.



4. Resolve any collisions.

Push a version

Pushing promotes changes from the development instance to the parent instance and commits the current version of a customized record on the development instance as the current version on the parent instance.

About this task

Pushing adds only the current development version to the parent, not all the development versions.

Note:
Updates to records from different applications cannot be pushed/pulled in the same push/pull. To resolve the error in the case that updates to other applications are mixed in: De-queue the updates to other applications. Push for one application. Re-queue the updates to one application. Push and then repeat as needed.

Pushing creates a local Update Set on the parent that is marked as complete. Pushed changes are also tracked as local changes on the parent. Therefore, you can promote changes through your development and test hierarchy by transferring the Update Set or by pushing the local changes. Each push is recorded in the Push or Pull table on the development instance.

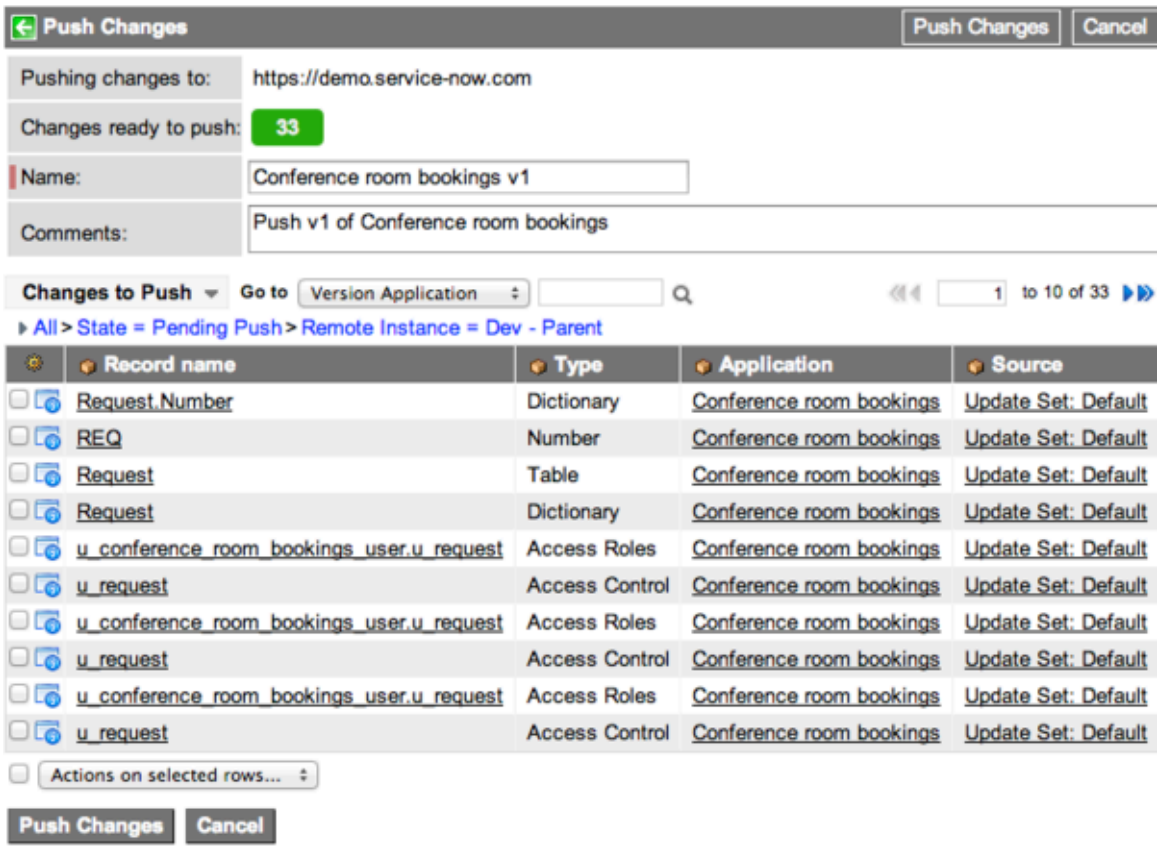
Procedure

1. Navigate to **All** > **Team Development** > **Team Dashboard**.
2. Queue the local changes that are ready to push.
3. Pull versions from the parent instance and resolve any collisions.

   You cannot push changes to the parent instance if collisions are detected.
4. In the control panel, click **Push**.
   The Push Changes page opens.

5. Provide a *Name* for the changes.
6. Review the list of changes to ensure that the correct changes are included.

Option	Description
To remove changes that you do not want to push	Select the check boxes beside the rows and select **Do Not Push** from the *Actions* choice list
To add changes	Click **Cancel** and repeat the procedure from step 2



7. **Optional:** Edit the name.
   The name identifies the push record on the development instance and the local Update Set record on the parent instance.
8. **Optional:** Enter comments.
   The comments are added to the push record on the development instance and the local Update Set record on the parent instance.
9. Click **Push Changes**.

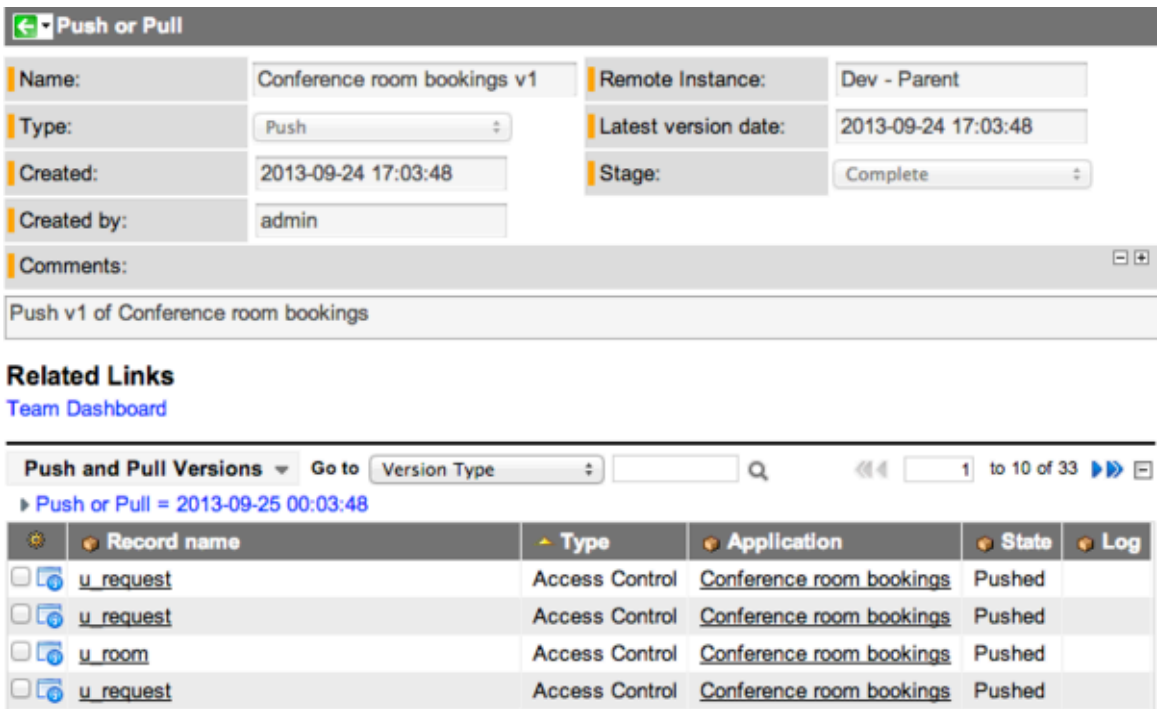
   The system initiates a pull to ensure that there are no collisions before the push proceeds.

   - If collisions are detected, the push is automatically canceled and you must repeat the procedure from step 3.
   - If no collisions are detected, the changes are staged on the parent instance. On the parent, each version is validated and then committed in the correct order to maintain dependencies between records. For example, a new table is committed before a field on that table to ensure the field is properly created.

Note:
You cannot push if there is a version conflict between instances or the pushing instance has changes in the *Awaiting Code Review* stage.

10. On the completion page, click **Show Results**.
11. Review the push record for any errors or skipped changes.
    - Changes with a state of *Pushed* were committed on the parent instance.
    - Changes with a state of *Skipped* were not committed on the parent instance and remain queued as local changes on the development instance.
12. For each skipped change, review the log message to determine why the change was skipped.

    Develop any changes that are necessary to commit the desired version on the parent instance, and then push them. Some examples of why a change may be skipped include:
    - A table does not exist on the parent because it was created when you activated a plugin on the development instance. Ensure the plugin is activated on the parent and push the change again.
    - An error occurred during the push. Try to push again.
    - The current version is invalid. Revert to a previous version and make the change again to ensure the version is valid
    - An error occurred on the parent during the push. The *Log* field on the push record contains the exception message. Review the system logs on the parent instance and troubleshoot any problems with the instance.



Back out a push

Application developers can back out a push to remove unwanted changes.

Procedure

1. Navigate to **Team Development > Pushes and Pulls**.
2. Select the push to back out.
3. Click **Back Out**.
4. Click **OK** when the confirmation message appears.

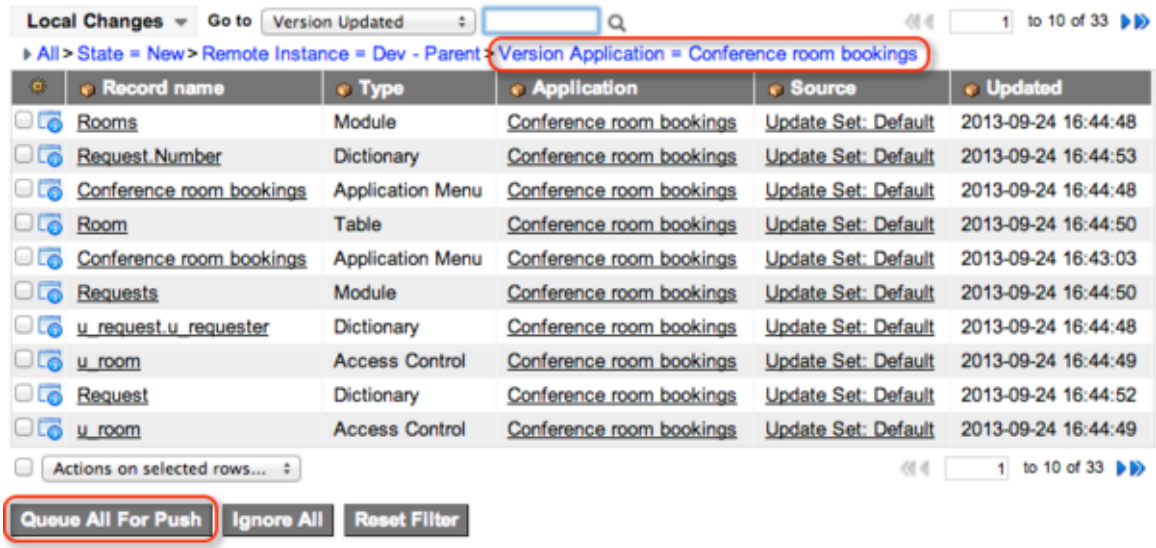
Queue a local change for a push

Application developers can queue a local change for a push to ensure the changes are available to other developers.

Procedure

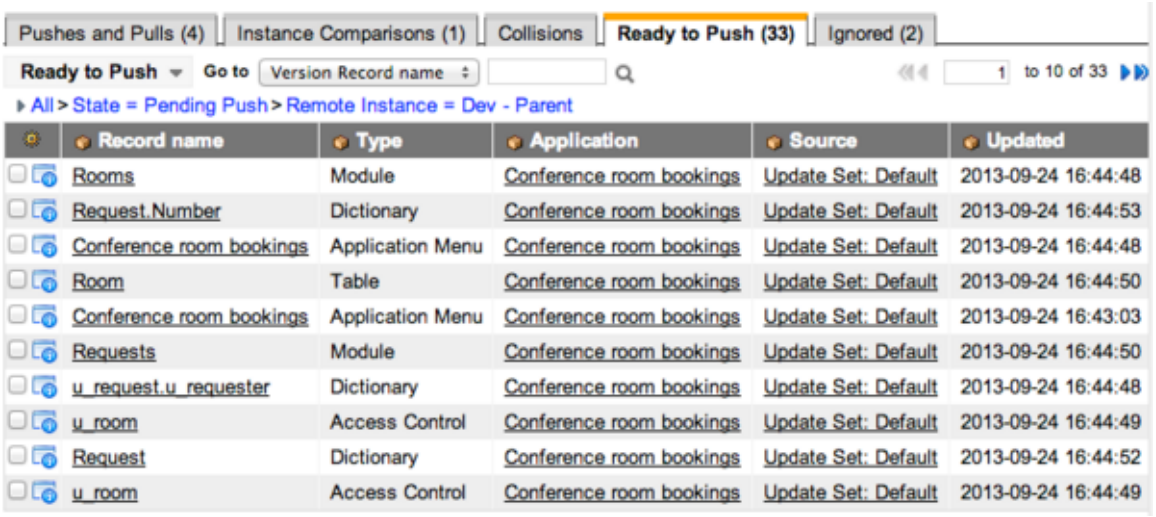
1. Navigate to **All** > **Team Development** > **Team Dashboard**.
2. Filter the *Local Changes* list to show only the changes that are ready to push.

   For example, filter the list to show only the changes associated with a particular application.



3. Click **Queue All For Push**.
4. Review the *Ready to Push* list to ensure that the correct changes are in the queue.
   This step is a recommended best practice.

Option	Description
To remove changes from the queue	Select the check boxes beside the rows and select **Do Not Push** from the *Actions* choice list.
To remove changes from the queue and choose to ignore them instead	Select the check boxes beside the rows and select **Ignore This Change** from the *Actions* choice list.



Record name	Type	Application	Source	Updated
Rooms	Module	Conference room bookings	Update Set: Default	2013-09-24 16:44:48
Request.Number	Dictionary	Conference room bookings	Update Set: Default	2013-09-24 16:44:53
Conference room bookings	Application Menu	Conference room bookings	Update Set: Default	2013-09-24 16:44:48
Room	Table	Conference room bookings	Update Set: Default	2013-09-24 16:44:50
Conference room bookings	Application Menu	Conference room bookings	Update Set: Default	2013-09-24 16:43:03
Requests	Module	Conference room bookings	Update Set: Default	2013-09-24 16:44:50
u_request.u_requester	Dictionary	Conference room bookings	Update Set: Default	2013-09-24 16:44:48
u_room	Access Control	Conference room bookings	Update Set: Default	2013-09-24 16:44:49
Request	Dictionary	Conference room bookings	Update Set: Default	2013-09-24 16:44:52
u_room	Access Control	Conference room bookings	Update Set: Default	2013-09-24 16:44:49

Note:
For the *Local Changes* list, click **Reset Filter** to remove any filter conditions you added and see all the local changes that have not been queued or ignored.

Reconcile changes

Reconciling first compares the local instance to the parent, and then generates the list of local changes and calculates the number of changes that are ready to pull from the parent.

About this task

A reconcile occurs automatically whenever you select a parent instance. You may need to manually reconcile after an external disruptive event on the parent instance, such as a clone or failover.

Note:
This process may take a while to complete depending on the size and age of the instance.

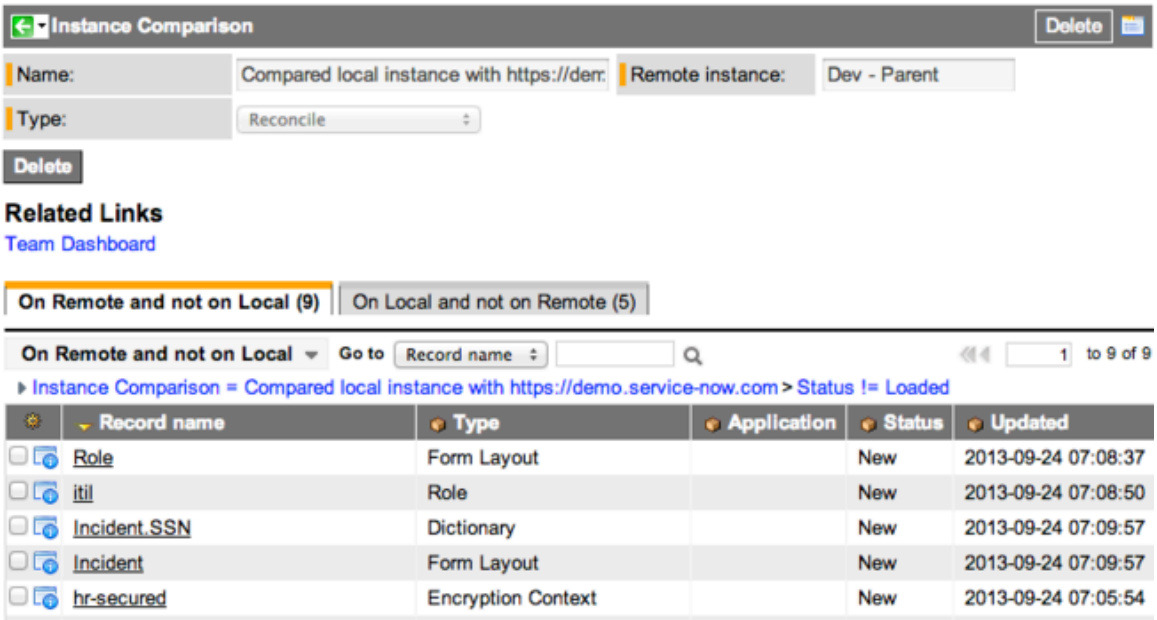
Procedure

1. Navigate to **All** > **Team Development** > **Team Dashboard**.
2. In the control panel, click **Reconcile**.
3. In the confirmation dialog box, click **OK**.

   The list of local changes that have not been queued or ignored is recreated. If you had previously queued or ignored a local change, that designation is maintained.
4. **Optional:** On the completion page, click **Show Results**.

   Review the instance comparison record.

   - The *On Remote and not Local* related list shows the versions that are ready to pull from the parent.
   - The *On Local and not on Remote* related list shows the local versions that are ready to queue or ignore.



5. Click **Team Dashboard**.
6. Pull versions from the parent instance and then resolve any collisions.
7. Review the local changes list and queue or ignore changes, as appropriate.

Resolve a collision in Team Development

A collision is detected when the pulled version and the current local version are modifications of a different version, indicating that someone else has modified the same record that you have modified. The team dashboard displays the number of collisions between the local and the parent instance.

About this task

To ensure that your changes do not conflict with other development efforts, you should resolve collisions as soon as they are identified. You must resolve all collisions before you can pull or push.

Procedure

1. Navigate to **All** > **Team Development** > **Team Dashboard**.
2. In the control panel, click **Collisions** or click the count of collisions.
   A list of collisions opens.
3. Right-click a row and select **Resolve Collision**.
   Alternatively, open the record and click the **Resolve Collision** related link.
   The Resolve Collision page displays a comparison between the version that was pulled from the parent and your local record. The page highlights the differences.
4. Review the differences and perform an action.
   The system performs that action and also clears the collision for future push/pulls.
5. Repeat the process for every remaining collision.

Result

The system saves the merged changes and resolves the collision.

Related topics

- Compare to the current version
- Compare a pushed version to a local version
- Compare two versions of an article
- Resolve conflicts for an individual record
- Revert a change
- View customizations and compare with current version

Limitations on updating records

There are some types of records that you cannot merge while resolving differences on the Compare to Current and Resolve Collision pages.

Record types that allow a choice only between reverting or accepting the pulled or current record

The following record types do not allow you to merge individual values. Instead, differences involving the following record types display a read-only comparison and allow a choice between updating and reverting:

- sys_choice [Choice]
- sys_choice_set [Choice Set]
- sys_ui_form [Form]
- sys_ui_list [List]
- sys_ui_related_list [Related List]
- sys_ui_section [Form Section]
- wf_workflow [Workflow]
- wf_workflow_version [Workflow Version]

Team Development, Resolve Collision page: **Use Pulled Version** and **Use Local Version** options.

Upgrade History, Compare to Current page: Comparing non-current update versions to current update version. Allows only **Revert to Base System** option.

Field types that do not support merging

The following field types do not support individual merging between versions or updates:

- auto_increment [Auto Increment]
- auto_number [Auto Number]
- breakdown_element [Breakdown Element]
- catalog_preview [Catalog Preview]
- collection [Collection]
- color_display [Color Display]
- composite_field [Composite Field]
- compressed [Compressed]
- counter [Counter]
- currency [Counter]

- data_array [Data Array]
- data_object [Data Object]
- data_structure [Data Structure]
- date [Other Date]
- datetime [Basic Date/Time]
- days_of_week [Days of Week]
- document_id [Document ID]
- due_date [Due Date]
- Email [Email]
- external_names [External Names]
- field_list [Field List]
- float [Floating Point Number]
- glide_action_list [UI Action List]
- glide_precise_time [Precise Time]
- glide_var [Glide Var]
- image [Basic Image]
- index_name [Index Name]
- int [Integer String]
- integer_time [Integer Time]
- ip_address [IP Address]
- journal [Journal]
- journal_input [Journal Input]
- journal_list [Journal List]
- long [Long Integer String]
- mask_code [Mask Code]
- metric_absolute [Metric Absolute]
- metric_counter [Metric Counter]
- metric_derive [Metric Derive]
- metric_gauge [Metric Gauge]
- mid_config [MID Server Configuration]
- month_of_year [Month of Year]
- multi_small [Multiple Line Small Text Area]
- name_values [Name/Values]
- nl_task_int1 [NL Task Integer 1]
- order_index [Order Index]
- password [Password (1 Way Encrypted)]
- percent_complete [Percent Complete]
- ph_number [Phone Number]
- phone_number [Phone Number (Unused)]

- phone_number_e164 [Phone Number (E164)]
- price [Price]
- reference_name [Reference Name]
- related_tags [Related Tags]
- reminder_field_name [Reminder Field Name]
- repeat_count [Repeat Count]
- repeat_type [Repeat Type]
- replication_payload [Replication Payload]
- schedule_date_time [Schedule Date/Time]
- short_field_name [Short Field Name]
- short_table_name [Short Table Name]
- Slushbucket [slushbucket]
- source_id [Source ID]
- source_name [Source Name]
- source_table [Source Table]
- string_boolean [<none>]
- sys_class_name [System Class Name]
- sysrule_field_name [System Rule Field Name]
- table []
- text []
- time []
- timer [Timer]
- translated [Translated]
- tree_code [Tree Code]
- tree_path [Tree Path]
- user_image [Image]
- user_input [User Input]
- variables [Variables]
- version [Version]
- video [Video]
- week_of_month [Week of Month]
- wide_text [Wide Text]
- wms_job [WMS Job]
- workflow [Workflow]

Related topics

Resolve a collision in Team Development

Resolve multiple collisions

You can resolve multiple collisions without reviewing the differences between the local and pulled versions.

Procedure

1. Navigate to **All** > **Team Development** > **Team Dashboard**.
2. In the control panel, click the number of collisions.
   A list of collisions opens.
3. Select the check boxes beside the rows you want to resolve.
4. In the *Actions* choice list, use one of the following methods to resolve the collision.

Option	Description
To load the version pulled from the parent as the current version for all selected collisions	Select **Use Pulled Version**
To maintain the local version (local record) as the current version for all selected collisions. **The pulled versions are added to the version history for the records.**	Select **Use Local Version**

Team Development setup

To enable parallel development on multiple non-production instances, administrators can set up the Team Development instance hierarchy and grant access rights for developers.

Access rights for developers

To use Team Development, application developers must have a set of credentials for each instance in the Team Development hierarchy.

An instance's placement in the team development hierarchy determines the credentials it requires.

Credentials for Team Development access

Desired Access	Credential Requirements
Access to the Team Development application	A user with the admin role on the instance you are accessing
Right to register a remote instance	One of following: • A user with the admin role on the instance you are registering • A user with the teamdev_user role on the instance you are registering
Right to push changes to the parent instance from a development instance	One of following:

Credentials for Team Development access (continued)

Desired Access	Credential Requirements
	• A user with the admin role on the parent instance • A user with the teamdev_user role on the parent instance
Right to compare to a registered remote instance	One of following: • A user with the admin role on the registered development instance • A user with the teamdev_user role on the registered development instance
Access to the Code Review Requests module	One of following: • A user with the admin role on the parent instance • A user with the teamdev_code_reviewer role on the parent instance

Note:
The teamdev_user role does not grant access to the Team Development application and is not intended for developers to work on local development instances. It is intended to grant developers non-admin access to remote instances such as the parent instance or a peer development instance.

Create an exclusion policy

Application developers can create an exclusion policy to prevent pushes or pulls to particular instances in the team development hierarchy.

Procedure

1. Navigate to **All** > **Team Development** > **Exclusion Policy**.
2. Click **New**.
3. Complete the Exclusion Policy form (see table).
4. Click **Submit**.

Exclusion policy form

Field	Description
Name	Enter a unique description of the policy.
Policy	Select when the policy applies. Options include: ○ Push only ○ Push and Pull ○ Pull only

Field	Description
Remote Instance	[Optional] Select a specific remote instance to ignore changes from during pull operations. Leaving this field blank ignores changes from all remote instances.
Table	Select which table to ignore changes for.
Conditions	Select any additional criteria a change must meet to be ignored other than the table name. This field is only visible when the *Policy* is Push only.

Define a remote instance

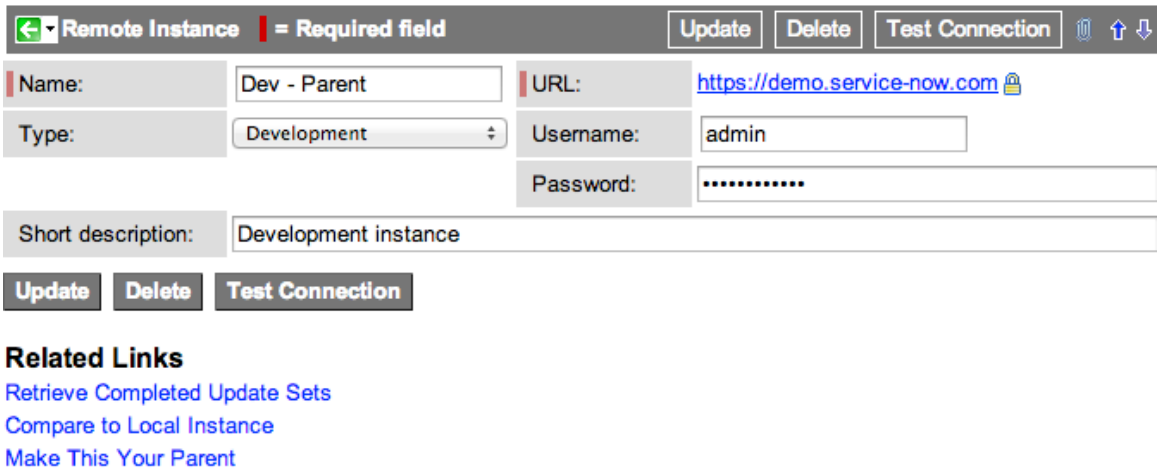
For each instance, define other instances in the hierarchy as remote instances.

About this task

For example, to set up remote instances for Sub-Dev 1:

Procedure

1. If IP address access control is enabled, log in to the remote instance and add Sub-Dev 1 as an exception.
2. On Sub-Dev 1, navigate to **Team Development** > **Remote Instances**.
3. Click **New**.
4. Define the remote instance, such as Dev-Parent, by completing the form (see table).



5. Click **Submit**.
6. Repeat step 1 through step 5 for each instance in the hierarchy that this instance needs to push and pull with (for example, Sub-Dev 2 and Sub-Dev 3).

Remote instance form

Field	Description
Name	Enter a unique name describing the instance.
Type	Specify whether the remote instance is a development, test, or UAT instance.
Active	Specify whether the local instance communicates with the remote instance as a member of Team Development. Team Development operations such as comparing changes between instances or selecting a parent instance are only available for active remote instances.

Field	Description
URL	Specify the URL of the remote instance using the appropriate transfer protocol. Each remote instance record should have a unique URL. Creating duplicate records with the same URL can cause errors.
Username	Enter the user on the remote instance who authorizes Team Development operations on the instance. This user account must have an appropriate role on the remote instance.
Password	Enter the password of the authorizing user.
Short description	[Optional] Enter any other relevant information about the remote instance.

Enable a code review

You can require a code review of all changes pushed to an instance.

Procedure

1. Navigate to **All** > **Team Development** > **Properties**.
2. Select the **Yes** check box for *If this property is set to Yes, code review is required before pushing to this instance* (com.snc.teamdev.requires_codereview).
3. Click **Save**.

   Setting this property adds the Code Review Requests module to the application menu and requires all changes pushed to this instance to remain in the *Awaiting Code Review* stage until someone in the Team Development Code Reviewers group approves them.

Select the parent instance

An instance can have multiple peer instances but only one parent instance.

About this task

The parent instance is the only instance you can pull changes from and push changes to.

The parent instance must be on the same release family as the local instance. For example, a development instance on the Geneva release family must have a parent instance also on the Geneva release family. If you select a parent from a different release family, the Team Development dashboard displays an error message and prevents you from pulling changes and reconciling. If you select a parent from a different patch release, the dashboard displays a warning message but allows you to pull changes and reconcile.

Do not use Team Development with production or test instances.

- Do not use a test or production instance as the parent instance in Team Development.
- Do not make any instance the parent of a production instance.
- Production instances should never have a parent.

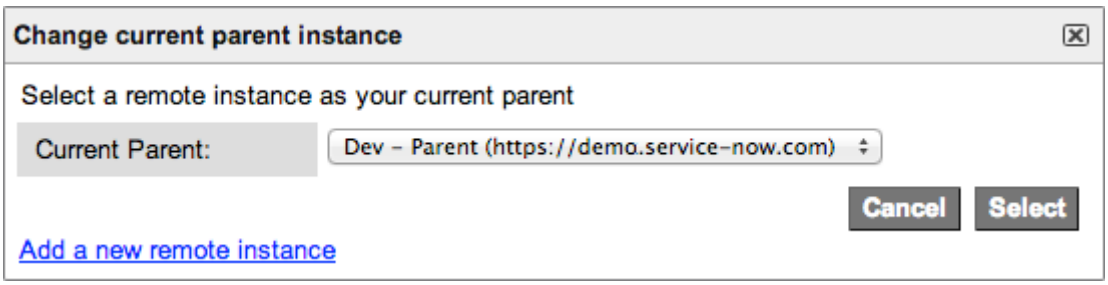
When you back out a change on a Team Development instance, it backs out the change all the way back down the chain, including undoing the work on the source instance. This behavior can cause major problems on test and production instances.

Procedure

1. Navigate to **All** > **Team Development** > **Team Dashboard**.
2. In the control panel, click the appropriate link:

Team dashboard control panel options

Option	Description
Use <instance name and URL>	Selects the most recently defined remote instance as the parent instance.
Select a different instance	Opens a dialog box where you can select another remote instance or define a new remote instance.
Register a new instance or List all remote instances	Opens the remote instance form or list, where you can define a new remote instance. These options are available when no remote instances are defined.



3. If you defined a new remote instance in step 2, repeat step 1 through step 2 and select the remote instance you defined.

   The system initiates a reconcile, which compares the local instance to the parent. It then generates the list of local changes and calculates the number of changes that are ready to pull from the parent. The reconcile also validates the instance versions.

4. Pull all changes from the parent instance if both instances are in the same release family.

Note:
The parent instance is saved in the `glide.apps.hub.current` system property.

Set up an instance hierarchy

Set up an instance hierarchy that best supports your development life cycle.

About this task

This example demonstrates how to set up an instance hierarchy where several peer sub-development instances have the same parent development instance, but a more complex configuration may be required to handle multiple project teams or other customer requirements.

Do not use Team Development with production or test instances.

- Do not use a test or production instance as the parent instance in Team Development.
- Do not make any instance the parent of a production instance.
- Production instances should never have a parent.

When you back out a change on a Team Development instance, it backs out the change all the way back down the chain, including undoing the work on the source instance. This behavior can cause major problems on test and production instances.

Procedure

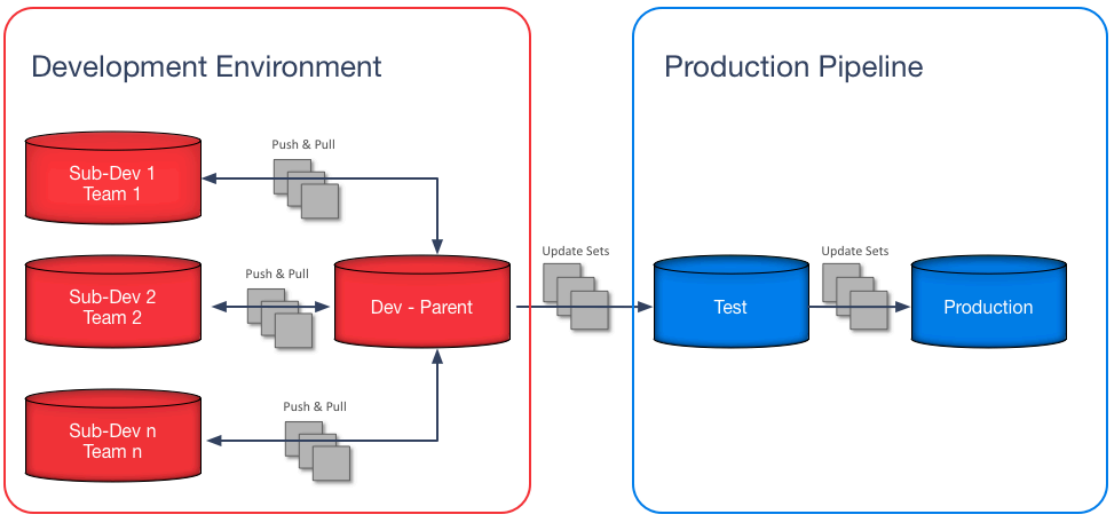
1. Provision a parent development instance on the same software version, such as Dublin, as the target instance, such as production.
2. Clone the production instance to the parent development instance.
   This step is a recommended best practice.
3. Provision sub-development instances on the same software version as the parent development instance.
4. **Optional:** Log in to the parent development instance and clone it to the sub-development instances.
5. On each sub-development instance:

   a. Define remote instance connections to other instances in the hierarchy that this instance needs to push and pull with.

   b. Select the parent instance.

   c. Pull all changes from the parent instance.

   d. Grant access rights to appropriate developers.



Code reviews

Team Development administrators can require that pushes undergo code review before accepting pushes.

When code review is enabled, pushing a change to the parent instance triggers the code review workflow. By default, users with the teamdev_code_reviewer role receive notifications to review changes and can approve or reject changes. The Team Development Code Reviewers has the teamdev_code_reviewer role.

For each change, reviewers can see the following information.

- Which remote instance the pushed change comes from.
- Who pushed the change to the parent.
- What the change is called.

- When the change was created.
- Which versions the change includes.

Reviewers must approve or reject a push from the Team Development application.

While changes are being reviewed on the parent instance, a child instance cannot do the following activities involving the parent instance:

- Push changes to the parent instance.
- Pull changes from the parent instance.
- Reconcile changes with the parent instance.
- Change the parent instance to another instance.
- Delete the remote instance record for the parent instance.

Code review notifications

You must enable email notifications on the instance requiring code review for that instance to send code review notifications.

The Team Development Code Review workflow sends notifications to members of the Team Development Code Reviewers group when:

- A push requires code review.
- A user cancels a push.

If the user who pushed the changes has a user record with an email address on the instance where code review was required, the user receives a notification when the approval stage is set to *Complete* (approved) or *Code Changes Rejected*.

The code review notifications contain the following information:

Code review notification table

Notification name	Table	Contents
Code review update for developer	Push or Pull [sys_sync_history]	• The push name • The approval stage of the push (approved or rejected) • A link to the instance where the code review request was made
Notify code reviewer of canceled review	Push or Pull [sys_sync_history]	• The user who canceled review • The push that was canceled

Code review workflow

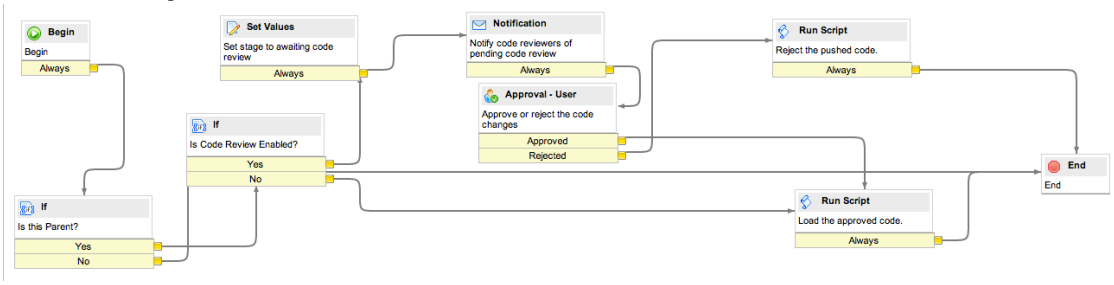
The Team Development Code Review workflow manages how changes are pushed to the parent.

By default this workflow:

- Starts when changes are pushed to the parent instance.
- Verifies that the code review property is active on the parent instance.

- Sets the stage of changes requiring approval to *Awaiting Code Review*.
- Notifies the Team Development Code Reviewers group to review pushed changes, if configured.
- Loads approved changes or sets the stage to *Code Changes Rejected*.

Team Development code review workflow



Warning:
Use caution when modifying this workflow, as the code review feature may not function properly.

Exclusion policies

You can exclude certain files from change tracking by creating an exclusion policy.

When a change matches an exclusion policy, the change does not generate records in the local changes list. The change still generates local version records and Update Set records as normal. See Creating an Exclusion Policy.

Note:
The exclusion policy applies to changes identified during a reconciliation operation. If you create an exclusion policy after a reconciliation, Team Development still tracks the changes until the next reconciliation.

Instance hierarchies

Team Development allows you to set up a distributed version control system between multiple ServiceNow instances where each instance acts as a source repository, or branch.

Developers use separate instances to work on different features, applications, or product releases at the same time. With Team Development, developers can share code between these instances and resolve collisions throughout the development process.

Team Development allows you to establish hierarchical relationships between instances and provides a mechanism for transferring changes between instances that integrates with the Update Set process where necessary. In a Team Development instance hierarchy, each non-production instance has a parent instance. Instances that have the same parent instance are peer instances. The shared parent instance becomes the central hub, or repository, and all peer instances synchronize to it.

Pulls and pushes

Developers synchronize their instances to the parent instance by pulling and pushing versions of customized records and resolving collisions between versions on the parent instance and the development instance.

Developers can compare peer instances to one another and share code or resolve collisions before pushing versions to the parent instance.

Pulling from the parent retrieves versions of records that have customer updates. Pulling retrieves all versions that have not already been pulled onto the development instance, including historical versions, and you cannot choose which versions to pull. You must resolve any collisions before proceeding with further pulls or pushes.

Pushing to the parent adds only the current development version to the parent, not all the development versions. You can choose which changes to push to the parent. Pushing creates a local Update Set on the parent that is marked as complete. Pushed versions are also tracked as local changes on the parent. Therefore, you can promote changes through your development and test hierarchy by transferring the Update Set or by pushing the local changes.

Comparing reports the differences between two peer instances. You can choose which versions to pull from a peer instance.

The *Pushes and Pulls* related list on the team dashboard displays the user who created a change and the remote instance where the change was created.

Team Development process

The basic Team Development process sets up the instance hierarchy, grants developer access rights, manages the movement of development changes from development instances to test instances, and promotes applications to the production instance.

Procedure

1. Set up the development instance hierarchy as described in Set up an instance hierarchy.
   a. Provision development instances on the same software version as the target instance. For example, use the software version that is running on your production instance.
   b. [Recommended] Clone the target to the development instances.
   c. For each instance, define the parent instance.
   d. [Optional] For each instance, define the peer instances.
   e. For each instance, pull all changes from the parent instance.
2. For sub-development instances, grant access rights to appropriate developers.
3. Develop customizations on sub-development instances.

   Use the team dashboard to track development activities.
   - Pull versions from the parent instance, such as versions that were pushed from other sub-development instances. Reconcile any conflicts with the current local version, as necessary.
   - Track local changes. Queue changes that are ready to push to the parent development instance.
   - Compare versions on peer instances. Reconcile any conflicts.
4. When a feature is ready to promote to the parent development instance, push the current version of the customized records.
5. **Optional:** Have code reviewers approve or reject the pushed version.
6. Test and promote the feature into production according to your testing and release management process.

Related topics

- Team Development
- Access rights for developers
- Push a version

Team Development roles

To use Team Development, developers must have admin access to their development instance.

To allow pushes to the parent instance, a remote instance connection must be defined with a user account that has admin access to the parent instance.

To limit developer access to the parent instance, see Granting Access Rights to Developers.

To use code review features, users must have the teamdev_code_reviewer role. See Code Review.

Versions

Version records track changes to a customized record over time so that administrators can compare or revert to specific versions later.

Administrators can also transfer versions between instances with Update Sets or team development.

Version record navigation

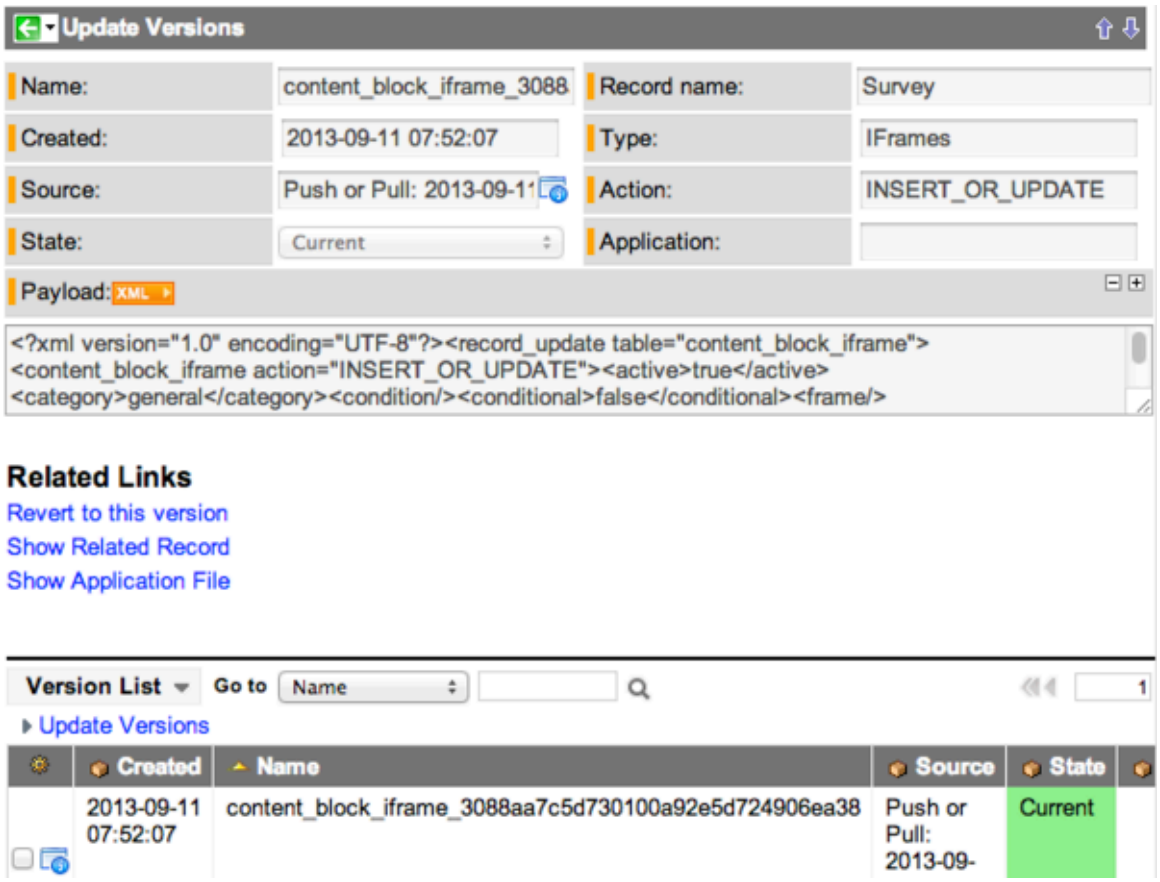
There are a variety of methods for viewing a list of versions for an object.

- For forms, right-click the header and select **Configure** > **Form Layout**.
- For lists, perform the appropriate action for the list version.
  - List v2: Right-click the header and select **Configure** > **List Layout**.
  - List v3: Open the list title menu and select **List Layout**.
- Click the **Show Versions** related link.
- For tables that use the update_synch attribute, add the **Versions** related list to the form. This list is on several forms by default, including, business rules, UI actions, and client scripts.
- For any customizable object, right-click the form header and select **Show Application File**, then scroll down to the *Related Record Versions* related list.

You can navigate from a version record to:

- The customized object: Click the **Show Related Record** related link.
- The application file record for the object: Click the **Show Application File** related link.

Version record



Versions transferring

Administrators transfer version records between instances by moving customizations with Update Sets or the Team Development application.

- Update sets: committing an Update Set adds versions. For each update in the Update Set, the version that corresponds to the update is added on the local instance.
- Team Development:
  - Pulling retrieves from the parent instance all versions of customized records that have not already been pulled and then adds them on the local instance.
  - Pushing adds to the parent instance only the current local version, not all the local versions.
  - Loading changes from peer instances adds selected versions to the local instance.

Version records

The Update Versions `[sys_update_version]` table contains records that represent the state of a customizable object at a particular time.

A customizable record is any object that is tracked by Update Sets, such as business rules or script includes. A new version record is created automatically whenever a user changes a customizable record or changes the application file for the customizable record.

A record represents the version of a base system object as it was delivered in the most recent upgrade. Baseline versions are created only for objects that have been modified by a user, and they are updated each time the system is upgraded.

Update versions table

Field	Description
Name	A unique identifier for coalescing versions of the same customized record.
Record name	Name of the customized record.
Source	Indicator of how the version was added on the instance. • System Upgrade: from a software upgrade (the baseline version). • Update Set: from an update set that was created or committed on the instance. • Pull History: from a pull in Team Development.
State	Indicator of whether the version is or has ever been loaded on the instance. • Current: the version is currently loaded. • Previous: the version has previously been loaded on the instance. When a current version is replaced by a new version, it becomes a previous version. • History: the version was never loaded on the instance and was only inserted for historical purposes, such as when pulling versions from the parent in Team Development.
Application	The application for the customized record, if it is assigned to an application.
Payload	The data for this version of the customized record.
Additional fields on the list view	
Reverted from	A reference to the older version record, if this version was created by reverting to an older version.
Fields that can be added by configuring the form	
Instance Name	The name of the remote instance where the version was originally created.
Instance ID	The URL of the remote instance where the version was originally created.
Related lists on the form view	
Version List	All versions of the customized record that are available on the instance.

Related topics

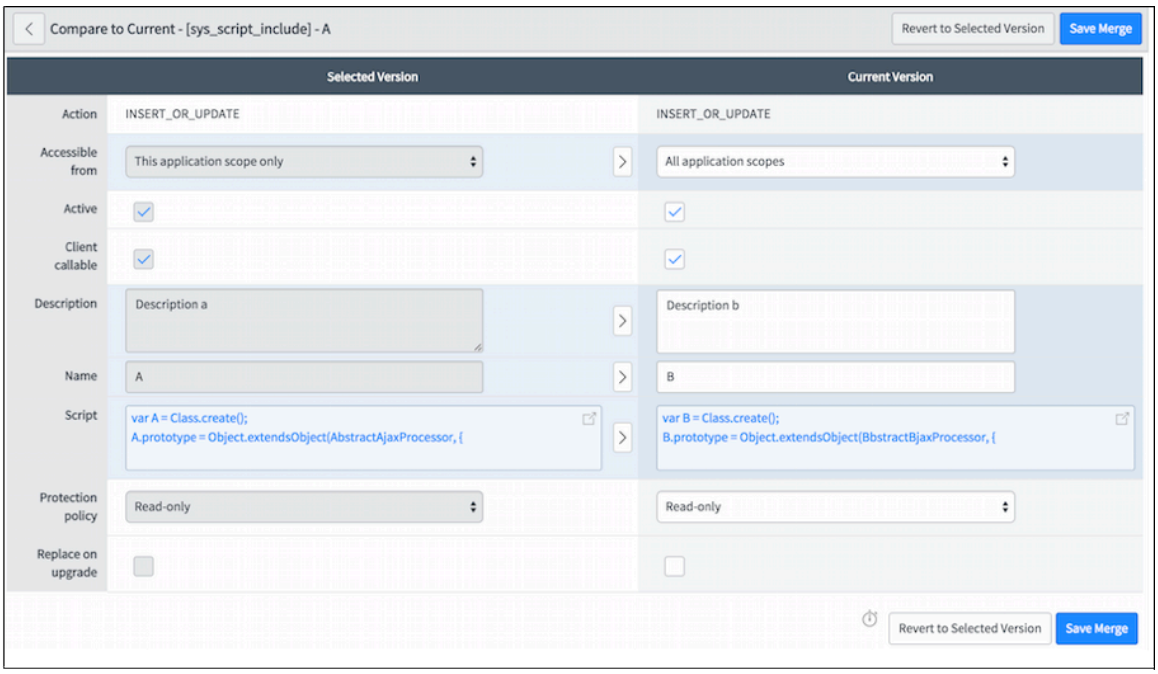
Team Development

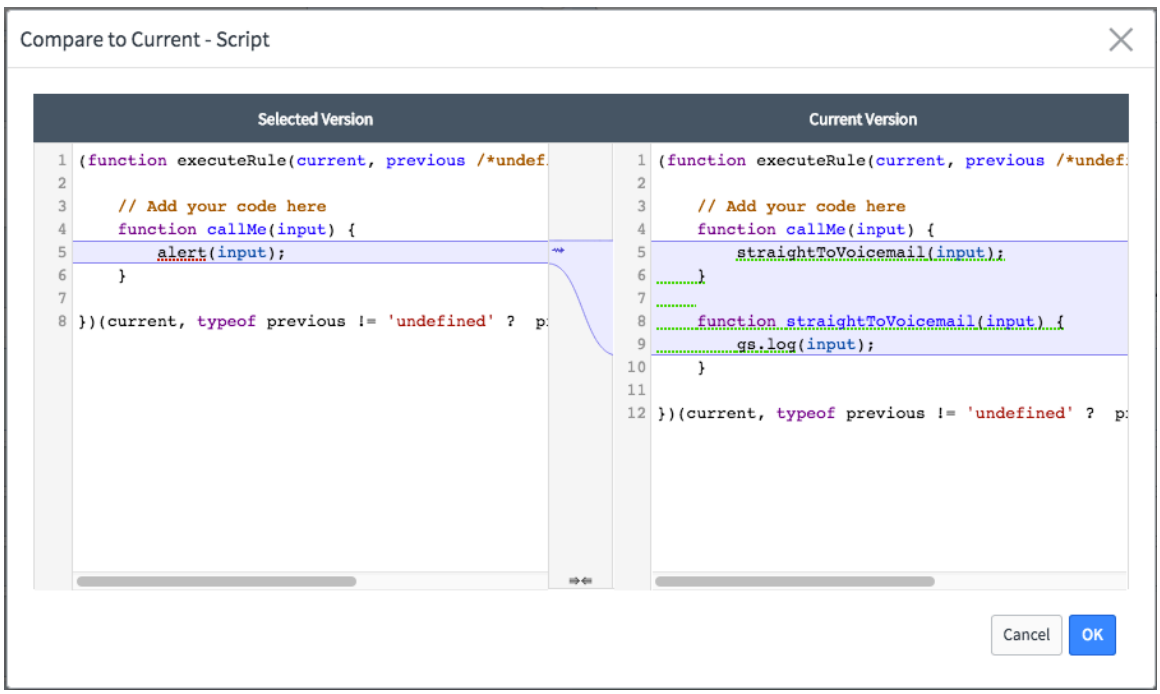
Merge tool

The Diff Merge tool enables administrative users to compare differences between two versions of a record.

Administrators can compare field-level changes between two versions, apply changes using Move Right field-level copy functionality and then merge results, or choose to revert to the non-

current version. You can access the Diff Merge tool by comparing versions, resolving conflicts, or resolving collisions, during development or after upgrades.





Accessibility Functions

The platform includes accessibility features that support Web Content Accessibility Guidelines (WCAG) 2.0 level A and make the interface accessible to users with disabilities. These features

improve the user experience when accessing platform functions with Using accessibility features.

In general, you can use the following set of standard keyboard navigation functions:

- Press **Tab** to navigate major groupings in a pre-defined sequence, including moving between standard interface controls (fields and lists) in a module, or between records within a tab.
- Press **Shift Tab** to move backwards in a pre-defined sequence.

Visually impaired users can navigate the Diff Merge tool. Screen readers can read all critical page content. All links and buttons can be reached when a section that is critical must be read. VoiceOver audible cues describe the content of the section that is necessary to read.

To enable accessibility functions, administrators should set these sys_properties:

1. Setting **glide.ui.javascript_editor** to false makes the following functions accessible:
   - Script fields (such as Script Include).
   - Side-by-side script comparison.
2. Setting sys_properties color settings enables high contrast visibility, which makes the left and right columns more accessible and easier to read by visually impaired users.
   - **mergetool.bg.left.highlight** - Left column cell color when values differ between versions.
   - **mergetool.bg.right.highlight** - Right column cell color when values differ between versions.
   - **mergetool.bg.left** - Left column cell color when version values are the same.
   - **mergetool.bg.right** - Right column cell color when version values are the same.

Related topics

Compare to the current version

Compare a pushed version to a local version

Compare two versions of an article

Resolve conflicts for an individual record

Resolve a collision in Team Development

Revert a change

View customizations and compare with current version

Compare to the current version

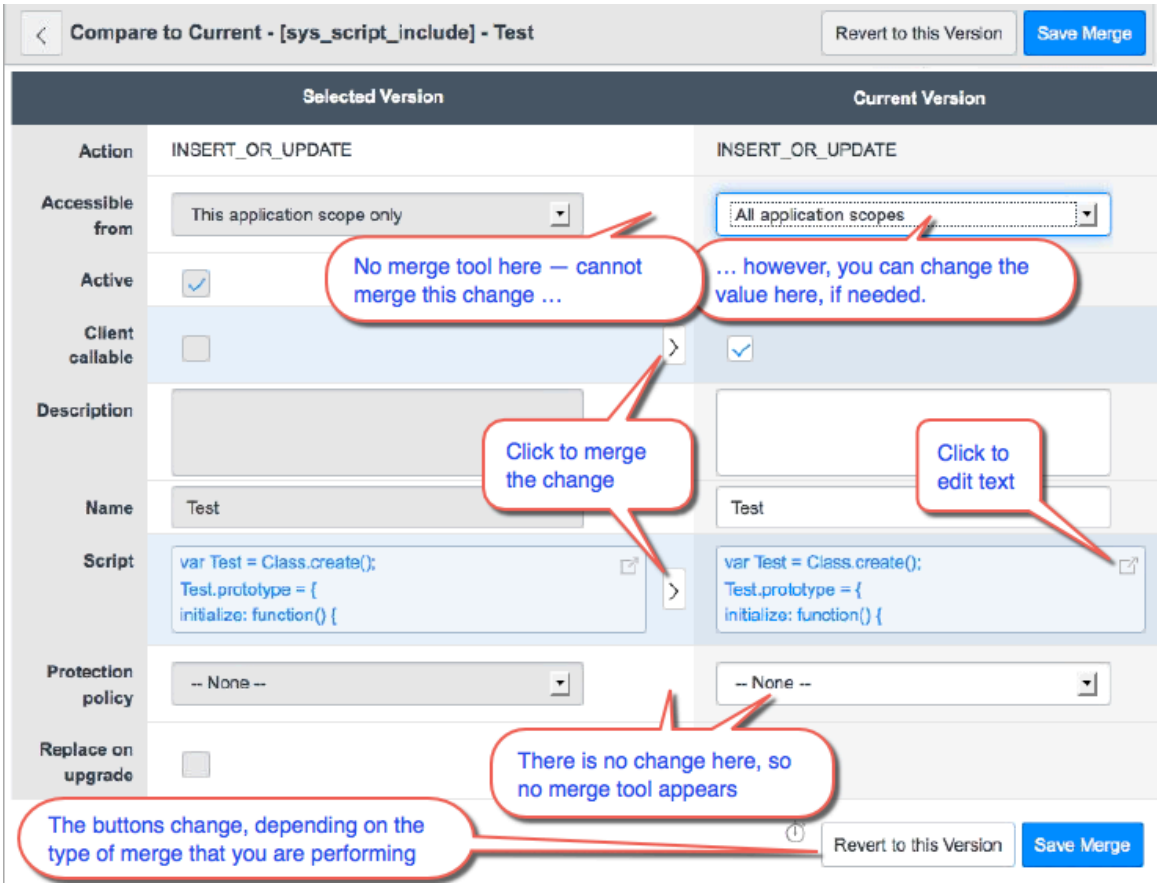
You can compare a version to the current version for any customizable object that a user has modified, such as a form layout or business rule. You can also compare the local and current pulled version of an object in Team Development. Administrators can suppress versions for specific tables.

About this task

To compare a version to the current version of an object:

Procedure

1. Open the Compare to Current page using one of the methods given in the below table.



2. On the Compare to Current page, review the fields that differ.

Related topics

- Merge tool
- Compare to the current version
- Compare a pushed version to a local version
- Compare two versions of an article
- Resolve conflicts for an individual record
- Resolve a collision in Team Development
- Revert a change
- View customizations and compare with current version

Revert a change

You can undo changes to a customized record by reverting to an older version.

Procedure

1. View a list of version records for an object.
2. **Optional:** Compare the current version to the older version to ensure that you are reverting the desired changes.
3. Right-click the older version and select **Revert to this version**.
   A confirmation dialog box appears.

If reverting to this version results in data loss due to a database schema change, a warning message appears in the dialog box.

4. Click **OK** to confirm the action.
   - The current version is marked as a previous version.
   - A new version record is added that duplicates the version that you selected in the preceding step. This new version is marked as the current version.

Note:
You can revert to the most recent baseline version. You cannot revert to an older baseline version.

Related topics

Merge tool

Compare to the current version

Compare two versions of an article

Compare a pushed version to a local version

Resolve a collision in Team Development

Resolve conflicts for an individual record

View customizations and compare with current version

Suppress versions

Administrators can configure a table so that it does not track customizations in the Versions `[sys_update_version]` table.

About this task

Warning:
If you suppress versions for tables, Team Development may work incorrectly, and you may be unable to compare and revert versions of records on the tables.

Procedure

1. Navigate to **sys_properties.list**.
2. Create a new property:
   - Name: glide.update.suppress_update_version
   - Type: string
   - Value: a comma-separated list of tables. The default value is sys_user,sys_import_set_row.

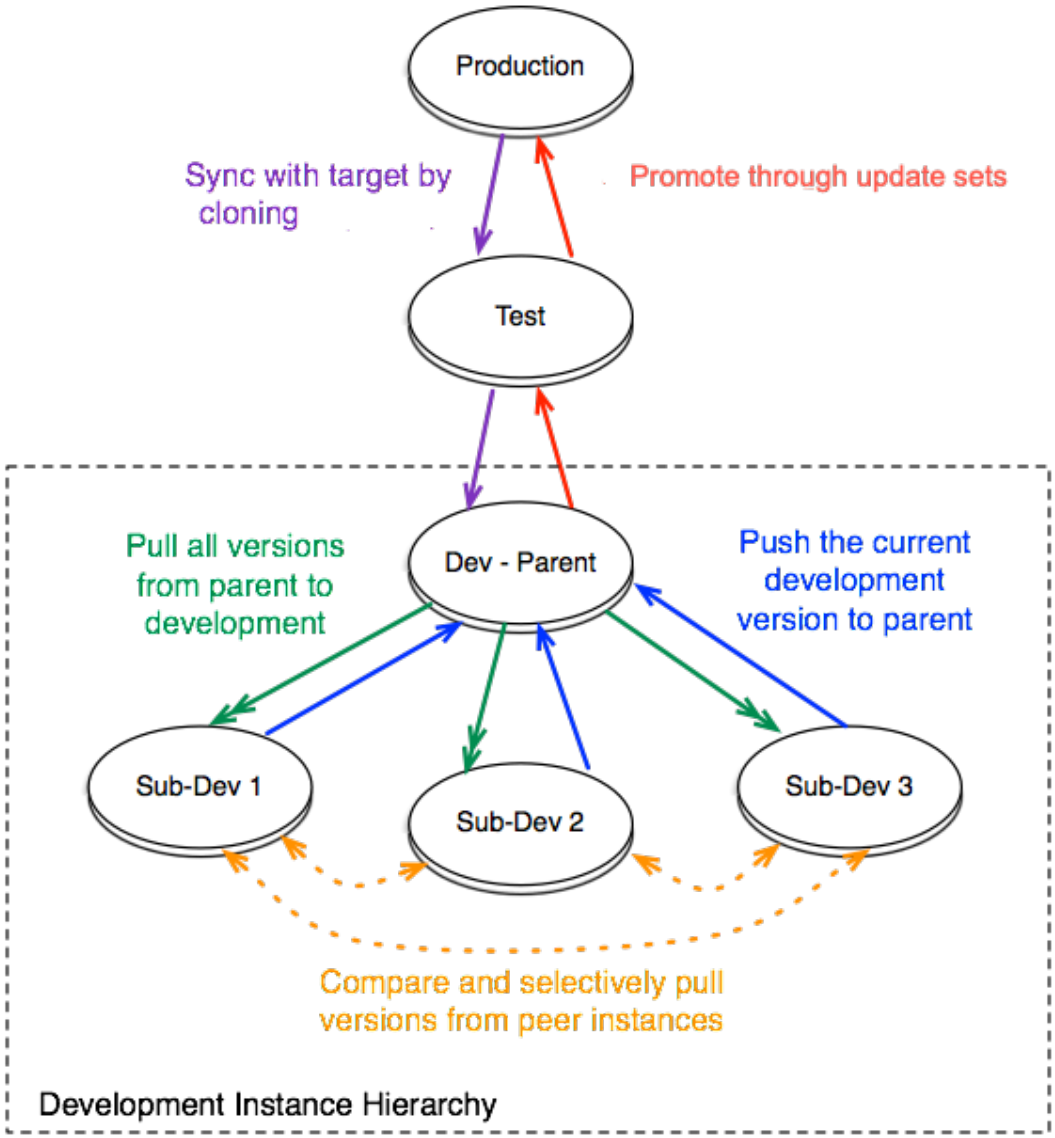
Versions and local changes

Version records track changes to a customizable record over time so that you can compare or revert to a specific version later.

A version record is created every time a developer changes a customizable record, so a single record can have multiple version records associated with it. A local change record is created or updated to reference the current version every time a developer modifies a customizable record, so a single record can have only one local change record associated with it.

Local change records track which customized records have changes on the development instance that are not on the parent instance so that you can collect changes in preparation for a push.

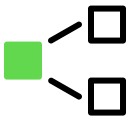
Team Development concepts



Developers can back out a local change to restore a previous version of a customizable record. The back out action sets the local customizable record to the last revision identified by a reconciliation action.

Developing your application

Build a custom application to meet the business needs of your organization. Choose a builder for the type of user experience or workflow that you want to create.

Build no-code applications with Creator Studio  Create and manage custom applications in a simple, no-code environment.	**Build low-code applications with App Engine**  Accelerate innovation with more creators and less complexity. Empower developers of all skill levels to build applications at scale.	**Build applications with ServiceNow Studio**  Empower developers with a modern, unified environment for building on the ServiceNow AI Platform using integrated builders and tools.
Build pro-code applications  Extend or maintain existing applications and user experiences. Connect applications and data across the enterprise.	**Builder library**  Choose a builder tool to quickly develop an application.	

Which builder should I use to create an app?

Types of builders

Want to build an app easily, without code?

Creator Studio specializes in helping you craft request-fulfillment applications without writing code. For example, an application to request office supplies by filling out a form, and someone approves or denies your request. For more information, see Exploring Creator Studio.

Need a more general app but still want low-code options?

App Engine Studio lets you build a broader range of apps than Creator Studio without being a programming pro. For more information, see Exploring App Engine Studio.

Are you a developer who wants more control in a centralized user interface?

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity. For more information, see Exploring ServiceNow Studio.

Are you a developer who wants to use industry-standard development tools and processes?

The ServiceNow IDE and ServiceNow SDK support developing applications in source code with ServiceNow Fluent, creating JavaScript modules, and using third-party libraries. ServiceNow Fluent is a domain-specific programming language for creating application metadata in code.

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. The ServiceNow SDK uses Visual Studio Code Desktop locally. For more information, see Building applications in source code.

Build no-code applications with Creator Studio

Create applications without coding

Build request-fulfill applications in a simple environment, without using code, in Creator Studio.

Build low-code applications with App Engine

Build low-code apps quickly, with more creators and less complexity. Safely scale cross-enterprise experiences that users want.

Empower creators

Bring creator workflow apps to production quickly for mission-critical tasks. Design with guidance and templates that are all within a holistic low-code dev experience.

Scale low-code development

Empower business and IT to collaborate, manage, and govern low-code app development.#Set development guardrails, apply standards, and check for compliance, all in one place.

Accelerate process automation

Automate processes fast and show value quickly with native integration and low code. Scale your workflows from simple to complex, with consistency across the enterprise.

Engage your users

Captivate users with a modern unified experience that's easy to understand. Build mobile-first experiences using an intuitive, low-code designer.

Build applications with the new ServiceNow Studio

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity.

Build pro-code applications

Build and deploy apps with fine-grained control. Debug code, manage source control, and publish your apps from a central hub.

Build your apps faster

Get all your work done in one place, accelerating the process from coding to deployment.

Develop applications in source code

Write code that defines application metadata in an integrated development environment.

Manage your source

Commit, branch, and merge to your Git repository with integrated source control.

Work as a team

Provide individual developer access to specific application resources for better collaboration.

Builder library

Each builder fulfills a specific need or produces a specific type of data, such as Decision Builder, UI Builder, and Workspace Builder. For a complete list of builders, see Builder Library.

Building no-code applications

Create and manage custom applications in a simple, no-code environment.

Creator Studio

Have you ever wanted to create an application, but you don't know how to code? Then Creator Studio was designed for you!

Which builder should I use to create an app?

Types of builders

Want to build an app easily, without code?

Creator Studio specializes in helping you craft request-fulfillment applications without writing code. For example, an application to request office supplies by filling out a form, and someone approves or denies your request. For more information, see Exploring Creator Studio.

Need a more general app but still want low-code options?

App Engine Studio lets you build a broader range of apps than Creator Studio without being a programming pro. For more information, see Exploring App Engine Studio.

Are you a developer who wants more control in a centralized user interface?

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity. For more information, see Exploring ServiceNow Studio.

Are you a developer who wants to use industry-standard development tools and processes?

The ServiceNow IDE and ServiceNow SDK support developing applications in source code with ServiceNow Fluent, creating JavaScript modules, and using third-party libraries. ServiceNow Fluent is a domain-specific programming language for creating application metadata in code.

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. The ServiceNow SDK uses Visual Studio Code Desktop locally. For more information, see Building applications in source code.

Related applications and features

App Engine Management Center

Track and manage your Creator Studio requests, deployments, applications, and collaborative developers using App Engine Management Center (AEMC).

App Engine Studio

Open Creator Studio apps in AES to build more custom experiences, such as email notifications and additional security.

Catalog Builder

Create or edit a catalog item using a visual and guided experience.

Form Builder

Visually create, configure, and customize the different form views for your users using the form editor in Form Builder.

Workflow Studio

Integrate workflow authoring, configuring, and monitoring into a single-page experience.

Workspace

Address customer requests and issues in a workspace, which provides a suite of tools where agents, case managers, help desk professionals, and managers work with tools to resolve customer needs.

Creator Studio

If you've ever wanted to create an application but you don't know how to code, then Creator Studio was designed for you!

https://player.vimeo.com/video/1024834247?badge=0&autopause=0&player_id=0&app_id=58479

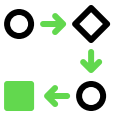
Creator Studio specializes in helping you craft request-fulfillment applications. In these types of apps, one person makes a request, such as for a new keyboard, and another person fulfills the request by providing them with one.

Keyboards, auto parts, recognition gift cards — imagine all the items employees use and ask for at your company. Your app could handle requests to approve travel, receive new equipment, make purchases, or get permission to take vacations. Your app would also enable someone to review, approve, and reject the requests.

Okay, so Creator Studio won't let you write an application for the next lunar landing module, but for what it's designed to do, it can do a lot!

Don't worry if you're new to creating applications. We're here to guide you every step of the way. Are you ready to dive in and start creating?

Get started

Explore  Learn about Creator Studio concepts and features.	**Configure**  Install and configure Creator Studio tools and access.	**Administer**  Administer Creator Studio requests and deployments.
Build  Build apps with forms and automation in Creator Studio.	**Reference**  Get details on Creator Studio components, such as fields, tables, and properties.	

Which builder should I use to create an app?

Types of builders

Want to build an app easily, without code?

Creator Studio specializes in helping you craft request-fulfillment applications without writing code. For example, an application to request office supplies by filling out a form, and someone approves or denies your request. For more information, see Exploring Creator Studio.

Need a more general app but still want low-code options?

App Engine Studio lets you build a broader range of apps than Creator Studio without being a programming pro. For more information, see Exploring App Engine Studio.

Are you a developer who wants more control in a centralized user interface?

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity. For more information, see Exploring ServiceNow Studio.

Are you a developer who wants to use industry-standard development tools and processes?

The ServiceNow IDE and ServiceNow SDK support developing applications in source code with ServiceNow Fluent, creating JavaScript modules, and using third-party libraries. ServiceNow Fluent is a domain-specific programming language for creating application metadata in code.

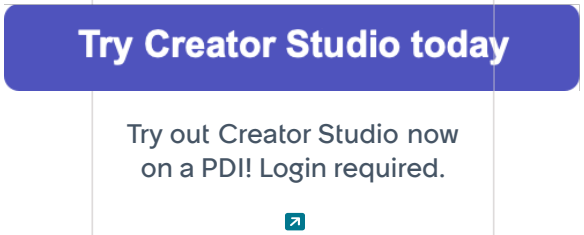
The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. The ServiceNow SDK uses Visual Studio Code Desktop locally. For more information, see Building applications in source code.

What you can do after you build an app in Creator Studio

If you build a simple app in Creator Studio and later decide that you want to make it more complex, you can open the app in any of the other ServiceNow builders. For example, you can add more complex automations in Workflow Studio or different interfaces in App Engine Studio.

Give Creator Studio a try

Ready to give Creator Studio a try? You can test it out using your own Personal Development Instance (PDI), which requires you signing in to the Developer Site. Find out more on PDIs in the Personal developer instance guide.



Exploring Creator Studio

Creator Studio makes creating apps easier by dividing their creation into simple steps. In this section, we'll explain each one.

All the steps involve requests. So, we'll start by explaining how Creator Studio stores requests.

Summary:
After reading this section, you'll understand:

- How requests are stored
- How people enter requests
- How your app automatically does things

Let's talk about storing requests

When people make a request, we have to store it somewhere so someone can review it later.

Tables hold requests

Creator Studio stores requests in a table. Each row in that table is a request. Here's a simplified version of a table with three requests.

Use cases:

- My monitor stopped working. I need a new one.
- I'm requesting a new mouse so I can have one at home and in the office.
- My laptop is over four years old. I want to request a replacement.

People enter requests using forms

How do people enter requests that eventually get stored in the app's table? They use your app to fill out a form. The questions on the form contain the information your fulfiller needs.

In our app's table, questions are stored as part of the record, which is composed of all the information on the form.

Request	Name	Email	Request date
I need a new monitor.	Sushma Singh	SushmaSingh@example.com	04-05-2024
I need a new laptop.	Peter Smith	PeterSmith@example.com	05-01-2024

In this example, the form has four fields:

- Request
- Requester's name
- Requester's email address
- Request date

These fields contain all the info the reviewer needs to accept or reject the request. You'll have to figure out all the information your reviewer needs to make a decision. Later, we'll show you how to create a form in the section Working with forms in Creator Studio.

Key terms:

Requester

Someone requesting something, like a piece of equipment or permission to do something.

Fulfiller

Someone who works on requests. Fulfillers may also approve or deny requests, depending on any approval automation for the app.

Okay, now you understand that requesters fill out forms to create requests and Creator Studio stores those requests in the app's table.

Try Creator Studio on a PDI

Want to play with Creator Studio on your Personal Development Instance (PDI)? It comes automatically installed from the Application Manager on your PDI. For more information, see Personal developer instance guide.

Example apps you can build in Creator Studio

Creator Studio enables you to build apps where people can make requests.

List of example apps

The following are some ideas of apps you can build in Creator Studio. These examples support ad-hoc requests that fall outside the scope of standard ServiceNow offerings, filling in gaps in non-routine business processes:

- Requesting marketing collateral
- Travel approval and booking apps
- Asset borrowing and returns, for example borrowing company art for an office
- New hire gift box
- Employee recognition awards and approval
- Help with using a product
- Request a head shot
- Backup coverage app where people can volunteer to cover for people on leave
- One-time funding requests, such as a retirement party
- Birthday party planning
- Internal skill sharing, where a department creates an app of skills they need for a project where people can sign up

Use App Engine instead of customization

App Engine development tools, such as Creator Studio, offer an excellent alternative to customizing existing applications on the ServiceNow AI Platform.

When your company needs to add new functionality to the ServiceNow AI Platform, you can customize existing applications, such as IT Service Management (ITSM), or create a new application using App Engine developer products, such as App Engine Studio, Creator Studio, or ServiceNow Studio. A simple guideline for which path to choose is:

- If the customization extends the intended purpose of the application, it works better to customize. For example, you can add IT functionality to ITSM.
- If the customization doesn't extend the intended purpose of the application, it works better to create a new application using App Engine developer products. For example, do not repurpose the ITSM workflow to add a travel request workflow.

Examples of when to use App Engine

ServiceNow products work best when they're used as they were intended. If you find yourself heavily customizing an application to repurpose it, a better plan is to create a new application using App Engine developer products.

The following scenarios demonstrate where creating a new application works better than heavily customizing an existing ServiceNow application:

- Your company has a business process that augments existing product functionality but doesn't follow the same workflow exactly.
- You have a novel use case for an app that doesn't align with any product workflow.
- You have a use case that could be built by heavily customizing an out-of-the-box application, but it doesn't align with what the existing application was intended to do.

Let's dive deeper into the last use case.

Issues with repurposing existing products

ServiceNow applications come with roles, processes, and flows that are specially tailored to their use case. For example, ITSM apps help with IT users, IT issues, IT reports, and IT cases.

You might have an idea for an app that's similar to but doesn't exactly align with ITSM. Because ITSM gives you a starting point, you might be tempted to customize ITSM to add the new functionality. For example, ITSM tracks IT issues, and a travel app you want to create might track travel requests. While the workflows sound similar, in fact, they use very different data, different user interfaces, and the details of each workflow vary greatly. Rather than heavily customize ITSM to repurpose it, a better plan is to use App Engine developer products for the following reasons:

- Combining two workflows creates conflicts.
- Customizing applications has implications.

Combining two workflows creates conflicts

In the ITSM example, the repurposing of ITSM to include a travel workflow uses different data, different tables, different roles, and different workflows than ITSM. As ITSM, ITSM customizations, and the travel workflow grow over time:

- Their features will continue to diverge.
- Adding new functionality or fixing problems in one workflow might adversely impact the other.

- The performance of ITSM may suffer.
- The code base will grow and the two purposes of ITSM will make troubleshooting more difficult.
- Quality engineers will require two different testing frameworks.

All these issues may cause unnecessary complications, poorer performance, upgrade delays, and software problems.

Customizing applications has implications

The ServiceNow AI Platform is built to embrace customization and configuration. The ServiceNow AI Platform is flexible enough to fit your company's business needs. How you customize ServiceNow applications, however, can have significant impacts on ServiceNow support, upgrading to future ServiceNow AI Platform versions, and the functionality of the platform.

Let's start by differentiating customization and configuration:

- Customization is any change made to the code that is part of the baseline installation of a ServiceNow instance. You use code to customize applications.
- Configuration is any change you make to the behavior of a product that does not touch the code in the baseline installation of a ServiceNow instance. You can use system properties, ServiceNow products, or code to configure an application.

The following are some of the implications that result from customizing applications:

- If you add code to an application, you own it whether or not it modifies the code in the baseline installation on a ServiceNow instance.
- The platform marks all customizations and skips them when you update to a new version of the platform. That means you are responsible for manually updating the customizations. This can have a significant impact on the time and resources required to update to new platform versions.
- The ServiceNow AI Platform uses a framework that supports applications in how they process tasks, how forms are rendered in multiple browsers, and the overall user experience. Introducing customizations can have unintended consequences on this framework.
- You own the burden of testing custom code and determining if it impacts platform functionality.
- ServiceNow Customer Support cannot troubleshoot custom code or issues caused by custom code.

Customization is one of the key features of the ServiceNow AI Platform. However, over-customizing an application to repurpose it is likely to generate technical debt, lengthen your upgrade cycle, and complicate future platform upgrades because the custom code may not easily migrate to new platform versions.

Conclusion

Customization and configuration are hallmarks of the ServiceNow AI Platform that enable your company to customize workflows to fit its specific needs. Proceed with these tasks in the following order:

1. Configure ServiceNow applications as much as you can before customizing them.
2. Customize an application only when it extends the intent of the application.
3. Use App Engine developer products, such as App Engine Studio, Creator Studio, and ServiceNow Studio, to create new applications rather than customizing an application to create functionality that doesn't align with its original purpose.

Customization vs. configuration with Creator Studio

There are important differences between customizing and configuring ServiceNow applications. The ServiceNow platform is built to embrace customization and configuration but how you do so can have significant impacts on ServiceNow support, upgrading to future ServiceNow platform versions, and the functionality of the ServiceNow platform.

The general rules around customization are:

- Customize an application only if it extends the original intent of the application. For example, add IT functionality to ITSM but don't add a travel workflow. Instead of over-customizing an application, create a new application using App Engine products, such as Creator Studio or ServiceNow Studio.
- Configure as much as you can before customizing an application.
- If you add code or make other modifications to out-of-the-box functionality, you own them.

What is configuration

Configuration is the process of using ServiceNow built-in tools and features to modify an application's behavior without making changes to flows or the code that is part of the baseline installation on a ServiceNow instance.

Configuration can take the form of using ServiceNow built-in tools to add tables and more, setting instance-wide parameters, as well as using code to extend an application's functionality to meet business needs as long as the code does not modify the baseline code installation. The entire platform is designed for you to add configuration code.

If you add code, such as workflow scripts, you own it, even if it doesn't alter the baseline code installation. That includes owning the impact it has on the entire ServiceNow platform. Issues that arise from added code are beyond the scope of ServiceNow support to debug.

Reverting a configuration should not require any change to the baseline code.

Configuration examples include:

- Forms: Configure tables, fields, data types, default values, and field dependencies to configure the data you capture and display.
- UI elements: Modify layouts, add related lists, add buttons, and change field names.
- Service Catalog: Configure portals where your customers can request catalog items such as service and product offerings.
- ACLs: Restrict unauthorized users from accessing forms and data.
- System property values: Modify the application's experience for all users.

What is customization

Customization is any change made to the flows or the code that is part of the baseline installation on a ServiceNow instance. You use ServiceNow products or code to customize applications.

If you add code, you own it, even if it doesn't alter the baseline installation. That includes owning the impact it has on the entire ServiceNow platform.

Customization examples include:

- Scripting: Customize ServiceNow through scripting using JavaScript. This includes creating client scripts, server-side scripts, and business rules with intricate logic that modify the baseline code.
- Custom tables: Develop custom tables to accommodate specialized data that doesn't fit within standard tables.
- Integration: Customize integration with external systems, such as APIs and web services, for seamless data exchange.
- Widgets and portals: Create custom widgets and portals to provide unique features and user experiences.
- Workflows: Create and modify workflows using Workflow Studio. Create and manage playbooks, flows, actions, decision tables, and integrations from one design environment to automate tasks. Upgrading to a new version of a flow requires reapplying your customizations.

Tools for customization and configuration

ServiceNow offers many tools and features, such as business rules, to modify the out-of-the-box behavior of ServiceNow applications. Whether they customize or configure an application depends on how they're used. Using these tools to modify the installed code base constitutes customization. Using these tools to add code that does not modify the flows, or the installed code base constitutes configuration. In both cases, you own the code you add as well as the impact it has on the ServiceNow platform.

ServiceNow tools include:

- UI Policies: Dynamically modify the visibility of fields and attributes on a form according to user inputs.
- Business rules: Automatically trigger actions based on specified conditions.
- UI Actions: Extend and customize forms and lists by adding buttons, context menu items, or other UI elements that perform specific actions when clicked.
- Client-side scripts: Scripts that execute within the user's browser when certain actions occur on a form or a UI page.
- Server-side scripts: Scripts that execute on the ServiceNow server or database, for example, to update record fields when a database query runs.

What is personalization

Personalization is when users use out-of-the-box application tools to modify an application's look and feel only for themselves. Admins can change the look and feel for all users and that is considered configuration. Personalization examples include a user choosing to use dark mode or choosing which table columns to display.

Personalization does not change the baseline code installation on a ServiceNow instance. So, personalization does not impact customer support or interfere with upgrades to new ServiceNow versions.

Ramifications of customizing ServiceNow products

The ServiceNow platform is extremely flexible and built to embrace customization and configuration to fulfill a wide range of business requirements. How you customize ServiceNow applications, however, can have significant impacts on ServiceNow support, upgrading to future ServiceNow platform versions, and the functionality of the platform. Instead of customizing ServiceNow applications, consider using App Engine development products, such as Creator Studio and ServiceNow Studio to create new applications.

The ServiceNow platform uses a framework that supports applications in how they process tasks, how forms are rendered in multiple browsers, and the overall user experience. ServiceNow relies on the framework's integrity to develop and provide support in a consistent manner. Customizations can impair this framework, change platform functionality, and impair workflows and upgradeability.

Customizations trigger the platform to create sys_update_xml records, which are stored in the Customer Update table. The platform marks all customizations and skips the customized records when you update to a new version of the ServiceNow platform. That means you are responsible for manually updating the customizations. This can have a significant impact on the time and resources required to update to new platform versions.

Note:
The Customer Update table also contains modifications or additions to configuration metadata, for example, creating a new catalog item or a new workflow.

For more information, see the Customer Updates table. Note that the complexity of maintaining customizations dramatically increases as the number of customizations increases.

Customizing the installed code base can be costly, generate technical debt, lengthen your upgrade cycle, and complicate future platform upgrades because the custom code may not easily migrate to new platform versions. Custom code can change the standard functionality of the ServiceNow platform in unintended ways. Evaluate demands for customization carefully and only resort to customization where there is clear business value and no alternative. Wherever possible, avoid customization by using configuration instead.

If you customize a product:

- You are responsible for maintaining the customization going forward.
- Customer Service and Support will not support problems caused by custom code. If it is the cause of the problems, the support team will likely advise you to revert to the out-of-the-box code.

What is the Customer Service and Support stance on supporting customization

The ServiceNow Customer Service and Support stance on customization is if you add code, you own it and its consequences. Why? Customer Support is not privy to your custom business logic, doesn't know what the expected behavior should be, is unable to reproduce the issue on an out-of-the-box instance, and customer support engineers are not certified implementation specialists, so they are not certified to review customized code logic.

Alternatives to customization

If you have requirements and ideas for enhancements, instead of customizing the installed code base, you can:

- Use configuration instead of customization.
- Submit an Enhancement Request to the ServiceNow development team. Each request is evaluated and, if approved, will be incorporated into a future release.
- Create an app using App Engine developer products to handle desired functionality.

When to use App Engine developer products instead of customizing

When your company needs to add new functionality to the ServiceNow platform, you can customize existing applications, such as ITSM, or create a new application using App Engine developer products, such as Creator Studio or ServiceNow Studio. A simple guideline for which path to choose is:

- If the customization extends the intended purpose of the application, it works better to customize. For example, you can add IT functionality to ITSM.
- If the customization doesn't extend the intended purpose of the application, it works better to create a new application using App Engine developer products. For example, do not repurpose the ITSM workflow to create a travel request workflow.

For example, ITSM is designed to handle IT issues. To customize it to handle travel requests goes beyond the original intention of ITSM. Because IT and travel requests have different workflows, it's better to create a travel request app using App Engine developer tools, such as Creator Studio and ServiceNow Studio, instead of customizing ITSM.

For more information, see Creator Studio.

Examples of when to use App Engine developer products

ServiceNow products work best when they're used as they were intended. If you find yourself heavily customizing an application to repurpose it, a better plan is to create a new application using App Engine developer products.

The following scenarios demonstrate where creating a new application works better than heavily customizing an existing ServiceNow application:

- You have a novel use case for an app that doesn't align with any product workflow.
- You have a use case that could be built by heavily customizing an out-of-the-box application, but it doesn't align with what the existing application was intended to do.
- Your company has a user group or business process that should be separate from the OOTB product workflow.

Guidelines for customizing ServiceNow products

If you must make a customization, consider the following suggestions:

- Maximize configuration options first.
- Avoid copying objects. Instead, update objects in place wherever possible, except for Service Portal widgets and other items designated to be reused.
- Default to "add before edit." This means that you should, for example, add fields to forms rather than change the type of an existing field. Avoid using the same names as out-of-the-box objects, methods, or classes when adding them.
- Minimize the number of fields you add to a form. The more fields you have on a form, the longer it may take to load.
- Export original records as backups before customizing them. Keep track of their out-of-the-box sys_id's in case you must restore them in the future.
- Use scoped applications as your default for any new custom development.
- Document all customizations. Add comments explaining why you customized (including business justification). Review all comments before upgrading to determine if you can revert to out-of-the-box code.
- Create tests for all customizations. Write Automated Test Framework (ATF) tests for all customizations where possible.
- Use HealthScan regularly to identify unnecessary customizations.
- Customizations should be made to baseline objects where necessary so that conflict resolution and decision-making can be appropriately recorded in the updates. Hidden

customizations may cause administrators to overlook updates in future assessments in case reverts or merges are necessary.

- Test your customization for all use cases. Include performance testing and the introduction of unintended consequences.
- Administrators are responsible for verifying that their customizations work after a ServiceNow platform upgrade, and for keeping track of what customizations are made.

Handling customizations when you upgrade

Customizations trigger the platform to create sys_update_xml records, which are stored in the Customer Updates table. These records are not updated during platform version upgrades. ServiceNow marks them as skipped records in the ServiceNow Upgrade Monitor. To make sure they're successfully ported to the upgraded instance, you must manually process the skipped changes. For more information, see the Customer Updates table.

Assuming you've documented all your customizations—including the business justification—take your documented inventory and compare it with the skipped records identified in the Upgrade Monitor. After filtering out low-risk changes that have resulted in skipped records (for example, field labels or form layouts), you'll need to decide whether to:

- Retain each customization
- Revert to out-of-the-box
- Merge your customization with the base system to resolve conflicts

Creator Studio quick start

This quick start guides you through the process of building your first app in Creator Studio and requesting its deployment.

At a minimum, you need to create the foundation of an app and customize its form, which is where users submit their fulfillment requests.

Building your first app is a good way to understand how Creator Studio enables easy app creation.

Your system administrator must add you to the Creator Studio Users group.

Quick start: Build the foundation of an app

Create your app's foundation before building it out.

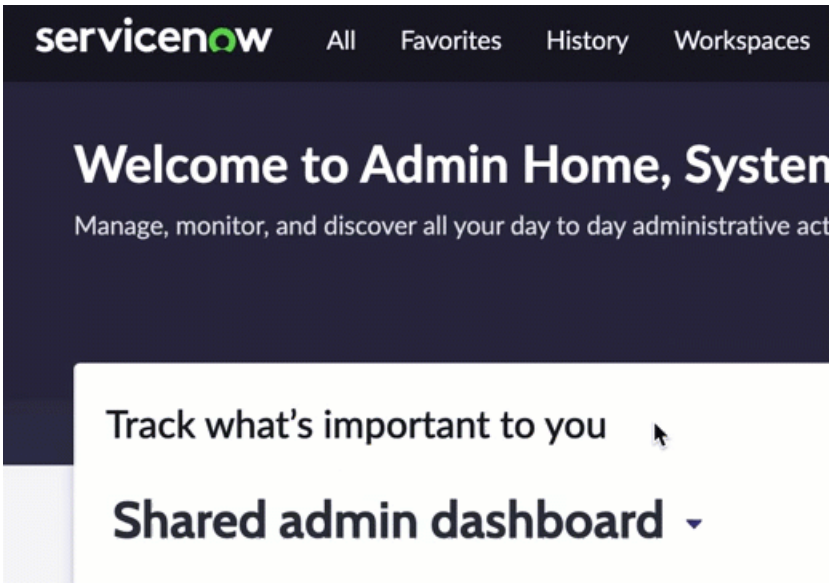
Before you begin

Your system administrator must add you to the Creator Studio Users group.

Procedure

1. Head over to the Creator Studio home page by going to **All** > **App Engine** > **Creator Studio**.

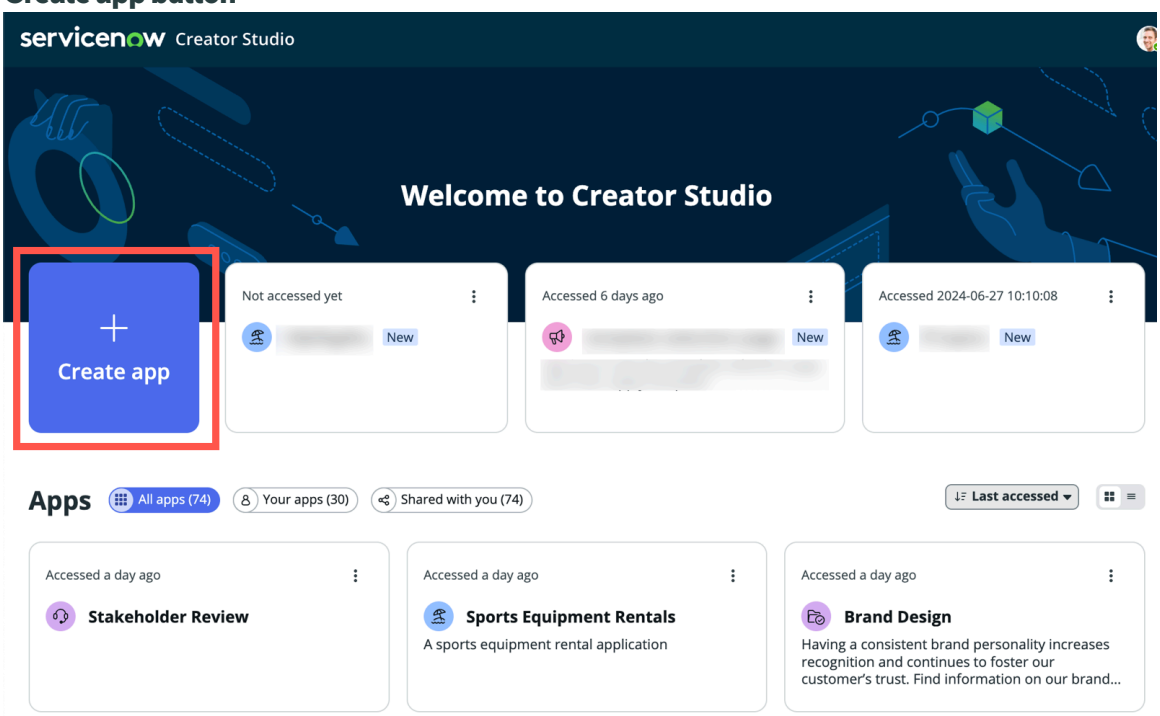
Start Creator Studio



The Creator Studio home page appears. To learn more about working in the home page, check out Find existing apps in Creator Studio.

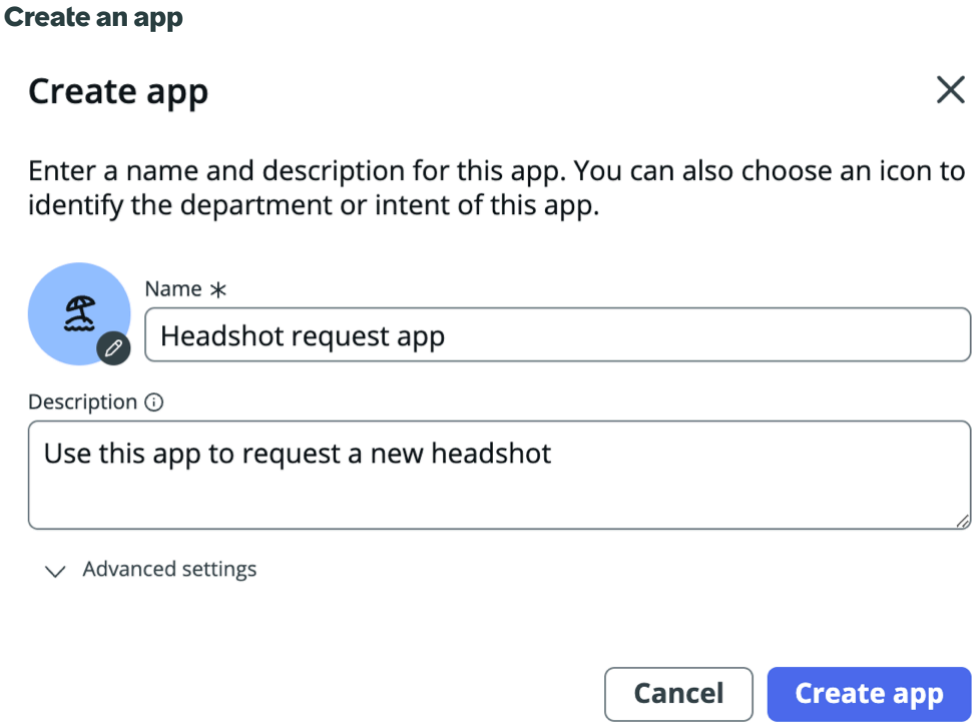
2. Select the **Create app** button to start the process of creating an app.

Create app button



- If you're a system administrator, you can read more about this topic in Application collaboration.
- If you want to know how to request an admin to create the app for you, check out Ask an admin to create an app for you in Creator Studio.

3. On the modal that pops up, enter a **Name** and brief **Description** of the app's purpose.



4. Select **Create app** to finish building the foundation of the app.

Result

Okay, you've created the beginning of your very own application! You've named and described it. A list of catalog templates appears.

Quick start: Choose catalogs and topics

Associate your app with catalogs and topics to determine and categorize their content.

Before you begin

Your system administrator must add you to the Creator Studio Users group.

Procedure

1. Select a catalog template from the list, which your admin can customize, and then select **Apply this template**.

   You may not see a catalog template that is exactly what you want. So, choose one that's the closest. If none look close, select the **Creator Studio Default Template** option, if your admin hasn't removed it. It's your best bet.

   Tip:
   Feel free to click through the list of catalog templates. Previews will help you pick the one you want.

   Don't worry if you don't see other catalog templates to choose from! Your admins may not have created any custom catalog templates for you yet.

2. Add a form to a catalog to specify its business area.

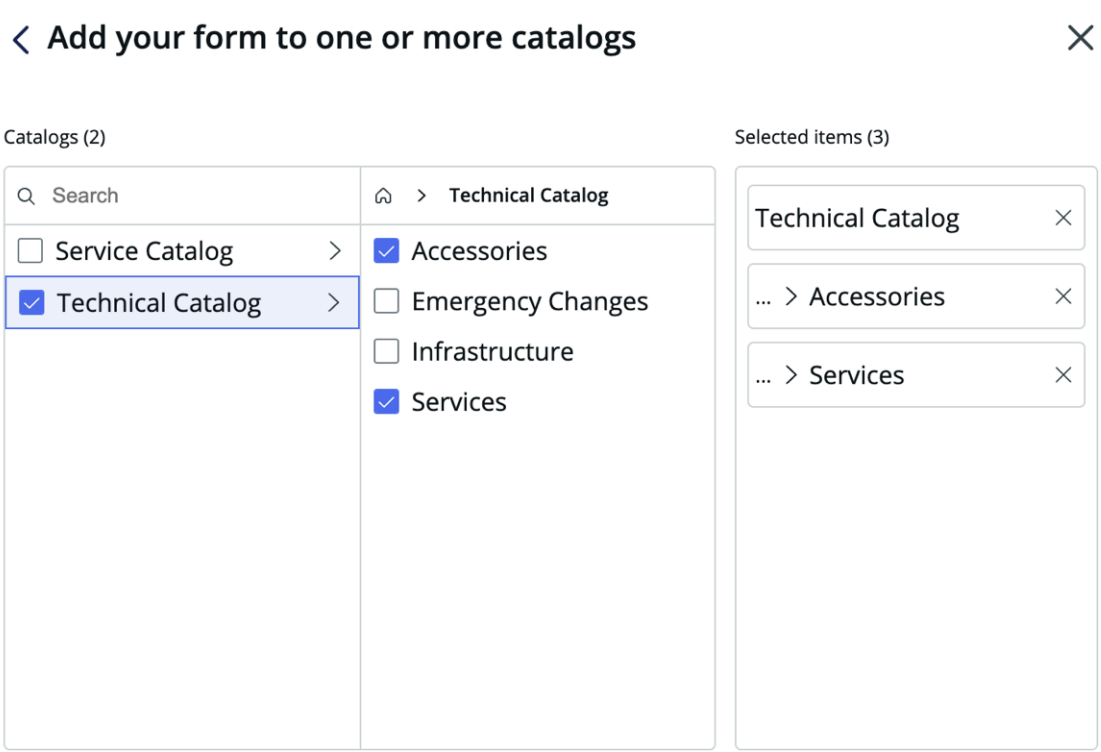
Note:
You can skip this step for now and revisit it later when you have a better idea of where it fits.

The available catalogs are configured by your admin, contact them if you don't see the one you want.

a. Select the **Edit button** (Edit) for the Catalogs and categories card.

b. Select the catalog that represents the business area the app will use.
For example, choose a service catalog that contains software and laptop cables. Expand the carat for each catalog to see its sub-catalogs.

Select the catalog



c. Select as many items in the catalogs as you need.

d. Select **Apply**.

You can edit any of the app's basic settings any time after you finish creating the app. For more information, see Creator Studio form settings.

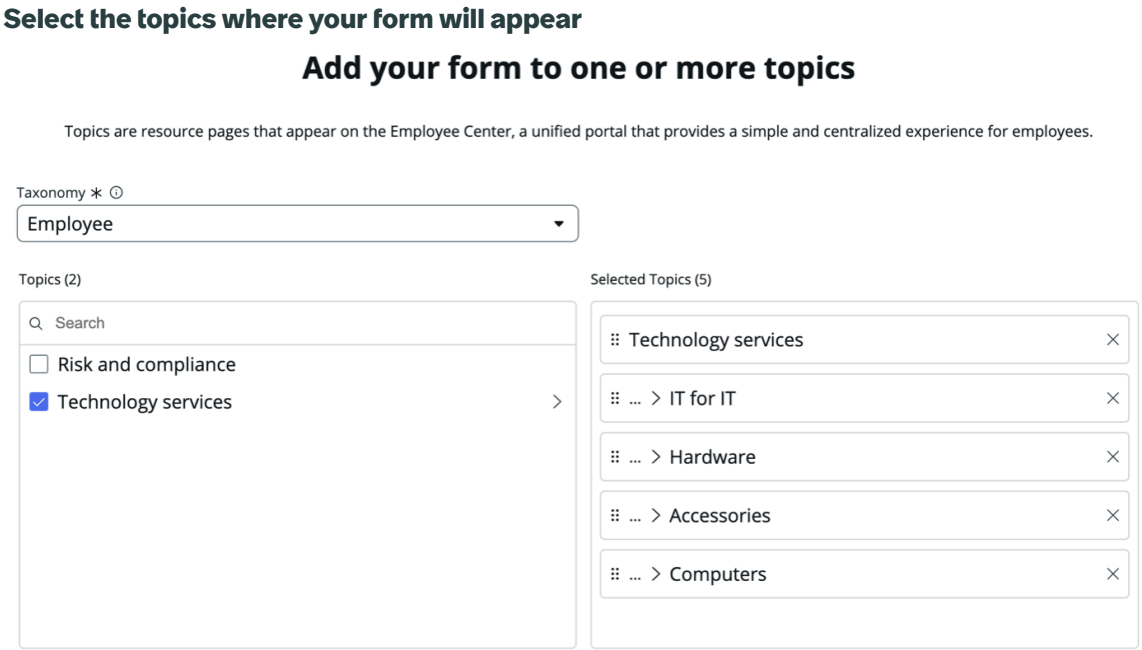
3. Next, choose one or more topics to specify where the form will appear.
Find out more about topics in Associate a catalog item with a taxonomy topic in Employee Center, and more about taxonomy, which is a categorization method, in Unified Taxonomy for Employee Center.

Note:
You can skip this step for now and revisit it later when you have a better idea of where it fits.

a. Select the **Edit button** (Edit) for the Topics card.

b. Choose the **Taxonomy** page where you want the form to appear, such as **Employee**.
Taxonomy is a method of categorization that Employee Center uses.

c. Select the topic(s) that represent the Employee Center areas where you want the form to appear.
For example, choose a topic that contains technology services, and expand its carat to see each of its sub-topics.



d. Select the topics where you want the form to appear, as many as you need.

e. Select the **Apply** button to save your changes.

4. Select **Save and continue**.

Result

Great, you've defined where the app will appear! Next, we'll customize how it looks.

Quick start: Customize your app's look and feel

Check out how your app will appear and then make adjustments to how it looks.

Before you begin

Your system administrator must add you to the Creator Studio Users group.

Procedure

1. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

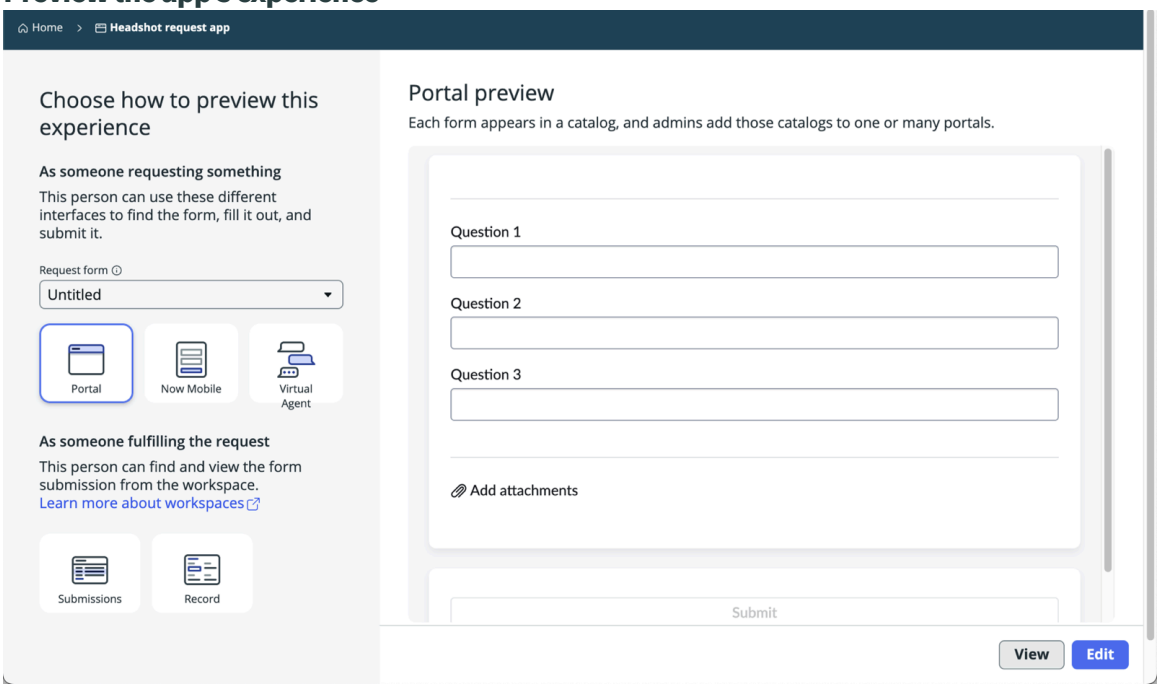
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

- **Portal** (see a preview of how it'll appear on a desktop website)
- **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
- **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

> **Note:**
> Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

Preview the app's experience



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

2. **Optional:** Next, you can customize the form's look and feel on the **Request forms** tab by completing one of the following steps.

   a. Add or modify the image that appears on your form by selecting the add image icon ( ) and then selecting an image.

   b. Change the form's title, short description, and other text by selecting those parts of the form and typing in your changes.
   You can enhance how the longer **Description** appears using rich text, such as font changes and sizing.

Result

Get more details on customizing your form in Customize your form for an app in Creator Studio.

Next, we'll build and publish a form.

Quick start: Add questions to and publish a form

Add questions to the form to gather the information that your fulfiller needs to evaluate the request, and then publish it to make its catalog items available.

Before you begin

Your system administrator must add you to the Creator Studio Users group.

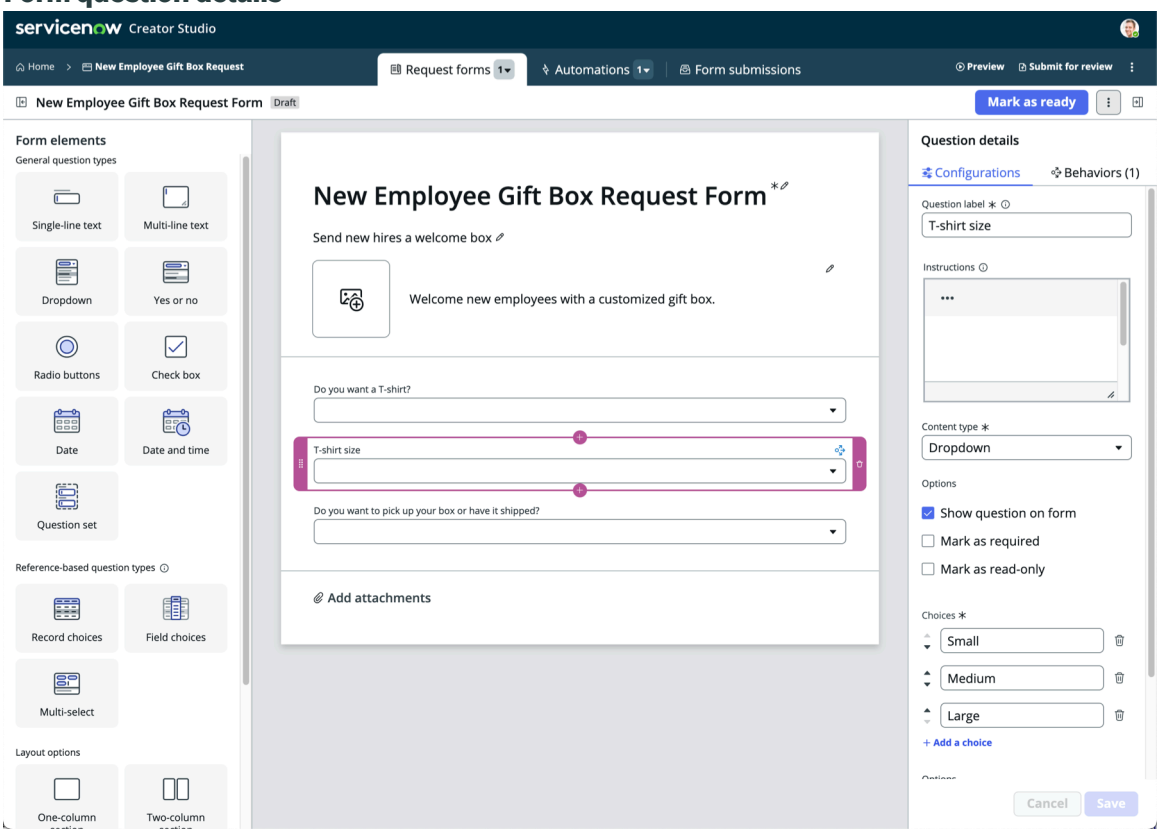
Procedure

1. In the Form elements panel, drag the type of question you want onto the form and drop it on the canvas where you'd like it. You can also add questions by selecting the add icon (+) that appears when you click on an existing question on the form.
   If you're adding a pre-configured **Question set**, you must select the question set from the modal that appears when you drag it onto the form.

   For a description of question types and how they're used, see Available question types in Creator Studio.

2. Select the question.
   When you select a question, it's highlighted on the form so you know what you're working on.
3. In the **Configurations** tab of the Question details panel, specify information about the question you added, such as whether a question must be answered for the requester to submit the form.
   The details vary by question type. For example, if you add a **Dropdown** question, you must supply the options to choose from.

Form question details



4. **Optional:** Make the form's appearance change based on how users answer questions by adding dynamic behavior to it on the **Behaviors** tab.

For example, if a user says they want a T-shirt for an event they're attending, you can make a **T-shirt size** field required. Get the details on adding dynamic in Make a form change based on responses in Creator Studio.

5. Select **Save and close** when you finish modifying the question.
6. Revise or add more questions to the form using this procedure. To change a question type, select the question and then select the new question type in the **Content type** field of the Question details panel. Selecting a new type may introduce new values you must supply. Keep these specifications in mind as you create your questions:
   - You can't change an existing question into a question set. To include a question set on a form, you must newly add it to the form.
   - If you put two checkbox questions side-by-side on a form, they make a section. You can't add other types of questions to that section.
7. Now, let's arrange the questions and images that you've added to the form.

   a. From the Form elements panel, drag the layout option you like onto the form and drop it where you want it to appear, for example, a **Divider line**.
   Don't worry if you don't like the layout, just try another one by dragging it onto the form's canvas.

   b. Revise the form layout you chose using the Section details or Question details panel (depending on the layout you're working on). You can do things like make text bold, add links, and so forth.

   Note:
   To edit or delete a section, you must hover over the section name and then select **Section** to see the section details in the properties panel, as well as the delete icon.

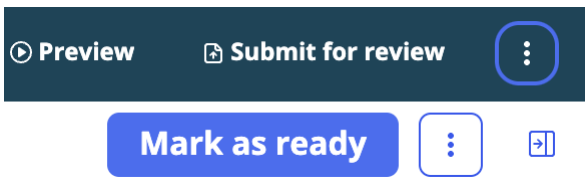
   Selecting a section

   For more information, see Layout options for forms in Creator Studio.

   c. Select **Save** in the Section details/Question details panel when you're done revising the form's layout.

8. Finally, publish the form when it's ready by selecting the **Mark as ready** button.
   Publishing forms makes them available as catalog items in the production instance for published apps.



Result

Now that the form is available as a catalog item, we can use it to create an automated playbook.

Quick start: Add a playbook

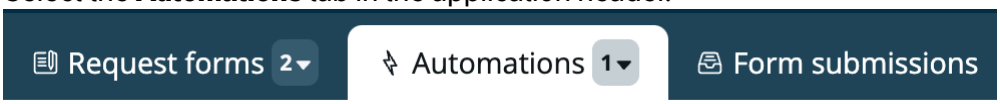
Add a playbook to create automation for your app, for example, to automatically assign a record to a manager for approval.

Before you begin

Your system administrator must add you to the Creator Studio Users group.

Procedure

1. Select the **Automations** tab in the application header.

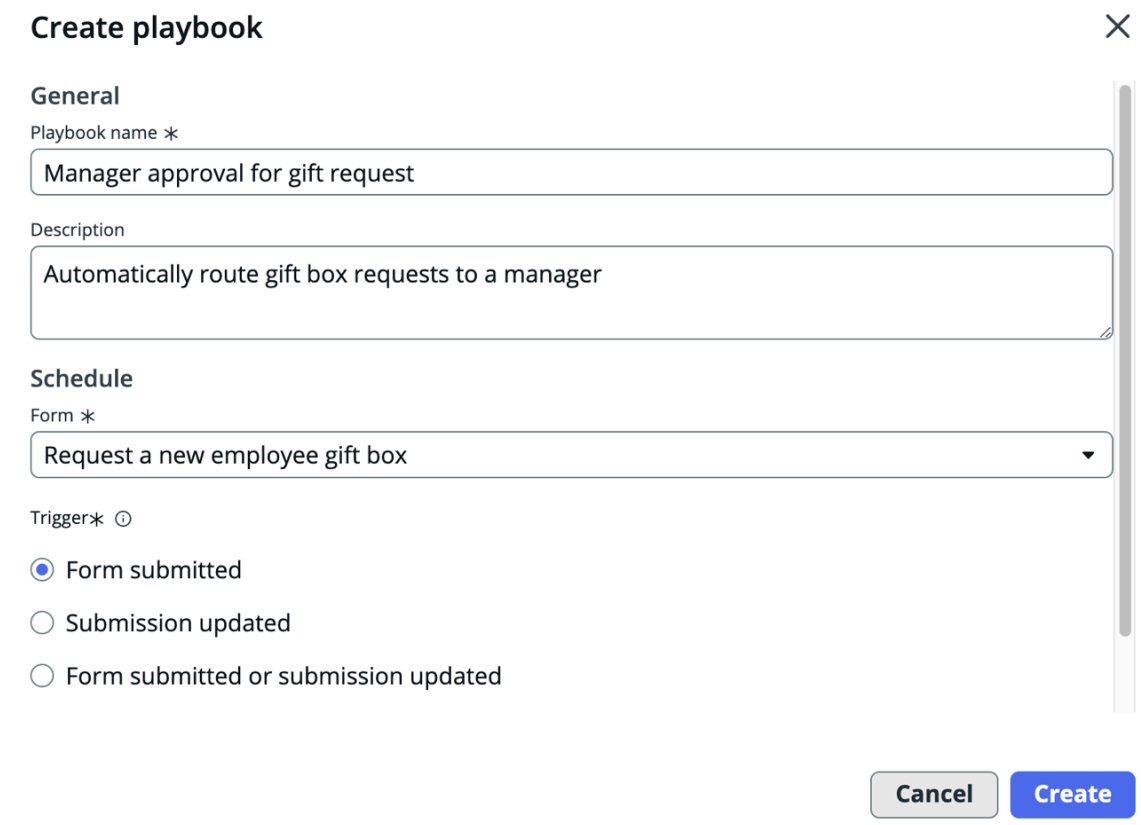


2. Select the **Create a playbook** button.
3. Specify the playbook's General attributes in the Create playbook modal.

General playbook definition fields

Field	Description
Playbook name	Descriptive name for the playbook you're creating.
Description	Brief explanation of what the playbook does, for example, the end goal for the record type.

Create a playbook



4. Specify the Schedule for the playbook.

   a. Select the **Form** whose catalog item generates the record type that you want the playbook to run on.
   The table for the form you choose is always the app's task table, which is specified when the app is created.

   b. Select the type of **Trigger** that initiates the playbook.

   Playbook trigger options

Trigger	Description
Form submitted	Start running the playbook when a user submits the form you chose.
Form updated	Start running the playbook when a user updates the form you chose.
Form submitted or updated	Start running the playbook when a user submits or updates the form you chose.

   Note:
   You can't change an playbook's trigger type after you finish creating the playbook. Instead, create a new playbook with a different trigger.

   However, you can edit the trigger condition, such as making the playbook run conditionally based on a specific answer to a question. For more information, see Edit the trigger for a playbook in Creator Studio.

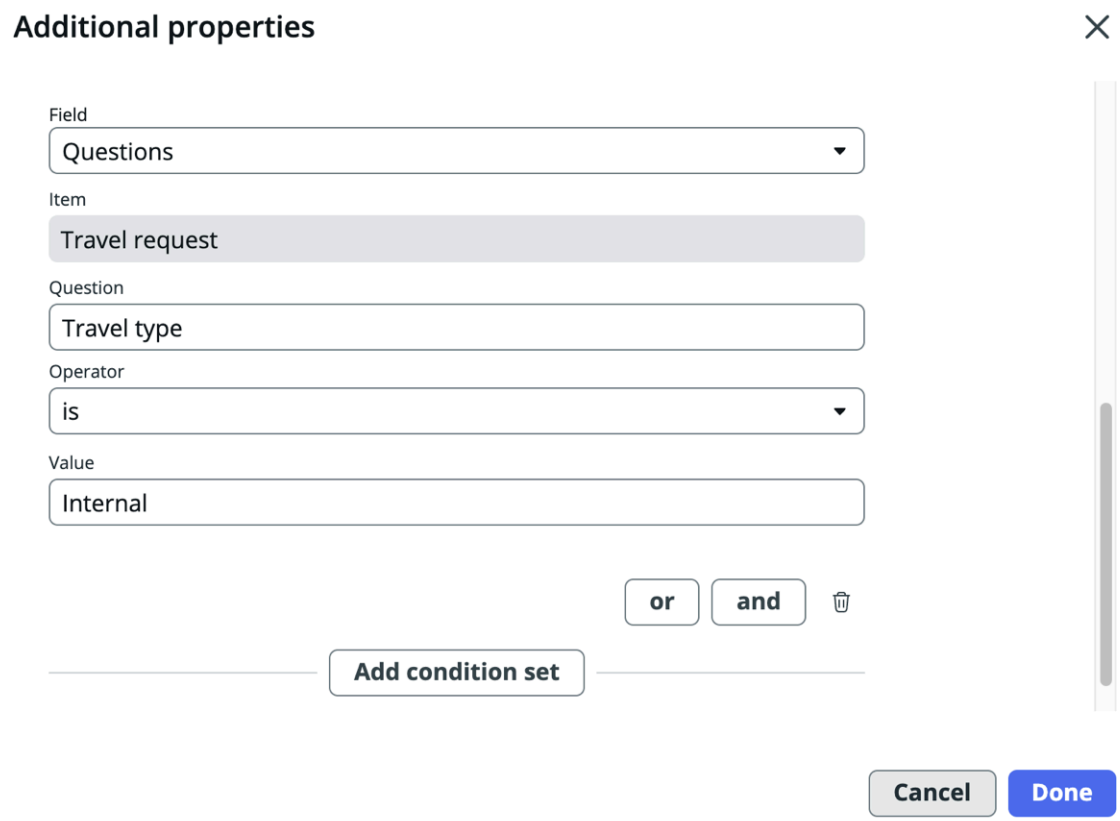
   c. If you chose a trigger that includes a form being updated, specify how often that app should **Run your playbook**.
   The options are:

   - **Once**
   - **For each unique change**
   - **Only if not currently running**
   - **For every update**

   d. Specify the conditions that must be met for the playbook to begin running by selecting **Add condition set**.

- If you want to trigger the playbook based on the value of a column in a table, select the **Field** that you want to be the trigger, as well as its condition **Operator** and the specific trigger **Value**. For example, when a **Start date** is **after** the **Date** needed.
- If you want to trigger the playbook based on the response from a form, select **Questions** as the trigger **Field**. Then select the question you want in the **Question** field, the condition **Operator** and the answer's **Value**.

Question answer as trigger for an automation



Add as many conditions as you need. For more information, see Create a condition statement using the condition builder.

5. Select **Save** before moving on to add an activity.

Result

Next, we'll add an activity to the playbook to define what the automation does.

Quick start: Add an activity and activate a playbook

Add an activity to your playbook to define what the automation does.

Before you begin

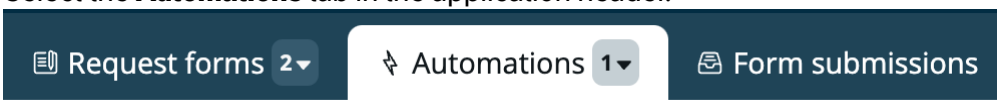
Your system administrator must add you to the Creator Studio Users group.

About this task

If you want to include an if/then statement to define circumstances for the activity, add a decision. See Add a decision to an app's playbook in Creator Studio for details.

Procedure

1. Select the **Automations** tab in the application header.



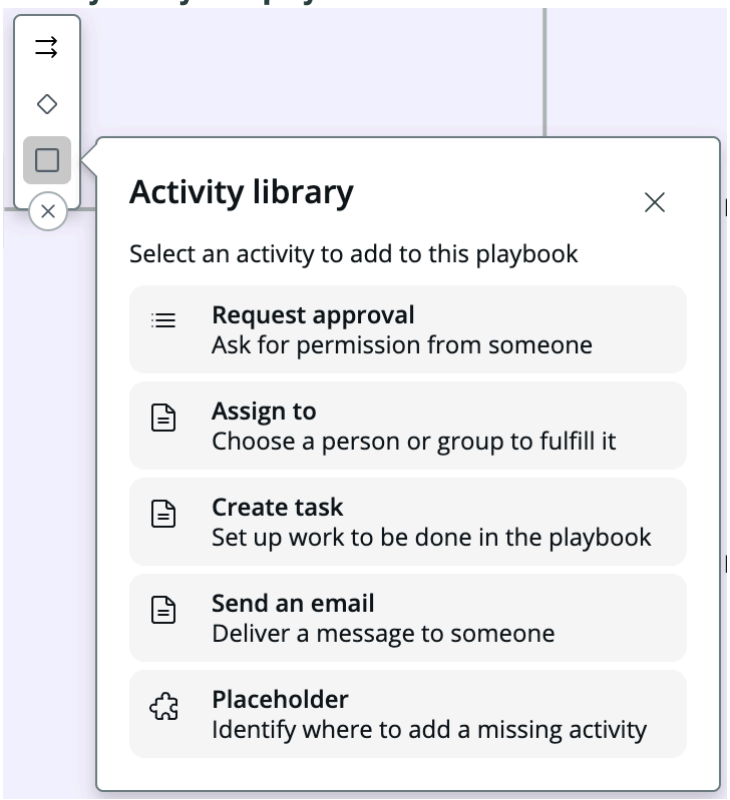
2. Select the add icon on the connector where you want to add an activity and choose the square **Add an activity** icon ( ) in the menu that pops up.
3. Choose the type of activity that you want from the Activity library pop-up.

Note:
In addition to the following standard activities, you may see some custom activities that your admin created.

Types of activities

Activity type	Description
Request approval	Ask someone for permission to accomplish a task.
Assign to	Choose a person who should fulfill the task.
Create task	Specify a process that must be done as part of the playbook.
Send an email	Send an email to one or more people. You can specify images and enrich text for the email that gets sent automatically.
Placeholder	Set an undefined activity to be specified later, or a more advanced activity such as an email notification, when an activity is completed. Placeholder activities don't have any logic assigned to them yet, and must be edited in Workflow Studio. Or, you can swap them out later for another type of activity in Creator Studio.

Activity library for a playbook

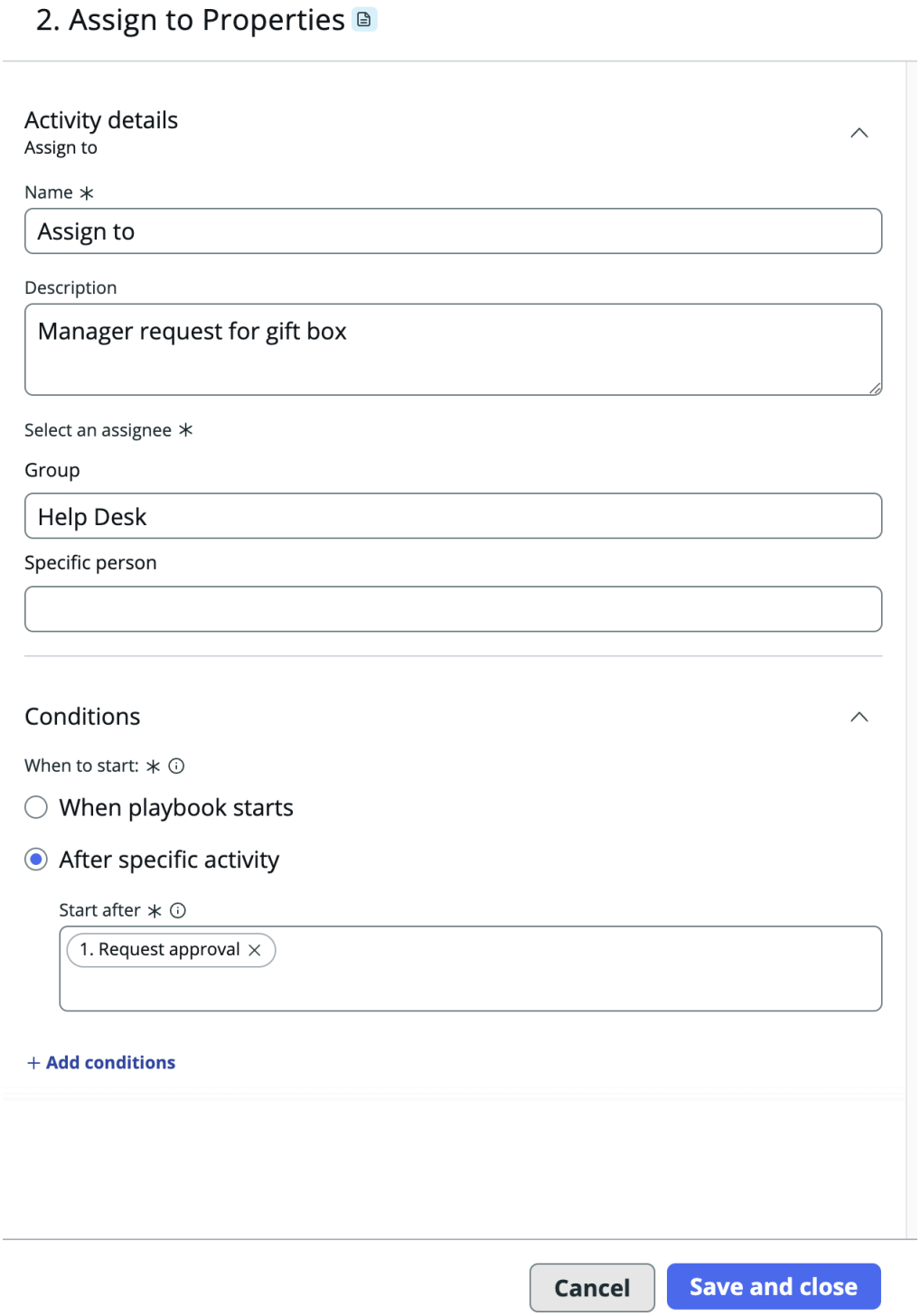


4. Enter the basic details for the activity.

General activity details

Field	Description
Name	Unique, user-facing name for your activity, which appears to agents and fulfillers while the playbook is running.
Description	Optional details about what the activity accomplishes.

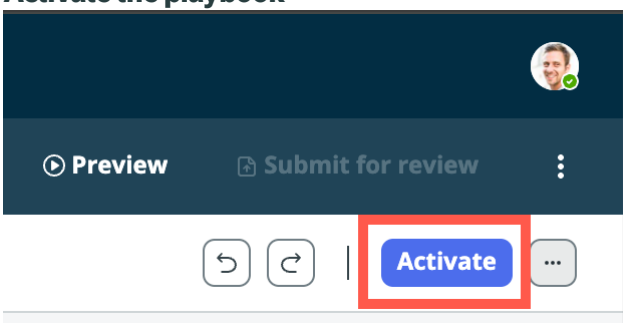
Activity details panel



5. Complete the details for the activity, using Add activities to an app's playbook in Creator Studio for information on how to finish creating the activity.
6. Select the **Save and close** button to finish setting up your activity.
7. Once the automation is done, activate it by selecting the **Activate** button.

Activating a playbook makes it available to run when its related form is created or updated on your non-production, development instance.

Activate the playbook



Result

The app is now ready to be deployed to production! Let's request deployment in the final step of this quick start guide.

Quick Start: Submit your app for deployment

Now that you've made an app, it's time to submit it for review so admins can approve and deploy it.

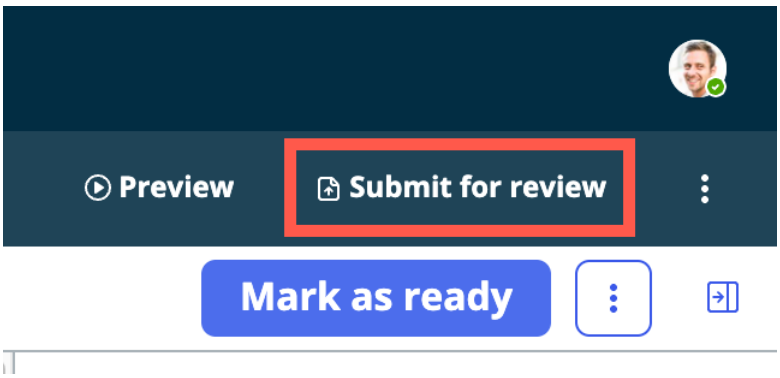
Before you begin

Your system administrator must add you to the Creator Studio Users group.

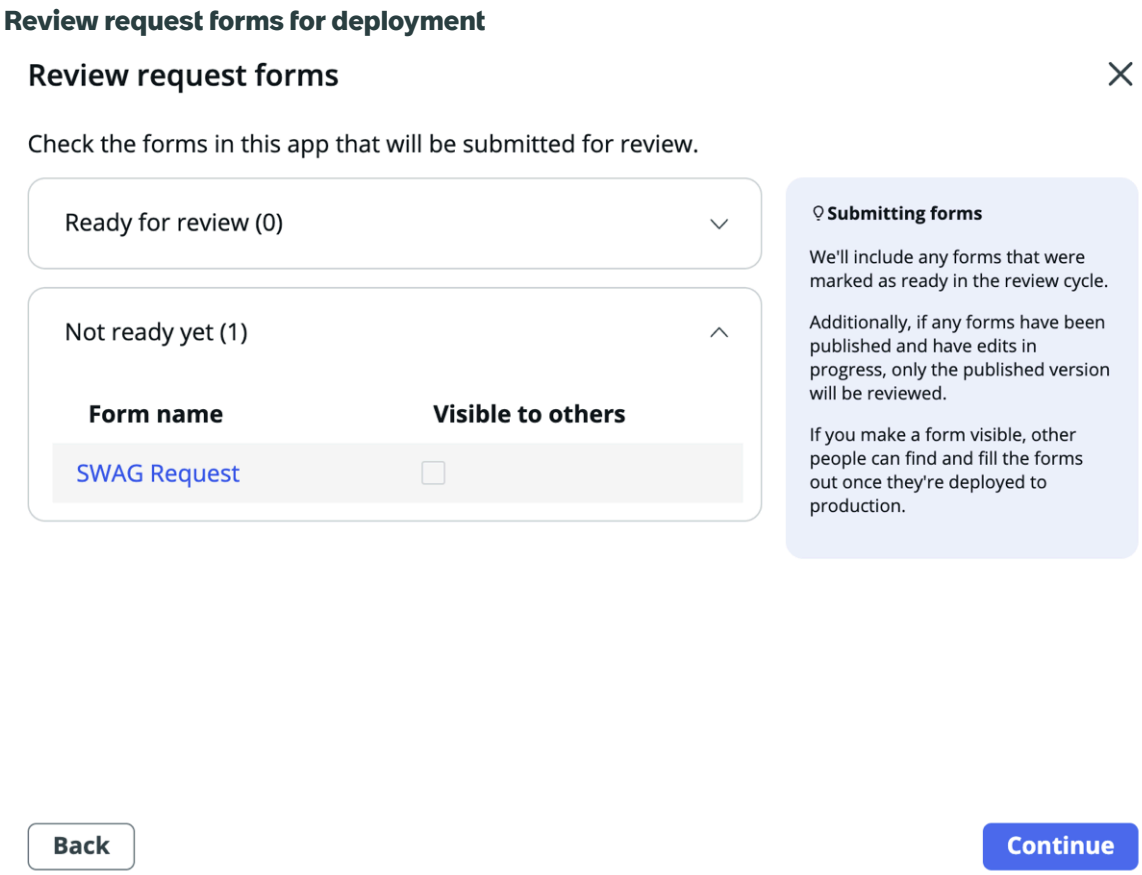
Procedure

1. In the application header, select **Submit for review**.

Submit for review



2. Select **Continue** on the Submit app for review modal.
3. Now you need to choose which published forms are visible to users in the catalog. In the Ready for review section of the Review request forms modal, select which of the app's published forms that you want to be available after the app is deployed selecting the **Visible to others** option.



4. Select **Continue** when you're happy with the forms being deployed to production.
5. Next, you must decide which of the app's activated playbooks will run on production after the app is deployed. In the Review playbooks modal, select the **Run on production** option for each playbook that you want to run on records that the app generates.

   Note:
   If you can't select a playbook, you need to go back to the **Automations** tab of Creator Studio and activate it. If you need a refresher on that, check out Activate a playbook in Creator Studio.

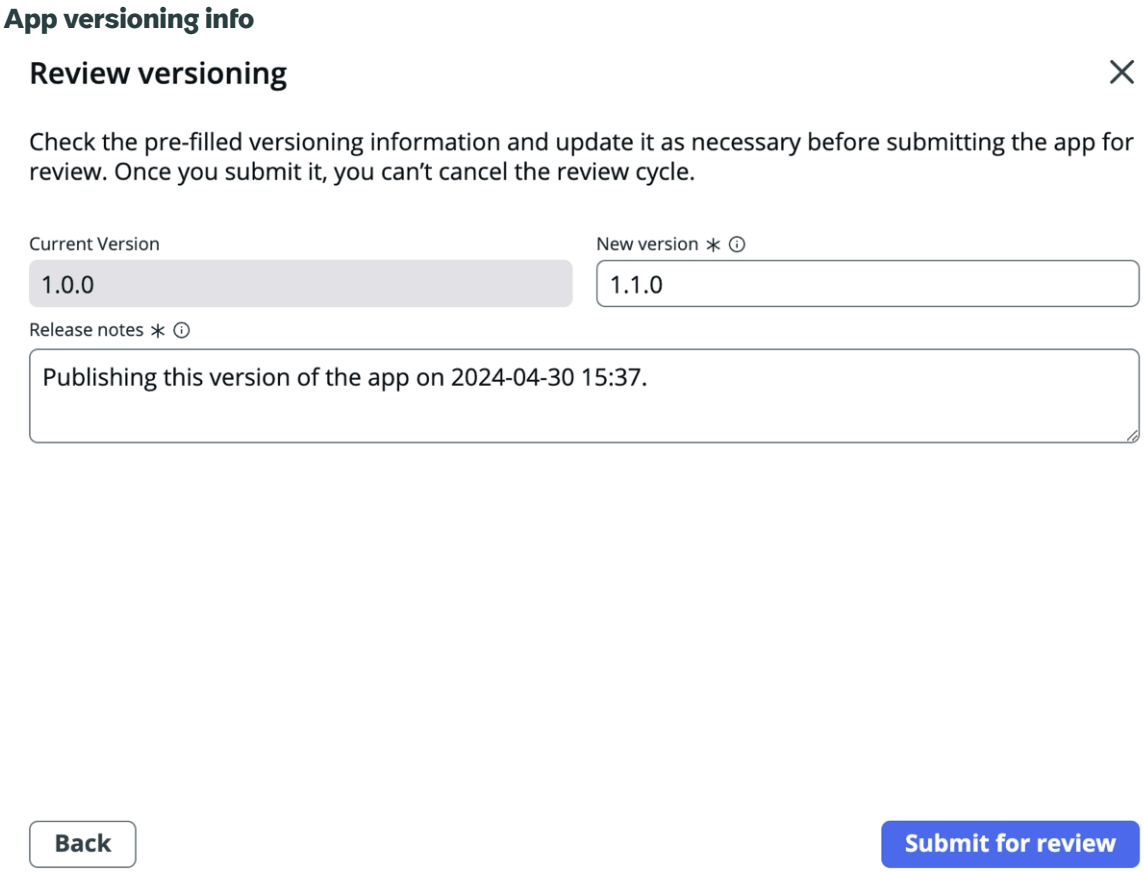
6. Select **Continue** when you're happy with the playbooks being deployed to run on production.
7. Finally, make sure that all the release details for the published app are correct.

Versioning options for deployment

Field	Description
New version	Version number of the app you're requesting for deployment. Creator Studio automatically generates an updated version number, but you can change it. Follow your organization's versioning guidelines, or use the x.y.z format, where x = major update, y = minor update, and z = patch.

Field	Description
Release notes	Details on what's changed in this new version of the app, or a general description of what the app does if this is its first version.

Read more about this step of requesting deployment in App versioning and release notes for Creator Studio apps.



8. Select **Submit for review** when everything is correct and ready for your admin to review and deploy.

Result

Congrats:
Hooray! You've created your app, customized the form, added automation, and submitted it for review. After your admin reviews and deploys it, people will use your brand-new app to make requests.

Get help with Creator Studio

If you need help with Creator Studio, you can watch a short video on how to contact your admin and check out some helpful resources.

Get help now from your Now Support administrator

Watch the following video to find out how to get help quickly.

https://player.vimeo.com/video/1056112088?h=1a5cd3b245&badge=0&autopause=0&player_id=0&app_id=58479

Helpful resources

Some ServiceNow resources that can provide you with helpful information are:

Video

Watch Build apps fast with Creator Studio.

ServiceNow Community

Creator Studio Community

FAQs

Creator Studio FAQs

Support

Migrating to Creator Studio from Service Creator

Creator Studio is intended to replace the legacy Service Creator offering, which will be retired in the Australia release.

Note:
You can find content for old versions of Service Creator in previous release documentation.

Why you should use Creator Studio instead of Service Creator

Creator Studio has a more modern interface than Service Creator. Creator Studio uses Record Producers directly, implements workflows with Playbooks, and allows fulfillment via Workspaces. It also integrates with deployment pipelines to promote applications created with Creator Studio to production environments.

Process for trying out Creator Studio

If you're a Service Creator user ready to check out Creator Studio, you must do the following:

- Confirm that your instance is on a minimum version of Xanadu patch 3.
- Purchase an App Engine Enterprise license.
- Download Creator Studio from the ServiceNow Store. For more information, see Installing Creator Studio from the ServiceNow Store.
- Activate Creator Studio.

Playbooks vs. workflows

Unlike the workflows that Service Creator used, Creator Studio uses playbooks to automate processes in the apps you build.

Playbooks are a set of activities that occur in order based on a trigger. Each activity is more like a step in a chain of steps. You can add multiple playbooks to an app if you need to.

Playbook activities in Creator Studio replace the process of defining service configurations in Service Creator.

In Service Creator, you had to first create a category for services, and then create the service and its table. Creator Studio automatically creates the table for you when you create an app, and streamlines the category selection process.

For more information, see Working with automation in Creator Studio.

Creating your first app with Creator Studio

Creator Studio helps you create your app by dividing it into smaller parts. Each does something special, and you work on them sequentially.

Summary:
After reading this section, you'll understand:

- The general workflow for building an app
- What goes into the idea for an app
- Who to ask for help

Let's give you a very high-level overview of what you will do in each part.

1. **Let us help you get started.** Start by selecting a catalog template that closely matches the app you want to create. For instance, if you're creating an app for IT issues, there could be a catalog template with pre-designed fields for specifying IT-related problems. While the template might not cover everything you need, it gives you a solid starting point. Check out how you select a catalog template to create your app's first form in Create the foundation of an app in Creator Studio.
2. **Or ask to work on someone else's app!** If you see an app that you want to work on but don't have access to, you can request to get access to it. Find out how in Ask to work on an app in Creator Studio.
3. **Ask for the info you need.** Customize the default catalog template to ask for the specific information your fulfiller needs. How do you know what customizations to make? Ask yourself what info the fulfiller needs to fulfill a request. We'll show you how to add, subtract, and modify the form fields in Customize your form for an app in Creator Studio.
4. **Make your forms available.** Once your form is ready, publish it to make it accessible to requesters. The process is straightforward, and we'll walk you through it in Publish a form for your app in Creator Studio.
5. **Specify the automatic actions that your forms trigger.** You'll want your app to take some action when a requester submits or revises a form. For instance, a new request might trigger approval from a manager. Another action could be emailing the requester to confirm their request submission and provide them with a tracking number. Discover more automation possibilities in Working with automation in Creator Studio.
6. **Customize the workspace fulfillers use.** Creator Studio automatically creates a form submission workspace category where fulfillers work on requests that are ready to use, out of the box. However, you can tweak it to suit your needs, find out how in Working with form submission workspaces in Creator Studio.
7. **Get your app deployed to production.** Once your app is spiffed up, tested, and ready to go, ask an admin to move it to a production environment for you so real users can begin using it to make requests. Find out how in Request deployment for your app from Creator Studio to production.

Use case:
Sam's job at the Example Company is to answer employee requests for office equipment. They've done this for years by receiving emails and keeping spreadsheets of requests, requesters, equipment, prices, dates, and fulfillments. Their company is growing, and the number of requests is now overwhelming. "I'm spending all my time reviewing requests instead of doing all the other parts of my job." They'd like to create an app that enables people to request office equipment, automatically sends an email confirmation of the request, stores the request, and enables them to fulfill or reject the request with the click of a button. They've never written a line of code in their life, but with Creator Studio, they can create the exact application they want.

App compatibility with Creator Studio

You can open apps built in Creator Studio in other ServiceNow products, but only apps built in Creator Studio can be opened in Creator Studio.

Opening Creator Studio apps in other products

After you build an app in Creator Studio, you can open it in several other products. For example, you could build an app in Creator Studio and then open it in ServiceNow Studio to add more complex flows in the automation.

If you add complex functionality to an app in another builder and then re-open the app in Creator Studio, you'll be warned in the appropriate context. For example, if you add complexity to a playbook that's not supported in Creator Studio, then Creator Studio will display a minimal version of the playbook or activity and instruct you to open it in Workflow Studio.

For more information, see the following products:

- ServiceNow Studio
- App Engine Studio
- Catalog Builder

Opening apps built in other products in Creator Studio

You can't open apps created outside of Creator Studio in Creator Studio. For example, you can't open apps built in ServiceNow Studio in Creator Studio.

However, if you create an app in Creator Studio and then add a form to it from outside of Creator Studio (for example, from Service Catalog), then Creator Studio tries to show the outside form.

Creator Studio apps and tables

Creator Studio uses catalog templates to streamline the app creation process. Apps use the Task table, generating a new row for each submitted request.

Summary:
After reading this section, you'll understand:

- The parts that go into building an app
- Where requests from the apps go
- How tables are used to store requests

Parts of building an app

Apps contain one or all of the following parts.

Parts of an app

Functionality	Description
Form	When filled out, forms create a record with an associated task or workflow on the ServiceNow AI Platform.
Automation	Automation contains playbooks, or automated processes with limited actions that enable users to update records and complete tasks across multiple activities.

Parts of an app (continued)

Functionality	Description
Form submissions	Workspace configuration that shows requests for the app built in Creator Studio, with standard and customizable filtered lists. Selecting a record in the **Form submissions** tab displays the record, which you can interact with in Creator Studio, enabling you to see how it will look.

Your admin can create catalog templates to guide the process of building forms. For more information, see Working with forms in Creator Studio.

Where requests from the apps go

Every app in Creator Studio automatically creates a task table where all opened records go when people use the app. The tables that Creator Studio apps generate extend the Request Task table that comes with Creator Studio, which extends the Task table.

Extending a table means the new table inherits the parent (extended) table's columns, as well as its business logic. For more information on Task tables, see Working with the Task table.

For sys admin eyes only: Every app built in Creator Studio adds a record in the Request App Config table. The name of the table follows the format of scope_request, for example, x_snc_02_03_request.

Key term:

Record

A record is what the ServiceNow AI Platform generates for each request submitted through your app. Each record corresponds to a row in the app's Request Task table.

Publishing, activation, and deployment workflow for forms, automation, and apps

When you build an app in Creator Studio, you must create forms and automation. You can also customize the form submission workspace before everything is deployed to production.

As you develop apps in Creator Studio, you should move sequentially through the following sections. For example, you should mark all of an app's forms as ready in the **Forms** section of Creator Studio before you begin to work on the app's **Automations** section.

1. When you're done building forms, publish them by marking them as ready.

   Published forms are then available in the catalog on the non-production instance that you're using for development.

2. When automated playbooks (which are based on the forms you already created) are ready, activate them.

   Published playbooks can be triggered when a form is created or updated on the non-production instance you're using for development.

3. The app's form submission workspace is already available on the non-production instance that you're using for development for you to test and customize. Additionally, you can interact with

test records by selecting the record in the **Form submissions** tab, which opens the test record in a new tab within Creator Studio.

4. When everything looks good and you're ready to deploy the app to production, you submit it for review.
5. After your admin deploys the app to production, the app is available to users whose role gives them access to it. On the production instance:
   - The app is available to users
   - The catalog contains all published forms
   - Activated playbooks are triggered on applicable records
   - Unactivated playbooks are deployed but must be activated by an admin before they can run
   - Users can access the workspace to review the app's form submissions
6. To update a published app, make the changes on your non-production instance and then request that your admin redeploys the updated app to production.

Configuring Creator Studio

Admins need to install Creator Studio before it can be configured for users to start building apps.

Configuration overview

The following is a general overview of installing and configuring Creator Studio.

1. Decide on an instance strategy, for example, what are your development (non-production) and production instances. For more information, see Creator Studio development instance strategy.
2. Download Creator Studio from the ServiceNow Store and install it. For more information, see Installing Creator Studio from the ServiceNow Store.
3. Run Guided Setup to configure administration and collaboration. For more information, see Configure Creator Studio using Guided Setup.
4. Ensure all users who need access have the correct role. For more information, see Creator Studio roles and personas.

Using Guided Setup to implement Creator Studio

Guided Setup provides a sequence of tasks that help you configure Creator Studio on your ServiceNow instance. Currently, the Guided Setup helps you to set up the following:

- The admin group to define who can administer Creator Studio
- Collaboration descriptors to manage what app owners and editors can do
- Who has full and restricted access to creating apps in Creator Studio

To open Guided Setup for Creator Studio, navigate to **All** > **App Engine** > **Guided Setup – Shared**.

For more information, see Guided setup.

Catalog configuration requirement for Creator Studio

To ensure that forms appear correctly for users, the non-production and production instances must have the same Service Catalog and all of its categories.

Configuring Virtual Agent chatbot previews

If you want forms to appear in the Virtual Agent chatbot, you must install the necessary plugins. For more information, see Activate Virtual Agent.

Installing Creator Studio from the ServiceNow Store

Installing Creator Studio from the ServiceNow Store makes it available for people to build apps on your instance.

Your company must have an App Engine Enterprise license to install and configure Creator Studio.

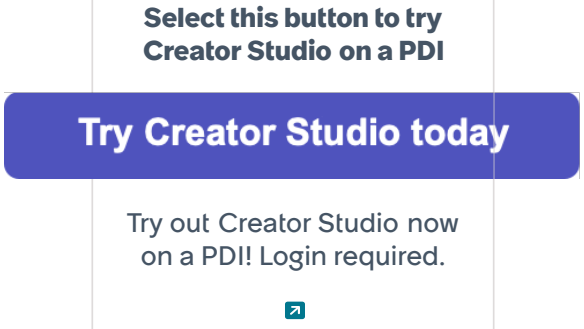
For instructions on installing Creator Studio and other products from the ServiceNow Store, see Install a ServiceNow Store application.

Note:
If you want forms to appear in the Virtual Agent chatbot, you must install the necessary plugins. For more information, see Activate Virtual Agent.

After Creator Studio is installed on an instance, admins have to configure it. The easiest way to do that is using Guided Setup, find out how in Configure Creator Studio using Guided Setup.

Give Creator Studio a try

Ready to give Creator Studio a try? You can test it out using your own Personal Development Instance (PDI), which requires you signing in to the Developer Site. Find out more on PDIs in the Personal developer instance guide.



Creator Studio development instance strategy

Make sure to install Creator Studio on all ServiceNow instances where users will be building applications, including the production instance.

Deciding on your instance strategy

You need to decide how you want to manage access to Creator Studio within your company. Consider the following options:

- **Open Access:** Allow everyone in your company to use Creator Studio to create apps.
- **Restricted Access:** Limit access to a specific group of users.
- **Request-Based Access:** Set up a form where users can apply for access. Admins will review these requests and decide whether to grant access.

Development and deploying to production instances

A non-production instance that is similarly configured to your production instance may be the best candidate for your test environment. You can then more accurately find issues that may arise if the application is deployed to production.

You should ensure that developers build apps in Creator Studio on a non-production instance, and then deploy apps that are ready and approved to production.

When you deploy an app, the records are referenced in the Store Apps [sys_store_app] table on the production instance. But when you're developing an app, the records are referenced in the System Applications [sys_app] table. So if you develop in production, you're developing using the [sys_app] table instead of [sys_store_app].

After you establish your instance strategy, you must also establish and automate your approval or review process. Creator Studio runs on your non-production environment, and admins then deploy apps to the production environment. For more information on the deployment process, see Deploying your Creator Studio app.

If your organization has multiple non-production environments, you must decide which non-production environment Creator Studio will run on. You must also determine which pipeline to use for promoting apps from a particular non-production instance to your test instance, and then finally to production where the app will be running live. For more information, see Pipelines and Deployments.

Catalog configuration requirement for Creator Studio

To ensure that forms appear correctly for users, the non-production and production instances must have the same Service Catalog and all of its categories.

Developer roles and testing apps on instances

If you have a Creator Studio role of sn_creatorstudio.user or sn_creatorstudio.restricted_user, you won't be able to test the apps you build on the non-production instance's Request App Workspace. You should be able to test the app on the non-production instance using Creator Studio's app previews. You will be able to test the apps as a fulfiller in the workspace on the app that's been deployed to production.

Use case:

Let's say that a user is in the Creator Studio Users group, so when that user builds an app, that user gets delegated development permissions for that app. That user can then publish a request form, and if there are no roles required for the form, that user can submit requests with the form.

However, that user won't be able to fulfill requests or access the Request App Workspace because that user won't have the x_acme_user_app.agent role, and that user can't give that role to themself. Administrators must assign additional roles as necessary.

Components installed with Creator Studio

When you activate the Creator Studio plugin, various components like tables and user roles are automatically installed.

Note:

The Application Files table lists the components that are installed with this application. For instructions on how to access this table, see Find components installed with an application.

What plugins are required to activate Creator Studio?

You must install the following plugins for Creator Studio:

- Creator Studio [sn_creatorstudio]
- Creator Studio - Global [com.glide.creator_studio.global]

Properties installed with Creator Studio

sn_creatorstudio.history_record_limit

Limits the last accessed history, for example in recent apps and sorting on the Creator Studio home page.

- Type: integer
- Default value: 1000
- Set this value to -1 to disable the limit
- Reduce this value from the default if the home page isn't loading fast enough

Roles installed with Creator Studio

The following table lists all roles installed with Creator Studio. For details on how the roles work with Creator Studio, see Creator Studio roles and personas.

Roles installed with Creator Studio

Role	Description	Contains role
sn_creatorstudio.user	• Provides access to Creator Studio • Users can create new apps	• catalog_builder_editor • sn_collab_request.basic_write • sn_udc.basic_read • sn_creatorstudio.basic_read • sn_collab_request.basic_read • app_template_runner • sn_g_app_creator.app_creator
sn_creatorstudio.restricted_user	• Provides access to Creator Studio • Users can't create new apps	• catalog_builder_editor • sn_creatorstudio.app_requestor • sn_collab_request.basic_write • sn_udc.basic_read • sn_creatorstudio.basic_read • sn_collab_request.basic_read • app_template_runner
sn_creatorstudio.basic_read	Primitive read access to Creator Studio resources	
sn_creatorstudio.basic_write	Primitive write access to Creator Studio resources	

Roles installed with Creator Studio (continued)

Role	Description	Contains role
sn_creatorstudio.admin_write	Admin write access to Creator Studio	• sn_creatorstudio.basic_read • sn_creatorstudio.basic_write
sn_creatorstudio.app_requestor	Primitive app requester access to Creator Studio resources	
sn_creatorstudio.basic_fulfiller	Gives some fulfillers the following: • Access to the Request App Workspace • Ability to edit some fields in the sn_creatorstudio_task table	canvas_user

Tables installed with Creator Studio

Tables installed

Table	Description
Request Subtask [sn_creatorstudio_child_task]	Extends the Task table.
New Application Admin Task [sn_creatorstudio_new_application_admin_task]	Extends the Task table.
New Application Task [sn_creatorstudio_new_application_task]	Extends the Task table.
Request App Config [sn_creatorstudio_request_app_config]	Extends the Application File table.
Request Task [sn_creatorstudio_task]	Table where all requests from apps made in Creator Studio are stored. **Note:** You must have the Request Task table installed on the production instance as well as the non-production instance.
Creator Studio Activities [sn_creatorstudio_activity]	Table where all standard and custom activities for Creator Studio automations are stored. **Note:** This table is readable by Creator Studio users and delegated developers.

Configure Creator Studio using Guided Setup

So you've installed Creator Studio on an instance. Now what? You must configure it before users can start building apps.

Before you begin

Creator Studio must be installed on the instance before you can configure it. Find out more about that in Installing Creator Studio from the ServiceNow Store.

To ensure that forms appear correctly for users, the non-production and production instances must have the same Service Catalog and all of its categories.

Role required: admin

About this task

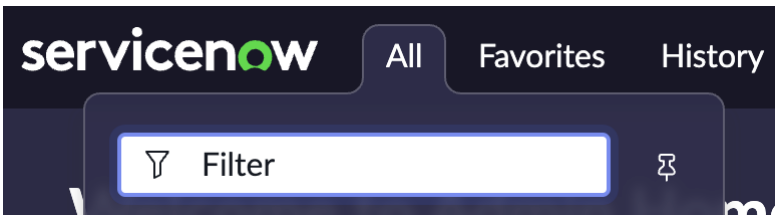
Guided Setup streamlines configuring Creator Studio, and keeps track of what you have completed, so you can stop and start again where you left off.

In this Guided Setup, you'll configure user access and collaboration settings to control who can do what in Creator Studio.

Procedure

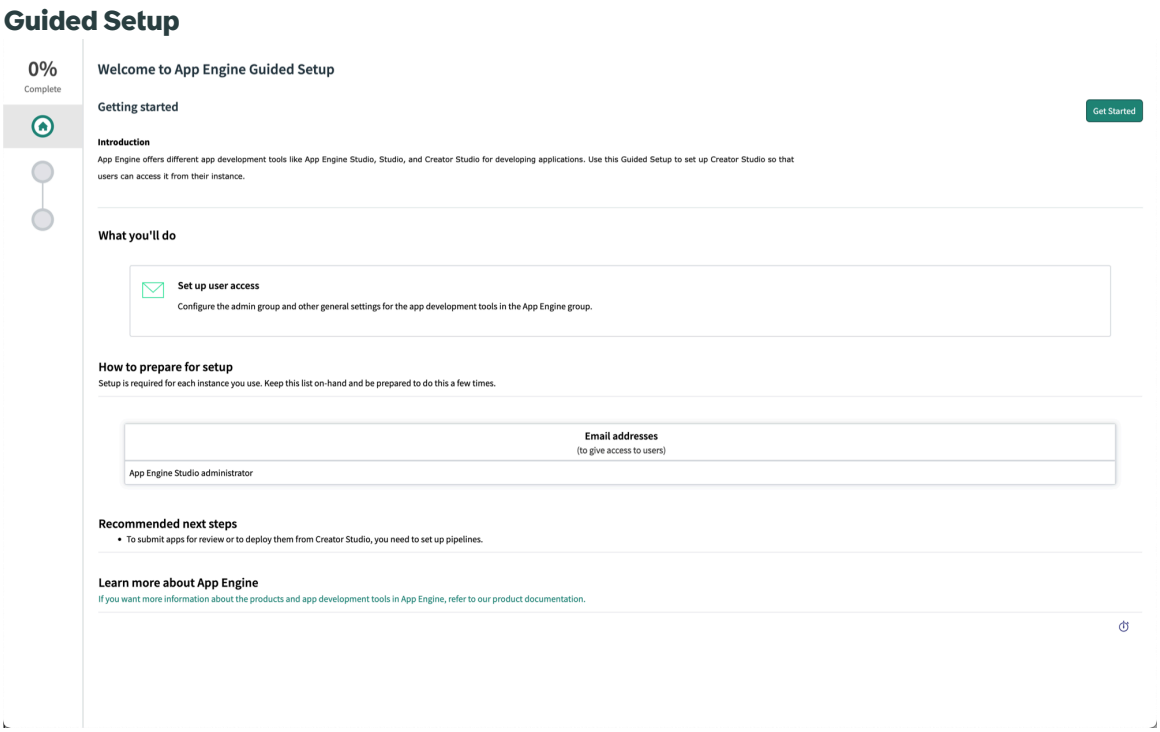
1. Let's open Guided Setup. On your instance, select the **All** tab.

   All tab

   This tab lets you see all the apps installed on your instance.

2. In the white text field, enter `App Engine`.
   The list of apps displayed underneath the text box now pertains only to App Engine.
3. In the list, select **Guided Setup – Shared**, which starts the configuration process.
   The Guided Setup app opens, where you make some basic configurations.



4. Start the setup process by selecting the **Get Started** button.
5. Configure user access by selecting the **Get Started** button in the Set up user access section.
6. Set up the admin group to enable ServiceNow administrators access to work on and configure Creator Studio.

   a. Select the **Configure** button for the Set up the admin group section.

   b. Make any changes to the App Engine Admins group.
   App Engine Admins approve tasks for app development, such as when restricted users need an app created for them.

   c. Update the record to save your changes.

   d. Select the **Mark as complete** button on the Guided Setup page to indicate that you're done with this step.

7. Set up collaboration descriptors to manage what app Owner and Editor collaboration types can do.
   By default, owners can do anything on an app, while editors are more restricted in what they can do. For more information, see Collaboration in Creator Studio.

   a. Select the **Configure** button for the Set up collaboration descriptors to manage your user's capabilities section.

   b. Customize the collaboration roles or add new ones.
   For more information, see Application collaboration.

c. Update the record to save your changes.

d. Select the **Mark as complete** button on the Guided Setup page to indicate that you're done with this step.

8. Select the **Welcome to App Engine Guided Setup** breadcrumb link to return to the main Guided Setup page.
9. Set up who has full access to Creator Studio by selecting the **Get Started** button in the Set up access to Creator Studio section.
   In this section, you decide which users get full access to work on app in Creator Studio, and who has more limited capabilities. For example, restricted users must request that an admin create a new app for them. For more information, see Creator Studio roles and personas.
10. Set up full access for users by selecting the **Configure** button for the Grant full Creator Studio access to users section.

    a. Add users to the Creator Studio Users Group.
    People in the Creator Studio Users Group have full access to working with apps, and can create apps without needing help from an admin.

    b. Update the record to save your changes.

    c. Select the **Mark as complete** button on the Guided Setup page to indicate that you're done with this step.

11. Set up limited access for users by selecting the **Configure** button for the Grant Creator Studio Restricted Access section.

    a. Add users to the Creator Studio Restricted Users Group.
    People in the Creator Studio Restricted Users Group have limited access to working with apps, and must request that admins create apps for them.

    b. Update the record to save your changes.

    c. Select the **Mark as complete** button on the Guided Setup page to indicate that you're done with this step.

What to do next

After you finish configuring Creator Studio, you can configure Pipelines and Deployments to enable apps built in Creator Studio to be deployed to production. For information on how, see Configure Pipelines and Deployments.

Configuring Pipelines and Deployments to deploy apps built in Creator Studio

You can install Pipelines and Deployments and configure a controller instance to deploy apps built in Creator Studio to production instances.

For details on how to set up Pipelines and Deployments, see Configure Pipelines and Deployments.

Creator Studio roles and personas

Roles control what everyone you work with can do in Creator Studio. Administrators assign roles to give team members permission to configure or use Creator Studio.

The two roles for Creator Studio are used to restrict access from creating new apps, which helps make sure your instance isn't overfilled with redundant, unplanned, or unused apps.

Personas that use Creator Studio

Personas aren't explicitly part of Creator Studio, but administrators assign roles to give team members permission to configure or use Creator Studio.

Low-code/citizen developer

Low-code/citizen developers are tech savvy and interested in creating apps. Though they might not have formal coding or app development training, citizen developers can submit ideas for new apps and, if approved, build them using Creator Studio.

Low-code/citizen developers have either the sn_creatorstudio.user or sn_creatorstudio.restricted_user role.

App Engine admin

App Engine admins manage all processes related to app development in Creator Studio. They review new app ideas, handle app deployment, and manage collaborators, usually in the App Engine Management Center.

App Engine admins have the app_engine_admin role and must be in the app_engine_admin group.

Security admin

The security admin creates and modifies roles and access control lists for apps. This role is set on the platform level, and it is required for making updates to roles in Creator Studio.

System administrator

The system administrator has access to all system features, functions, and data, regardless of security constraints. Grant this privilege carefully. If you have sensitive information, such as HR records, that you must protect, create a custom admin role for that area and train a person who is authorized to see those records to act as the administrator.

Roles and what they can do in Creator Studio

In addition to the roles in the following table, users with the admin and delegated_developer roles can also access Creator Studio.

For complete details on which roles each role contains, see Components installed with Creator Studio.

Creator Studio roles

Role	Name	Description
Creator Studio User	sn_creatorstudio.user	• Users can create apps in Creator Studio. • The user is automatically delegated as the app owner. For more information, see

Creator Studio roles (continued)

Role	Name	Description
		Delegated development and deployment. • Contains sn_g_app_creator.app_creator. **Note:** This role gets assigned the delegated_developer role when they create or get access to an app.
Creator Studio Restricted User	sn_creatorstudio.restricted_user	• Users can't create apps in Creator Studio. • Users can request apps to be created for them, and to work on an app. • Users can work on apps that they've been designated as developers for. • When assigned to work on an app, this user gets the delegated_developer role for that app.
App Engine Admin	app_engine_admin	• Approve requests from restricted users to create an app. • Approve collaboration requests. • Contains sn_creator_studio.admin_write and sn_creator_studio.basic_write to enable admins to see the apps they need to approve.

Note:

To ensure that users can use the Collaboration Approval Workflow regardless of instance versions, admins must assign the catalog _builder_editor role to Creator Studio user groups.

User groups and what they can do in Creator Studio

Groups are a standard functionality that help you quickly control people's access to Creator Studio by adding them to a group.

Creator studio user groups

Group	Description
Creator Studio Users	• Users are automatically approved to create apps in Creator Studio. • Contains sn_creatorstudio.user.
Creator Studio Restricted Users	• Users in this group need to request applications be created in Creator Studio on their behalf. • Contains sn_creatorstudio.restricted_user.

Developer roles and testing apps on instances

If you have a Creator Studio role of sn_creatorstudio.user or sn_creatorstudio.restricted_user, you won't be able to test the apps you build on the non-production instance's Request App Workspace. You should be able to test the app on the non-production instance using Creator Studio's app previews. You will be able to test the apps as a fulfiller in the workspace on the app that's been deployed to production.

Use case:

Let's say that a user is in the Creator Studio Users group, so when that user builds an app, that user gets delegated development permissions for that app. That user can then publish a request form, and if there are no roles required for the form, that user can submit requests with the form.

However, that user won't be able to fulfill requests or access the Request App Workspace because that user won't have the x_acme_user_app.agent role, and that user can't give that role to themself. Administrators must assign additional roles as necessary.

Collaboration roles and instances on different versions

As admins implement Creator Studio, they may have it installed on a non-production instance while their production instance is on a previous version of the ServiceNow AI Platform that doesn't have Creator Studio. This mis-match of instance versions affects the Collaboration Approval Workflow, which specifies the non-production instance as the source and the production instance as the controller. If the controller doesn't have the version of the collaboration plugin that supports Creator Studio, collaboration is unsupported.

To ensure that users can use the Collaboration Approval Workflow regardless of instance versions, admins must assign the catalog _builder_editor role to Creator Studio user groups.

Roles and app development collaboration

Roles define user access to Creator Studio. Permission to work on individual apps is controlled on an app-by-app basis. That is, you must manage the collaborators for each app by inviting other citizen developers to work on the app with you, or request to join someone else's app. For more information, see Collaboration in Creator Studio.

Creator Studio and domain separation

Domain separation is not supported in Creator Studio. This means that you can't separate data, processes, and administrative tasks into distinct groups, called domains, within the ServiceNow AI Platform.

No support for domain separation

Though there may be a domain specified on data tables that Creator Studio uses, Creator Studio doesn't provide any special handling of data based on domains.

For more information on support levels, see Application support for domain separation.

Related topics

Domain separation for service providers

Administering Creator Studio

Admins have special privileges to approve and update tasks in Creator Studio that regular users can't. Admins can create apps for restricted users, manage collaboration requests, configure custom activities for playbooks, and deploy apps to production.

Approve Creator Studio requests to create an app

Respond to app creation requests made by users who can't create apps themselves, ensuring that your organization doesn't have too many or redundant apps.

Before you begin
Role required: app_engine_admin

About this task
People with the Creator Studio Restricted User role aren't allowed to create apps themselves, but they can work on them after you approve their creation in the Application Tasks table of Request App Administration.

If your organization has App Engine Management Center, you can approve requests there, too. For more information, see Manage app requests from Creator Studio.

Procedure

1. Navigate to **All** > **App Engine** > **Request App Administration** > **Application Tasks**.
   You can also select the link in any emails you receive notifying you of app creation requests.
2. Find the request you want to approve or decline.
   Filter on columns or use the search bar if you need help finding requests.
3. Select the **Number** of the request to review.
4. Select the **Approve** or **Reject** button to approve or decline the app creation request.

Result
The user who made the request receives an email notification with the approval result. If you approved, they can start building out their app in Creator Studio!

Administering collaboration in Creator Studio

In Creator Studio, as an administrator or App Engine admin, you have the power to manage how people work together on projects.

You can approve collaboration requests, enabling delegated developers to work together on an app.

Collaboration descriptors are like permission levels that determine what someone can do in an app. Admins can assign and customize collaboration descriptors.

The following collaboration descriptors are available for Creator Studio:

Collaboration descriptions

Permission	Description	What they can do in the app
Manage collaborators	You can control who has access to the app and what they can do.	• Change permissions for users and groups. • Remove people from working on the app.
Invite collaborators	You can bring in new people to work on the app.	• Invite users and groups to join the app. • Invited people become editors of the app, meaning they can make changes.

Approve a collaboration request

Admins and App Engine admin can review and approve (or reject) people's requests to join an app's development in Creator Studio.

Before you begin

If you have a controller instance configured, you must approve collaboration requests on your controller instance, not the non-production instance with Creator Studio. The controller instance is typically the production instance, but it can vary depending on how your Pipelines and Deployments application is configured. For more information, see Configure your controller instance.

Role required: app_engine_admin

About this task

Note:
Users with a delegated developer role are automatically approved for collaboration.

Procedure

1. Navigate to **All** > *App Engine* > **Collaboration** > **Collaboration Tasks**.
2. Select the collaboration request in the **Number** column of the collaboration table.
   You can search or filter on the **App name** or **Invitee name** columns to find the request faster.
3. Update the approval status in the **State** field.
   - **Closed – Approved**
   - **Closed – Rejected**
   - **Canceled**
   - **Closed – Failed**
4. Update the record to save your changes.

Manage Creator Studio collaboration permissions on the ServiceNow AI Platform

In Creator Studio, you can control how people work together on apps by setting up permissions called collaboration descriptors. These descriptors define what users can and cannot do.

Before you begin

Role required: app_engine_admin, and permission to Manage collaborators

About this task

Users with the Manage collaborators permission can adjust who can collaborate on the apps they own. For more information, check Manage collaborators for an app in Creator Studio.

Note:
The way that collaboration permissions appear may vary depending on whether your instance has App Engine Studio installed.

Procedure

1. Navigate to **All** > *App Engine* > **Collaboration** > **Descriptors**.
2. Edit the permissions for a collaboration descriptor, such as owner or editor.
   a. Find and select the **Name** of the descriptor you want to modify.
   b. Choose a **Development permission set**, such as **Service Catalog**, to update its permissions. For a list of permissions, see Collaboration permissions.
   c. Select **Save** and **Update**.
3. Assign the updated descriptor to users or groups.
   a. Select the descriptor **Name**, such as **Owner** or **Editor**.
   b. Select the applicable tab to assign either users or groups:
      - **App Collaboration Users** tab: For individual users
      - **App Collaboration Groups** tab: For groups of users
   c. Select the **New** button.
   d. On the form, fill in the fields.
      For more information, see the following topics:
      - Assign collaboration descriptors to users
      - Assign collaboration descriptors to groups
   e. Select **Save**.
4. **Optional:** If you need a new collaboration descriptor, create a custom one.
   a. Select the **New** button.
   b. On the form, fill in the fields.

App Collaboration Descriptor form

Field	Description
Name	Name for the descriptor.
Description	Description of what the descriptor can do. Examples are as follows: ▪ Owner ▪ Editor
Application	Scope of the collaboration descriptor. Currently, all collaboration descriptors must be created in the global scope.
Standard	Option for inviting other collaborators with this role.

c. Select **Submit**.

Result

You've now set up collaboration descriptors, defined permissions, and assigned roles to users and groups for specific apps. For complete details on using the Collaboration app, see Application collaboration.

Customized app collaboration permissions in Creator Studio

When you customize collaboration permissions, you can choose more granular actions and parts of the app that users can work with in Creator Studio.

Use these permissions when managing collaboration, either as an admin or app owner. For more information, see the following topics:

- Manage Creator Studio collaboration permissions on the ServiceNow AI Platform
- Manage collaborators for an app in Creator Studio

File types custom collaboration permissions

File type collaboration permissions

Permission	Description	Owner default setting	Editor default setting
All File Types	Grants access to work with all types of files, including additional file types not granted by the other options.	Yes	Yes
Integrations	Grants access to web service APIs, REST APIs, data sources, and Integration Hub - Import.	Yes	Yes
Reporting	Grants access to reports and scheduled reports.	Yes	Yes

File type collaboration permissions (continued)

Permission	Description	Owner default setting	Editor default setting
Mobile builders	Grants access to build mobile experiences, such as with Mobile App Builder.	Yes	Yes
UI Builder	Grants access to work with UI Builder to build out more complex interfaces.	Yes	Yes
Workflow	Grants access to Workflow Editor and Activity Creator.	Yes	Yes
Service Portal	Grants access to work with Service Portal editors and tools.	Yes	Yes
Workflow Studio	Grants access to the Workflow Studio design environment to create flows and actions.	Yes	Yes
Service Catalog	Grants access to work with catalog-related file types such as catalog items, record producers, and variables to add catalog items to apps.	Yes	Yes
Tables & forms	Grants access to model and layout-related file types such as table columns, form layout, and list layout.	Yes	Yes
Decision Tables	Grants access to work with decision tables to create decision logic based on multiple if-else rules.	Yes	Yes
Playbooks	Grants access to work with the Playbooks design environment to create processes. Editing activity subflows or actions requires the **Flow Designer** permission.	Yes	Yes
Notifications	Grants access to create automatic email notifications	Yes	Yes

File type collaboration permissions (continued)

Permission	Description	Owner default setting	Editor default setting
	and work with email templates.		

Security/Entitlement custom collaboration permissions

Manage access control grants access to security management files, such as Access Control Lists and roles.

The default setting for both owners and editors is selected.

Programming tools custom collaboration permissions

Allow scripting grants access to script fields, such as those in Business Rule, UI Action, and Client Script.

The default setting for both owners and editors is de-selected.

Application management custom collaboration permissions

Application management collaboration permissions

Permission	Description	Owner default setting	Editor default setting
Delete application	Grants access to delete the app.	Yes	No
Manage collaborators	Grants access to change the developers and their permissions to collaborate on an app.	Yes	No
Source control	Grants access to work with source control tool integrations.	No	No
Invite collaborators	Grants access to invite developers to collaborate on an app.	Yes	Yes

Deployment custom collaboration permissions

Deployment collaboration permissions

Permission	Description	Owner default setting	Editor default setting
Upgrade app	Grants access to upgrade applications.	No	No
Submit for deployment	Grants access to request deployment for an app.	Yes	No

Deployment collaboration permissions (continued)

Permission	Description	Owner default setting	Editor default setting
Publish app to repo	Grants access to publish the app to your repo.	No	No
Publish to app store	Grants access to publish the app to your app store.	No	No

Administering templates and forms for Creator Studio

As an admin, you can create custom templates and pre-configured questions to help streamline and guide users when they build apps in Creator Studio.

Check out an article on Creating catalog templates for Creator Studio.

Creating catalog templates for use in Creator Studio apps

As a ServiceNow admin, you can create customized catalog templates in Service Catalog to guide users through adding forms in Creator Studio.

For example, if your users will be building several apps for IT fulfilment, you may want to create a catalog template that contains only IT service requests.

When you add a catalog template to help users create forms in Creator Studio, you must use the following configurations:

- Create a record producer template.
- Leave the **Destination table** (which is called the **Record submission table in Creator Studio**) as undefined.
- Ensure that the Creator Studio Users group can access the template. The template can also be available to all users.

Additionally, you may want to use the catalog template to restrict the catalogs that users can choose when creating an app.

For more details on how to add catalog templates, see Creating or editing catalog item template.

Check out an article on Creating catalog templates for Creator Studio.

Creating question sets for use in Creator Studio forms

Admins can create pre-configured question sets that can be reused across multiple forms without being changed in Creator Studio.

Question sets enable admins to create curated, uneditable questions that users can include on forms without worrying about tweaking their content.

Additionally, you can create question sets with multiple questions if you want several reusable questions to always be grouped together.

See Create a variable set and add it to an item to find out how to create a question set in Service Catalog.

Administering activities in Creator Studio

As an admin, you can define custom activities, make them available in Creator Studio, and hide existing activities to help guide users when they add automation to apps.

You may want to create custom activities for complex or company-specific workflows. For example:

- The standard Request approval activity has a **Requester's manager** option, but you may need a **Department manager** instead.
- You may need an approval or flow with more complex decision logic than is easily built in the Creator Studio automation. You could use Workflow Studio to build that flow, and then create a custom action that calls it.
- You may want more notification types than the standard Email notification activity. You could create a flow that sends a notification to a MS Teams integration, and create an activity that calls that flow.

Steps to create a custom activity in Creator Studio

There are multiple steps to add custom activities for use in Creator Studio automations.

Adding a completely new activity

The steps to create a custom activity for a playbook are as follows:

1. **Create the flow, subflow, or action**: An admin or someone with the correct role must first create the flow, subflow, or action in Workflow Studio. For more information, see General guidelines for Workflow Studio flows, subflows, and actions.
2. **Define the activity**: An admin then defines the activity by adding it to the Activity Definition table and assigning the flow/subflow/action to a playbook activity. This step is also when the admin specifies which inputs can be made available as fields in the activity settings in Creator Studio playbooks. For more information, see Activity definitions and Create an activity definition.

   Note:
   If an activity definition already exists for the activity you want, you can skip this step.

3. **Make the activity available in Creator Studio**: An App Engine admin adds the activity to the Creator Studio Activities table. For more information, see Make a custom activity available for playbooks in Creator Studio.

Making existing activities and playbooks available in Creator Studio

Admins can make existing flows, subflows, and actions available in Creator Studio by defining the activity and then adding it to the Creator Studio Activities table (the preceding steps 2 and 3).

Admins can also make existing playbook activities available in Creator Studio by adding them to the Creator Studio Activities table (the preceding step 3).

Note:
Creator Studio admins can see the Activity Definitions [sn_pd_activity_definition] table by default, which also enables them to administer processes in Workflow Studio.

Make a custom activity available for playbooks in Creator Studio

You must add custom activities to the Creator Studio Activities table [sn_creatorstudio_activity] before users can add them to playbooks when building apps.

About this task

You may want to create custom activities for complex or company-specific workflows. For example:

- The standard Request approval activity has a **Requester's manager** option, but you may need a **Department manager** instead.
- You may need an approval or flow with more complex decision logic than is easily built in the Creator Studio automation. You could use Workflow Studio to build that flow, and then create a custom action that calls it.
- You may want more notification types than the standard Email notification activity. You could create a flow that sends a notification to a MS Teams integration, and create an activity that calls that flow.

Before you begin

The following steps must be completed before you can make a custom activity available in Creator Studio:

1. **Create the flow, subflow, or action**: An admin or someone with the correct role must first create the flow, subflow, or action in Workflow Studio. For more information, see General guidelines for Workflow Studio flows, subflows, and actions.
2. **Define the activity**: An admin then defines the activity by adding it to the Activity Definition table and assigning the flow/subflow/action to a playbook activity. This step is also when the admin specifies which inputs can be made available as fields in the activity settings in Creator Studio playbooks. For more information, see Activity definitions and Create an activity definition.

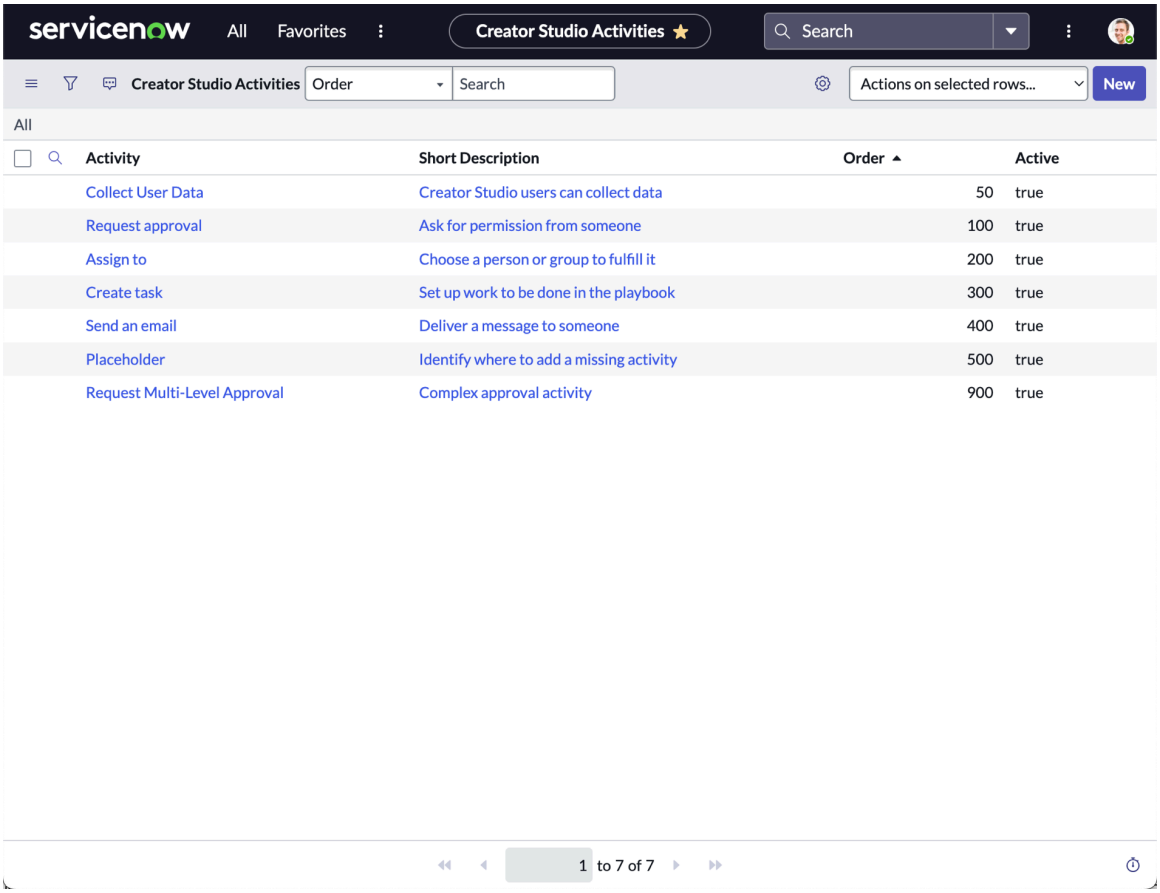
   Note:
   If an activity definition already exists for the activity you want, you can skip this step.

Role required: admin or app_engine_admin

Procedure

1. Navigate to **All** > *App Engine* > **Request App Administration** > *Creator Studio* **Activities**.
   All of the standard Creator Studio activities appear in the list, not just custom activities.
   The Creator Studio Activities table appears.

Creator Studio Activities table



2. Create the foundation of the new activity.

a. Select the **New** button.

b. On the record, fill in the fields.

Field	Description
Activity	Search for and select the type of activity that will be the foundation of your custom activity. For example, create a complex approval activity by selecting **Request Multi-Level Approval**. **Note:** Once you choose an activity, you can select the information icon ( ) to preview its Activity Definition record.
Order	Order that the activity appears in the Creator Studio activity picker list.

Field	Description
	For example, an activity with order number 400 appears before an activity with order number 500.
Short description	Brief description of the custom activity, which appears in the activity picker.

To see a list of all the available activities you can add:

- See Workflow Studio actions for all standard actions.
- See Workflow Studio flow logic for a list of all flows.
- See your list of all available subflows in Workflow Studio.

c. Confirm that the **Active** option is selected to make the activity appear in Creator Studio.

d. Select **Submit**.

3. Specify which fields appear in the playbook activity's properties panel.

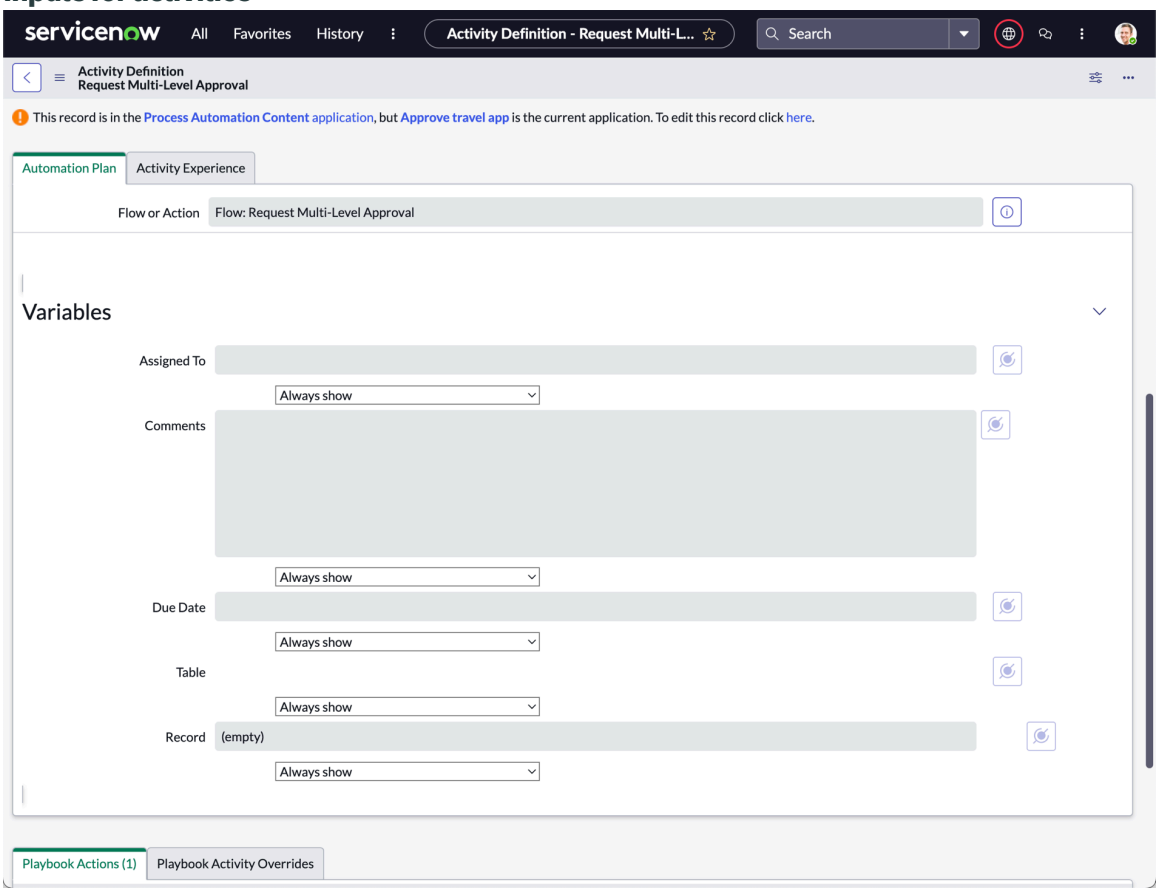
Note:
The fields that appear may have already been configured in the process of defining the activity. If so, follow these steps to verify that the available fields are correct.

a. Open the definition for the newly created activity by selecting it from the **Activity** column of the Creator Studio Activities table.

b. Confirm that the **Automation Plan** tab is selected.

c. Make the field appear for the activity's properties in a Creator Studio by selecting **Always Show**.
For example, set the **Assigned To** field to **Always Show** to users building a playbook in Creator Studio.

Inputs for activities



For more information, see Add dynamic inputs to an activity.

d. **Optional:** Hide a field from the properties panel by setting it to **Show as additional property for admins only.**

e. Repeat these steps for every field that should be visible on the properties panel.

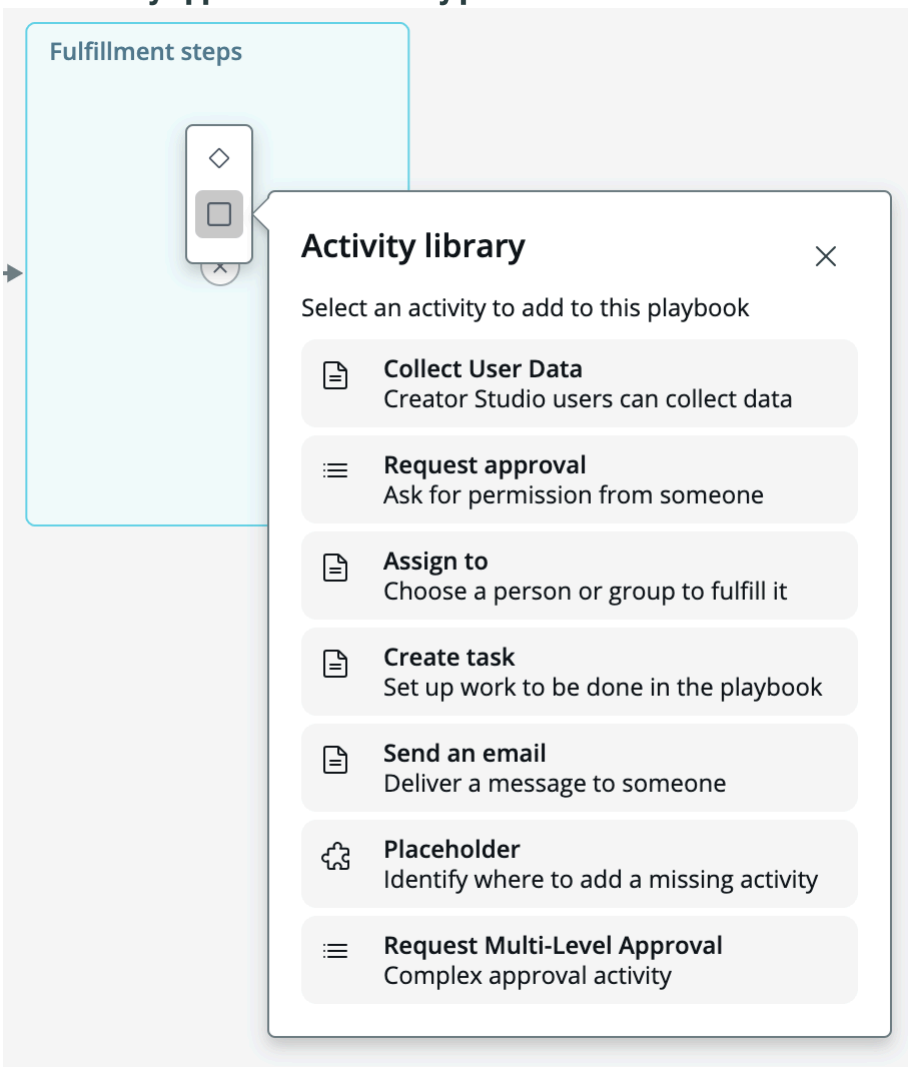
f. Save the record.

4. Return to the Creator Studio Activities table.
5. Review the Creator Studio Activity record by selecting is **Short Description** field.
6. Confirm that the **Active** option is selected, and that everything else is correct.
7. **Optional:** Save the record if you made any additional changes by selecting the **Update** button.

Result

The new activity appears in the activity picker in Creator Studio.

New activity appears in the activity picker



Reorder how activities appear in a playbook's activity picker

Change the order that activities appear in the activity picker in Creator Studio to present them in the order you want.

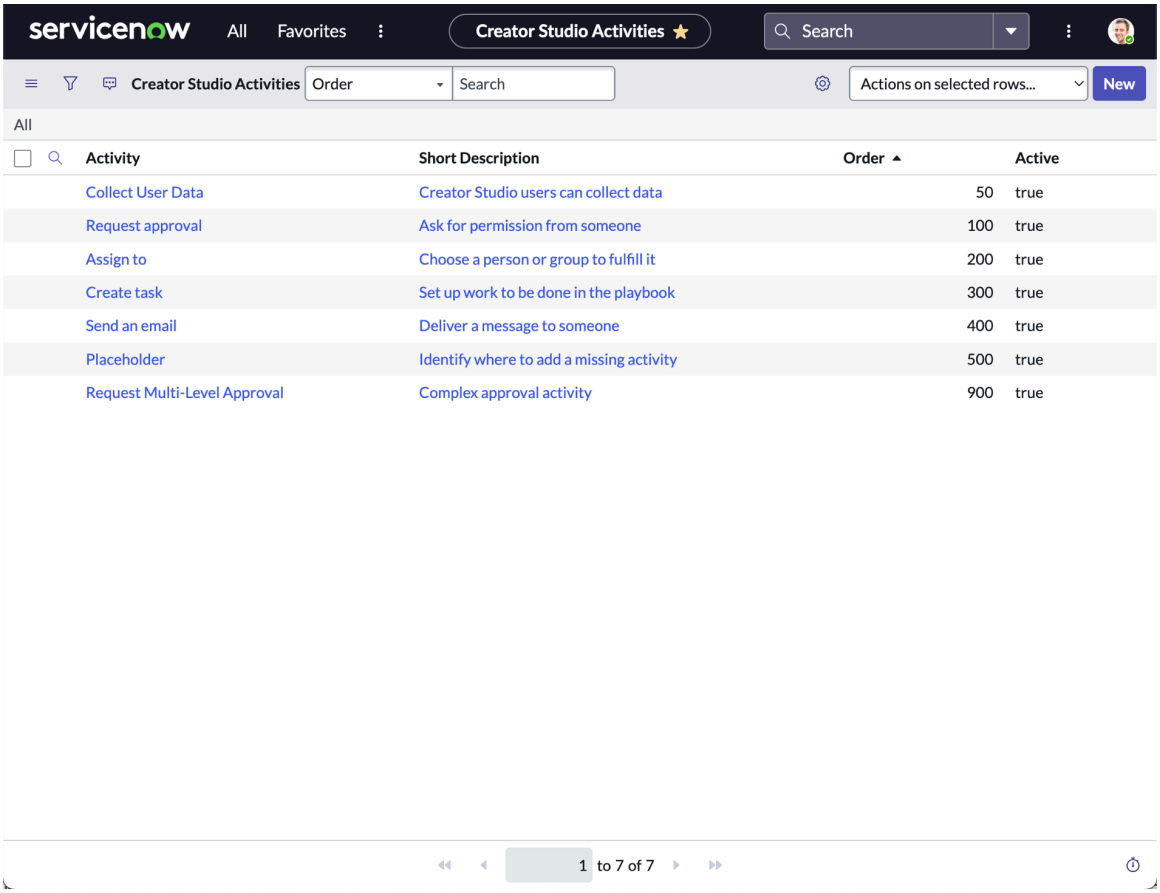
Before you begin

Role required: admin or app_engine_admin

Procedure

1. Navigate to **All** > *App Engine* > **Request App Administration** > *Creator Studio* **Activities**.
   All of the standard Creator Studio activities appear in the list, not just custom activities.
   The Creator Studio Activities table appears.

Creator Studio Activities table



2. Select into the **Order** column for the activity that you want to change the order of.
3. Enter a number to represent where in the activity picker list the activity should appear.

   For example, an activity with order number 400 appears before an activity with order number 500.

4. Select the save icon ( ) to complete the update.
5. Change the **Order** for every activity that you want.

What to do next

Alternatively, you can open each Creator Studio Activity record and update the **Order** one record at a time.

Make an activity unavailable in playbooks

Deactivate an activity to stop users from adding it to playbooks in Creator Studio.

Before you begin

For example, you could hide an base system activity, like sending an email, if it's not relevant to your workflow.

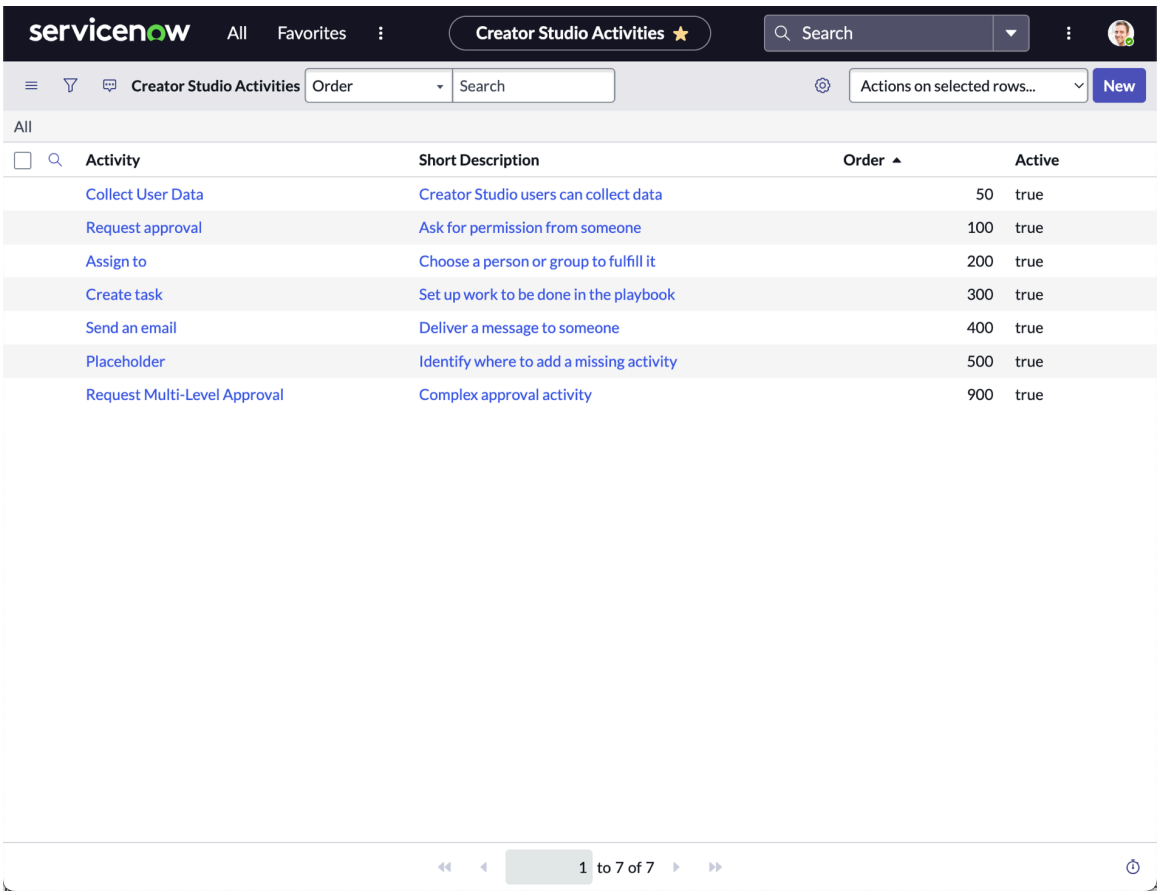
Note:
Hiding an activity doesn't change anything about the processes using that activity. Hiding an activity only makes it unavailable in Creator Studio.

Role required: admin or app_engine_admin

Procedure

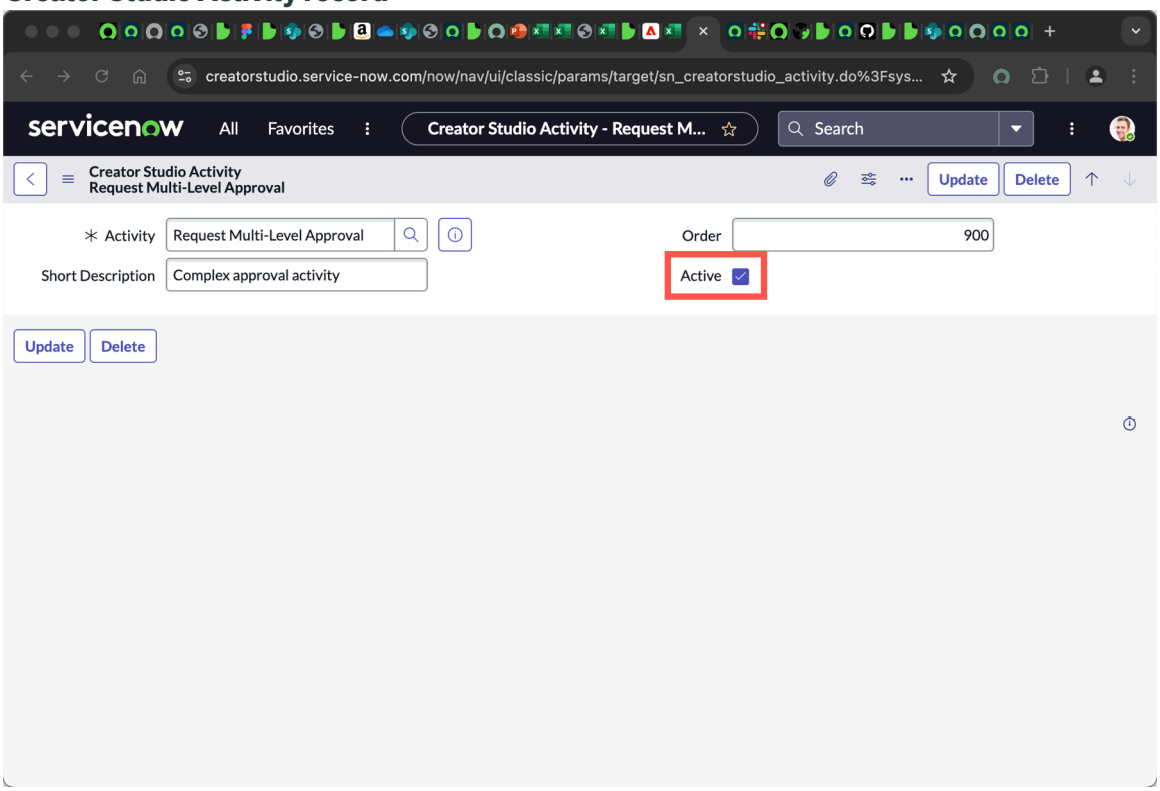
1. Navigate to **All** > *App Engine* > **Request App Administration** > *Creator Studio* **Activities**.
   All of the standard Creator Studio activities appear in the list, not just custom activities.
   The Creator Studio Activities table appears.

Creator Studio Activities table



2. Open the Creator Studio Activity record for the activity that you want to hide by selecting is **Short Description** field.
3. Hide the activity by clearing the **Active** check box.

Creator Studio Activity record



4. Select **Update** to save the record.

Managing Creator Studio deployments and monitoring app usage

After users create apps, they submit them for you, the admin, to review and deploy. How you deploy them depends on whether you have App Engine Management Center (AEMC) installed on the instance.

Managing app deployment with AEMC

Apps aren't deployed directly from Creator Studio. Instead, use App Engine Management Center (AEMC).

Find details on deploying apps in Managing deployments using pipelines in AEMC.

Managing app deployment without AEMC

If you don't have AEMC installed, you can use Pipelines and Deployments and the Deployment Tasks table to publish to the Application Repository. For more information, see Pipelines and Deployments workflow version 24.1.2.

You can view and edit details of the deployment on the **Scheduled Deployments** tab of Pipelines and Deployments For more information, see Managing app deployments using Pipelines and Deployments.

Monitoring deployed app usage

After you deploy an app to production, you can monitor its usage, adoption, and response rate, for each request form in the same way that you do for catalog items/record producers or scoped apps.

For example, you can use AEMC to monitor apps. For more information, see Managing custom apps using AEMC.

Artifacts that Creator Studio generates when users create an app

When a user creates an app in Creator Studio, the ServiceNow AI Platform creates several metadata artifacts, such as tables.

For each app users add in Creator Studio, the following artifacts are added to the ServiceNow AI Platform.

Installed with each created app

Artifact	Details
New table that extends from the base Request Task table	• Name: [scope]_request (for example, x_snc_marketing_request) • Label: [App name] + Requests (for example, Marketing Request) **Note:** All of the form questions are stored as catalog variables in the app's Request Task table.
Fulfiller role	• Name: [scope].agent • Provides access to the table created by the app
Access Control Lists (ACLs)	Provides CRUD access to the new table for the fulfiller and requester roles.
Category for the new app's workspace	Name: [App Name] + Requests
Default lists for the new workspace category	• Open (active=true) • Open – Unassigned (active=true^assigned_toISEMPTY) • Closed (active=false) • All (no conditions)
Dashboard in the new workspace category	Dashboard name: [App Name] + Requests Contents: • A row of single scores: ○ Open requests (active=true) ○ Unassigned requests (active=true and assigned to is empty) ○ Closed last 30 days (closed >= 30 days ago) • A simple list showing all open requests

Administering user access for deployed Creator Studio apps

After users build apps in Creator Studio and you deploy them to production, as an admin, you are responsible for granting access to those apps to users.

Assign user access in production

Applications must be deployed to the production instance, which is where you assign user access.

Users with the agent role are fulfillers, and should automatically get access to apps built in Creator Studio.

Assign users the agent role for the app's scope, for example, `x_snc_jd_it_fulf_0.agent` for an app with the scope `x_snc_jd_it_fulf_0`. For details on assigning roles to users, see Managing roles.

Restricting user access to forms

By default, all users on the production instance get access to forms built in Creator Studio, but Creator Studio developers can choose from existing access criteria if they want to restrict access. The **Available For** and **Not Available For** form settings are lists of User Criteria Records. For more information, see Set security for items and categories.

Developers can work with admins to define new access criteria for their app's forms to define the proper access from within Creator Studio. For more information, see Edit the settings for a form in Creator Studio.

Personas for apps built in Creator Studio

Requester

Someone requesting something, like a piece of equipment or permission to do something.

Fulfiller

Someone who works on requests. Fulfillers may also approve or deny requests, depending on any approval automation for the app.

Building apps with Creator Studio

By now, your admin has set up Creator Studio, given you permission to build an app, and you've got a great idea for a request-fulfillment app! How should you think about using Creator Studio to create your app?

Create forms to gather request information

When finished, your app will look like a catalog of items that people can request. To create your app, you'll:

1. Select one of the available catalog templates.
2. Customize a form to be included in the catalog using a WYSIWYG editor.

   A form describes one item in the catalog and provides space for questions that a requester will answer to specify details about what they want, such as the model number of a laptop. If your catalog needs multiple items, you can create multiple forms.

Automate workflows and simplify tasks with playbooks

Streamlining your request-fulfillment process goes beyond just collecting information. Creator Studio empowers you to automate tasks with powerful playbooks. Think of them as pre-programmed workflows that kick in after a form is submitted.

Here are some things that playbooks can do:

- **Effortless approval routing:** Automatically route requests to the right person for approval, eliminating manual steps and bottlenecks. Imagine a laptop request automatically going to the IT manager for a thumbs-up.
- **Intuitive playbook builder:** No coding required! Create custom playbooks using an easy-to-use editor.

The playbooks you create use the information that people supply when they fill out your forms. So, the more details you ask for, the smoother the process runs.

Workspaces give fulfillers a place to address requests

Once deployed, your app will sit in its own workspace category, which makes it easy for fulfillers to answer requests. You can customize the workspace or use it as-is.

For sys admin eyes only: Every app built in Creator Studio adds a record in the Request App Config table. The name of the table follows the format of scope_request, for example, x_snc_02_03_request.

Find existing apps in Creator Studio

Perhaps you've been given permission to work on an app someone else created, or you want to add new items to the catalog in your app. The Creator Studio home page acts as your central hub, listing all the apps people in your company have built using Creator Studio.

Before you begin

Summary:
In this topic, you'll learn how to:

- Find an app in Creator Studio.
- Open an existing app.
- Perform actions on an app without opening it.

Get permission

To work on an app, your system administrator has to assign you the role of sn_creatorstudio.user or sn_creatorstudio.restricted_user.

About this task

Techy detail:
All the apps listed on the Creator Studio home page have an entry in the Request App Config table (sn_creatorstudio_request_app_config).

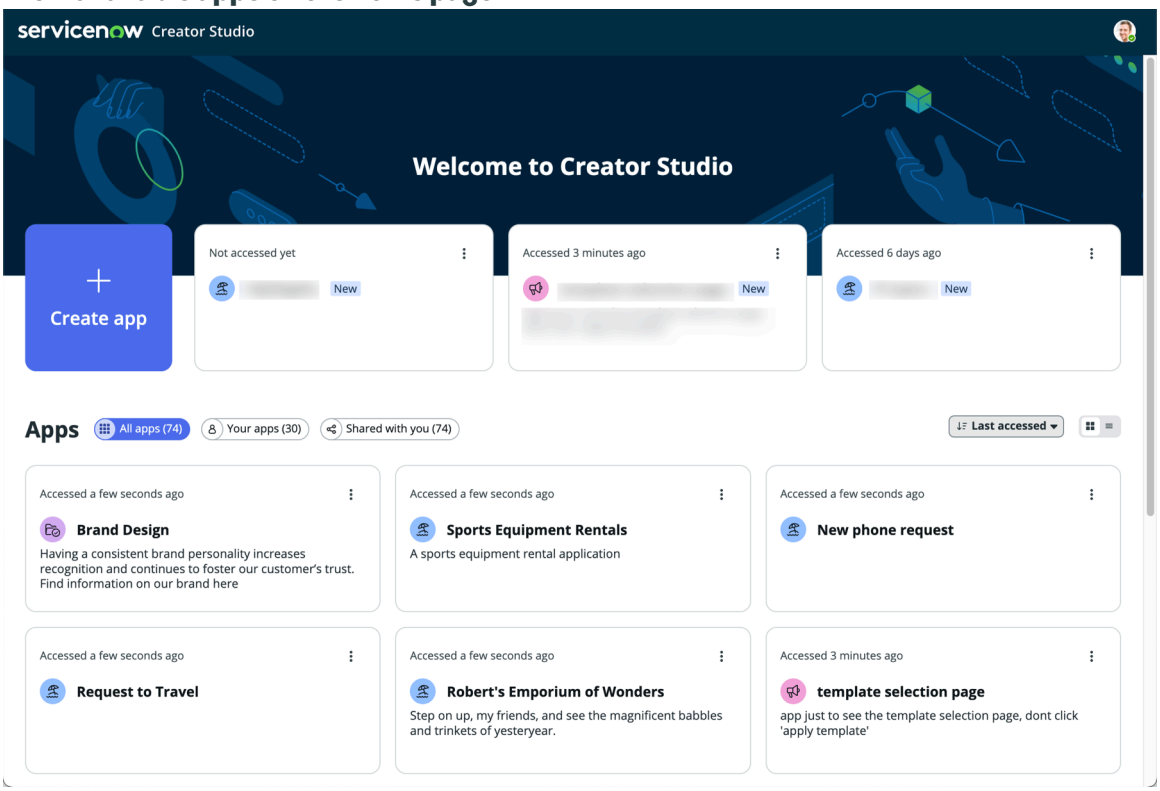
Find the app you want to work on

There could be a lot of apps listed on the Creator Studio home page. Here's how to find the one you want to work on.

Procedure

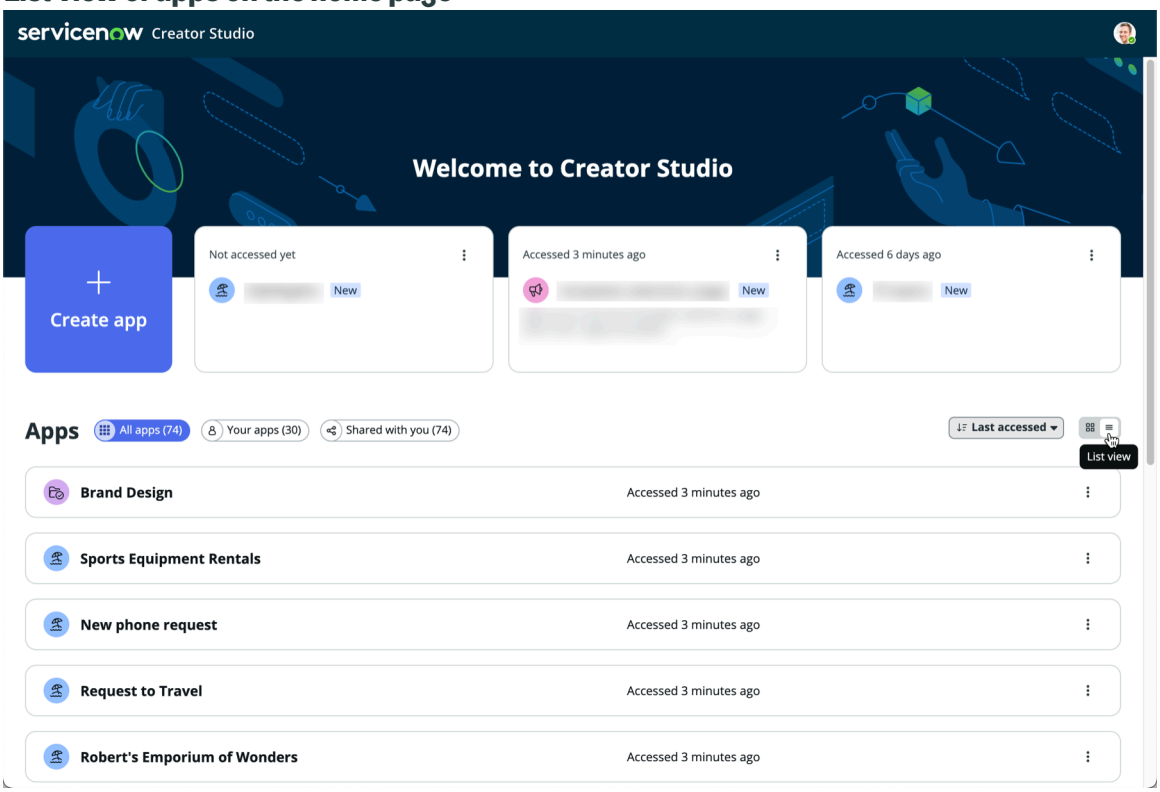
1. Go to **All** > **App Engine** > **Creator Studio**.
   The Creator Studio home page appears, which displays the apps most recently created by everyone in your company using Creator Studio.

View available apps on the home page



2. **Optional:** Would you prefer to see the apps in a list instead of displayed as cards? Select the view icon ( ) and switch to **List View**.

List view of apps on the home page



3. **Optional:** Narrow down the list of apps displayed by selecting one of the following:
   - **All apps**: Displays all the apps everyone in your company created using Creator Studio.
   - **Your apps**: Displays only the apps that you created. (You own the apps that you create.)
   - **Shared with you**: Displays only the apps you didn't create, but the owner said you could work on.

     For more information about working collaboratively on an app, see Delegated development and deployment.

4. **Optional:** Sort the list of displayed apps by selecting the drop-down sorting menu and then selecting one of the following options:
   - **Last accessed**: Lists the displayed apps based on the last time you worked on them; the most recently accessed app appears first in the list.
   - **Newest**: Lists the apps in the order of their creation date, newest to oldest.
   - **Oldest**: Lists the apps in the order of their creation date, oldest to newest.
   - **A-Z** or **Z-A**: Lists the displayed apps alphabetically in the order you choose.

5. **Optional:** 5. Select an app to open it in Creator Studio or select the more actions icon (…) and select one of the following:
   - **Open in new tab**: Open the app in a different browser tab.
   - **Manage collaborators**: If you're the app owner, you can invite others to work on the app with you. For more information, see Manage collaborators for an app in Creator Studio.
   - **Copy link**: Send collaborators a link to the app so they can find it.
   - **Settings**: Update the app's image and other settings. For more information, see Edit an app's settings in Creator Studio.

6. **Optional:** Select **Request access** to open an app that you don’t have permission to work on. To open an app, you must own it, or the owner must give you permission. For more information about requesting access, see Ask to work on an app in Creator Studio.

Result

Congrats:
Congratulations! You’ve found and opened an app or performed an action on it. Next, we’ll see how to work on it.

App creation in Creator Studio

Have you ever dreamed of creating your own app? With Creator Studio, you might be closer than you think! The following links will send you to the sections in this guide that will walk you through the steps.

Do you have permission to create an app?

If you don't have permission to build apps yourself, just ask an admin. They can give you permission, or you can give them an app name and a short description, and they can start the app for you. Then, you can jump in later and edit it.

Don't build your app from scratch!

Creating an application from scratch takes a lot of work. That's why Creator Studio does all the basic work for you. Just pick a form template (think starter kit) and provide some basic info (like a name). Later, we'll show you how to customize your app.

Edit your app's look and feel

Want your app to stand out? By editing the settings, you can give it a unique name, a head-turning image, and other great features.

Create the foundation of an app in Creator Studio

Before you can build out an app for people to use, you must create its foundation. Eventually, you'll have to customize the default values, but in the following procedure you'll select a catalog template for your first form and specify basic info for the app you're building in Creator Studio.

Before you begin

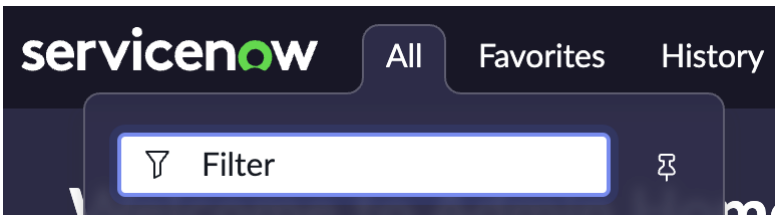
If you don't have permission to create an app, you can request that a teammate create one for you. For more information, see Ask an admin to create an app for you in Creator Studio.

Your administrator must add you to the Creator Studio Users group.

Procedure

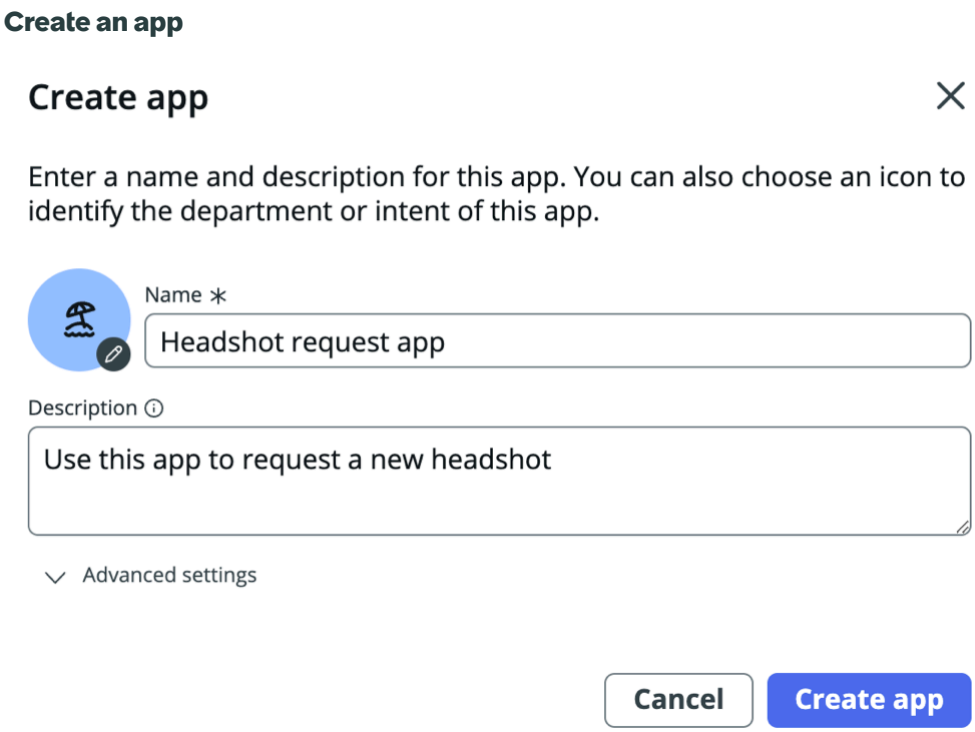
1. Let's open Creator Studio. On your instance, select the **All** tab.

All tab



This tab lets you see all the apps installed on your instance.

2. In the white text field, enter `App Engine`.
   The list of apps displayed underneath the text box now pertains only to App Engine.
3. In the list, select **Creator Studio**.
   That's how you open Creator Studio! You've landed on its home page. Under **Apps**, you see all the apps that people already created.
4. Select **Create app** to begin your journey.
   - If you're a system administrator, you can read more about this topic in Application collaboration.
   - If you want to know how to request an admin to create the app for you, check out Ask an admin to create an app for you in Creator Studio.
5. Fill in the fields on the modal that pops up.



   a. Give your app a **Name**, which should be descriptive and intuitive.

   b. Describe what your app does in the **Description**.

      For example, `This app enables a person to request office equipment.`

   c. **Optional:** Select **Advanced settings** and confirm that your app's name and tables are unique by specifying the **Scope**.
      For more advanced information about scopes, see Application scope.

   d. Select **Create app**.

   Okay, you've created the beginning of your very own application! You've named and described it. A list of catalog templates appears.
6. Select a catalog template from the list, which your admin can customize.

You may not see a catalog template that is exactly what you want. So, choose one that's the closest. If none look close, select the **Creator Studio Default Template** option, if your admin hasn't removed it. It's your best bet.

Tip:
Feel free to click through the list of catalog templates. Previews will help you pick the one you want.

Don't worry if you don't see other catalog templates to choose from! Your admins may not have created any custom catalog templates for you yet.

7. Select **Apply this template**.
8. Add a form to a catalog to specify its business area.

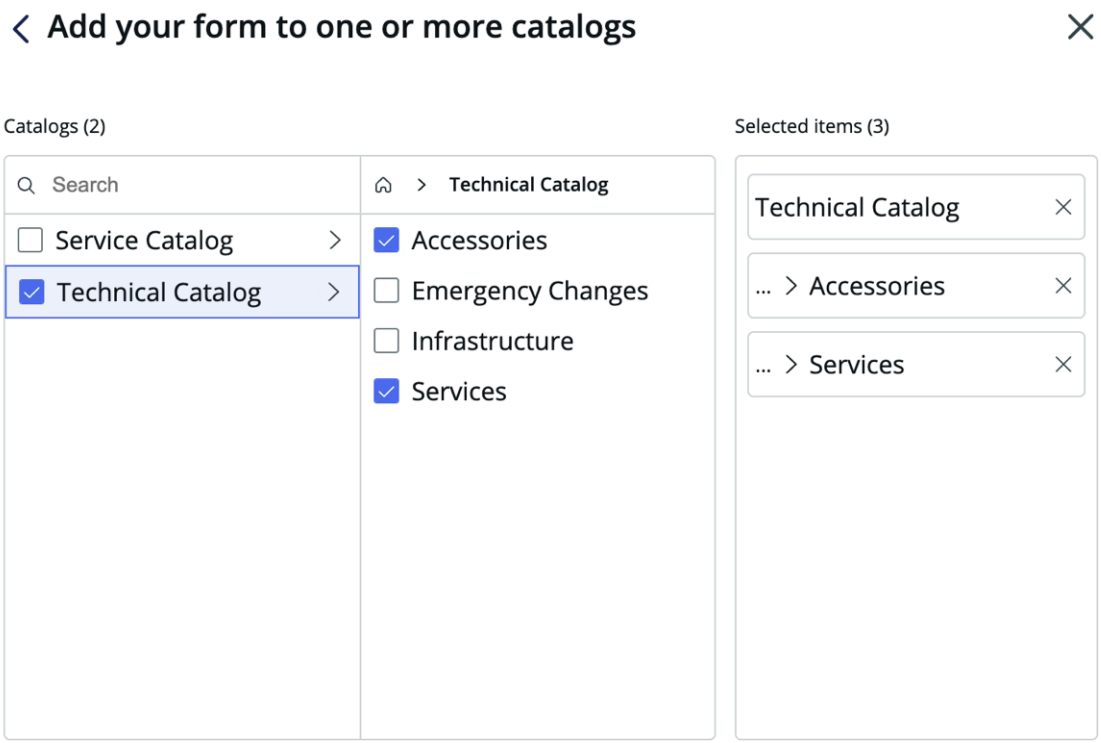
Note:
You can skip this step for now and revisit it later when you have a better idea of where it fits.

The available catalogs are configured by your admin, contact them if you don't see the one you want.

a. Select the **Edit button** (Edit) for the Catalogs and categories card.

b. Select the catalog that represents the business area the app will use.
For example, choose a service catalog that contains software and laptop cables. Expand the carat for each catalog to see its sub-catalogs.

Select the catalog



c. Select as many items in the catalogs as you need.

d. Select **Apply**.

You can edit any of the app's basic settings any time after you finish creating the app. For more information, see Creator Studio form settings.

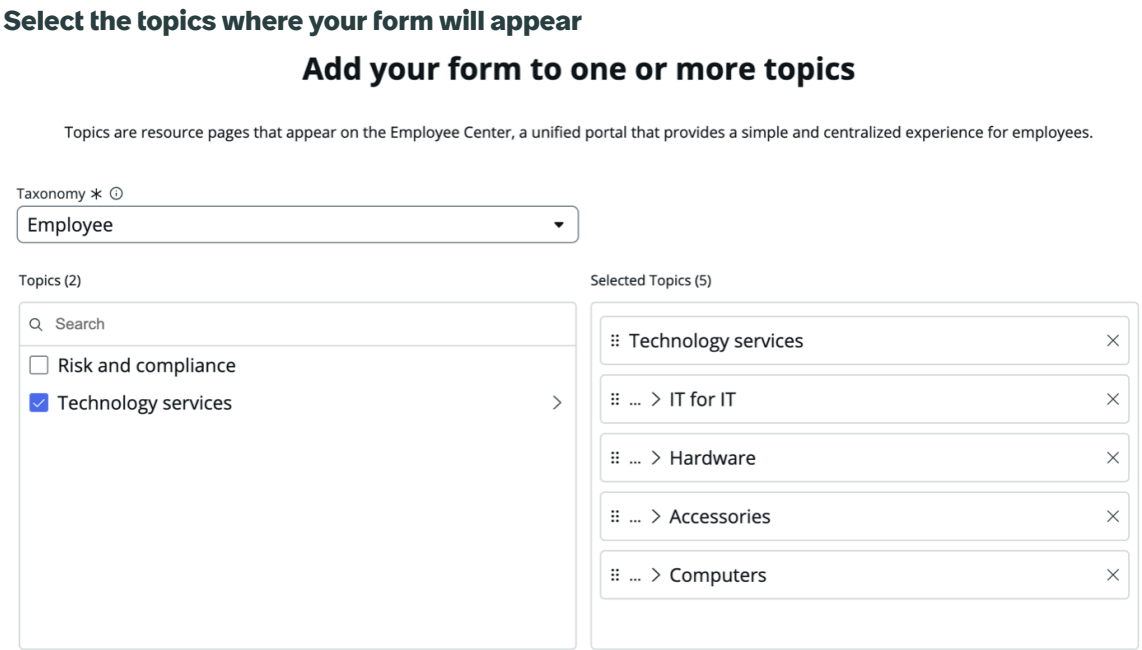
9. Next, choose one or more topics to specify where the form will appear.
Find out more about topics in Associate a catalog item with a taxonomy topic in Employee Center, and more about taxonomy, which is a categorization method, in Unified Taxonomy for Employee Center.

> **Note:**
> You can skip this step for now and revisit it later when you have a better idea of where it fits.

a. Select the **Edit button** (Edit) for the Topics card.

b. Choose the **Taxonomy** page where you want the form to appear, such as **Employee**.
Taxonomy is a method of categorization that Employee Center uses.

c. Select the topic(s) that represent the Employee Center areas where you want the form to appear.
For example, choose a topic that contains technology services, and expand its carat to see each of its sub-topics.



d. Select the topics where you want the form to appear, as many as you need.

e. Select the **Apply** button to save your changes.

10. Select **Save and continue**.

11. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

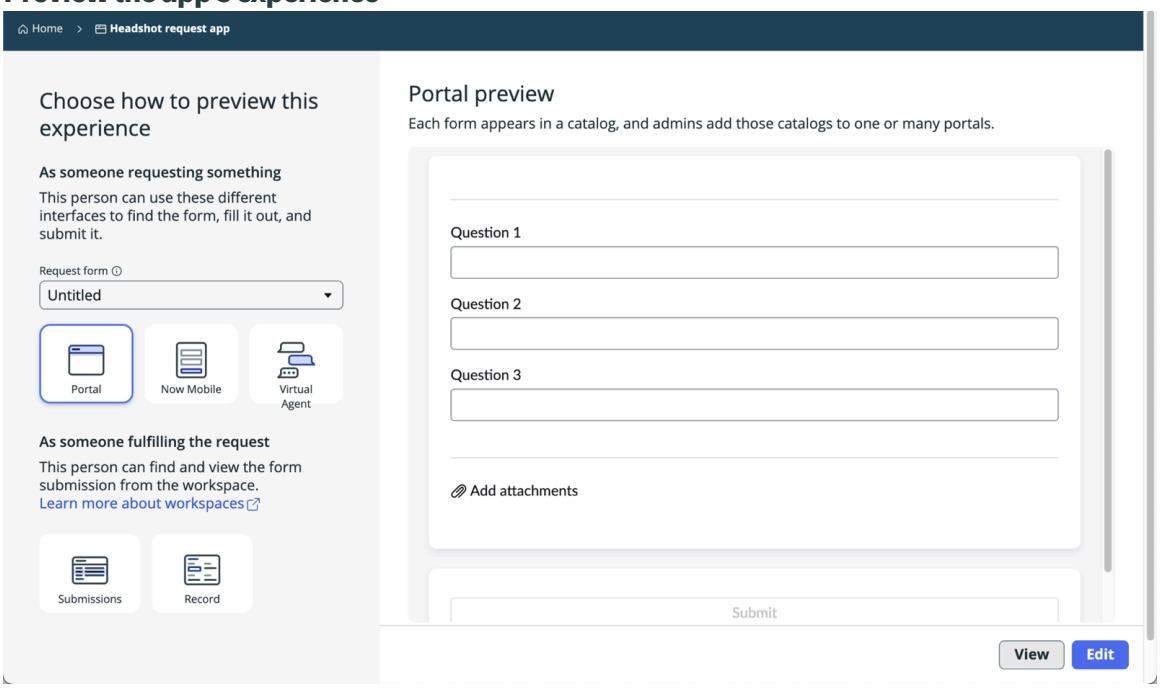
    Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

    - **Portal** (see a preview of how it'll appear on a desktop website)
    - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
    - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

    Note:
    Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

    The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

    Preview the app's experience

    You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

Result

Congrats:
Hooray! You've named your app and selected a catalog template that provides the basics for how your app's first form will look and work. Next, we will customize the forms people will use to fill out requests in your app in the section Working with forms in Creator Studio.

To learn all the things you can do as the app owner when building the app, see Application collaboration.

For sys admin eyes only: Every app built in Creator Studio adds a record in the Request App Config table. The name of the table follows the format of scope_request, for example, x_snc_02_03_request.

Ask an admin to create an app for you in Creator Studio

Don't have permission to build an app? Ask your admin to get you started by creating a basic app for you. Just give them a name and tell them what you want the app to do. Once they start it, you'll take over and modify it.

Before you begin

To request an app to be created for you, the admin needs to assign you the role of sn_creatorstudio.restricted_user.

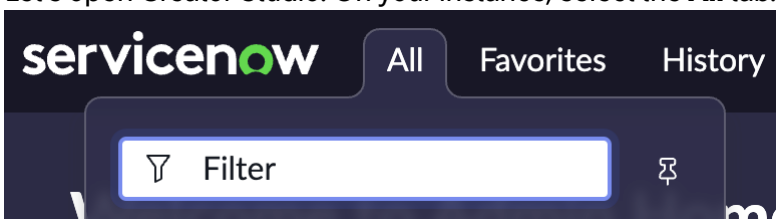
About this task

When you ask an admin to create an app for you, you automatically become the owner of it. For more information, see Collaboration in Creator Studio.

Here's how to ask an admin to create an app for you.

Procedure

1. Let's open Creator Studio. On your instance, select the **All** tab.



   This tab lets you see all the apps installed on your instance.

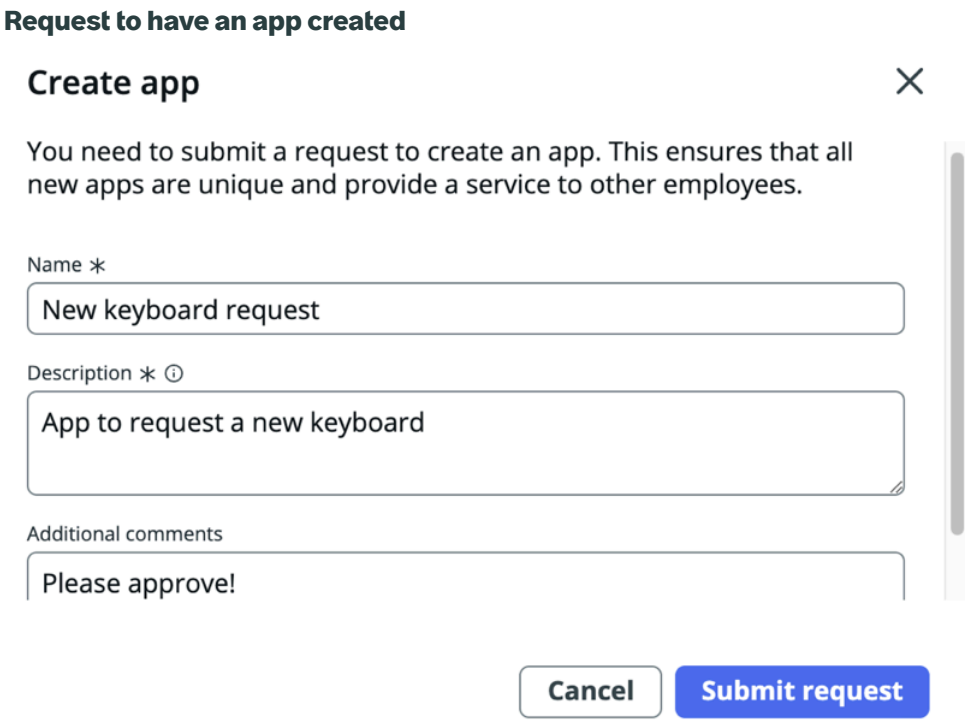
2. In the white text field, enter `App Engine`.
   The list of apps displayed underneath the text box now pertains only to App Engine.
3. In the list, select **Creator Studio**.
   That's how you open Creator Studio! You've landed on its home page. Under **Apps**, you see all the apps that people already created.
4. Select **Create app** to get started with your request.
5. Fill in the fields on the modal that pops up.

   a. Give your app a **Name**, which should be descriptive and intuitive.

   b. Describe what your app will do in the **Description**.

      For example, `This app enables a person to request office equipment.`

   c. Enter any additional comments that you want the admin to know when approving your request.



6. Select the **Submit request** button.

Result

After you ask your admin to create the app, they'll review and approve the request. They might need more details about the app from you. You'll get an email notification once they approve and create it. At that point, you're the owner of the app. You can find your app in Creator Studio and start building out the app's forms, automation, and workspace category.

To learn all the things you can do as the app owner when building the app, see Application collaboration.

For admins:

- Every app built in Creator Studio adds a record in the Request App Config table. The table's name follows the format of scope_request, for example, x_snc_02_03_request.
- You'll review and approve app requests in the Application Tasks table of the Request App Administration app.

Edit an app's settings in Creator Studio

Maybe you want to jazz up your app a bit, for example by giving it a new name, tweaking the description, or swapping in a new image. All it takes is just a few clicks in the settings! Let's dive into how you can do it, even if you're not a tech whiz.

Before you begin

To edit an app's settings, you must be given permission to work on the app.

About this task

There are some things you can't change after you've created the app, like its table or scope. If you want to change things up big time, you'll need to start from scratch with a new app.

Procedure

1. Head over to the Creator Studio home page by going to **All** > **App Engine** > **Creator Studio**. You'll see a list of all your apps right there on the home page.
2. Select the app you want to work on.
   It'll open up so you can start making changes.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

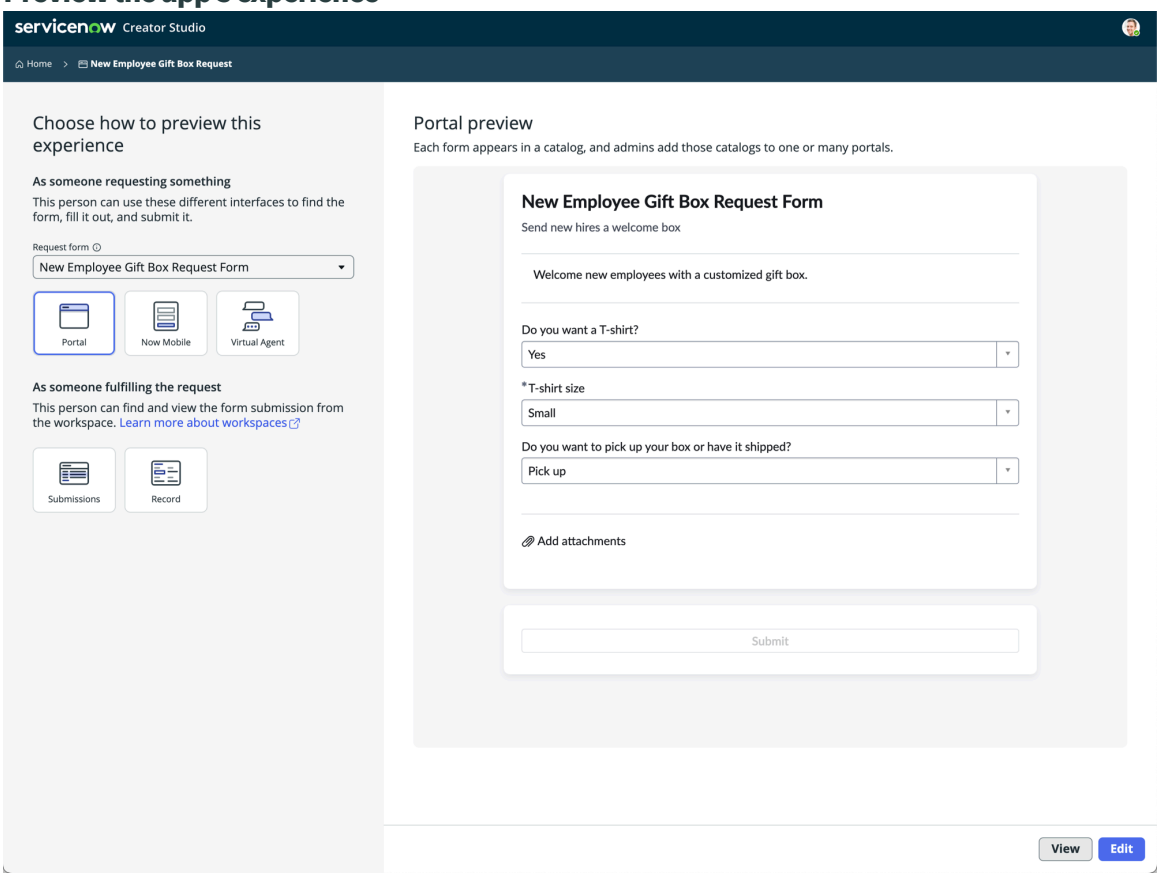
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

   Note:
   Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

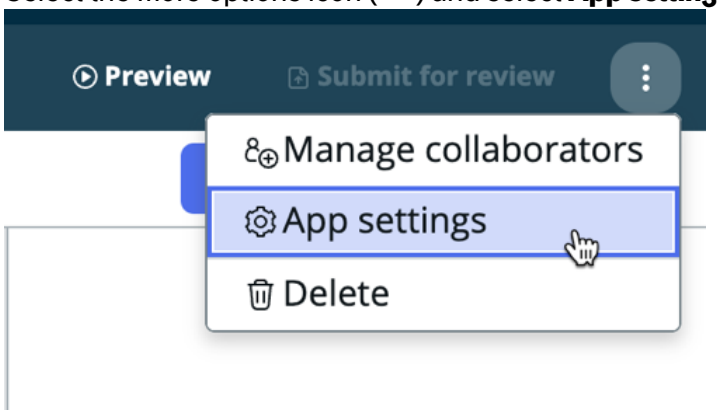
   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

Preview the app's experience



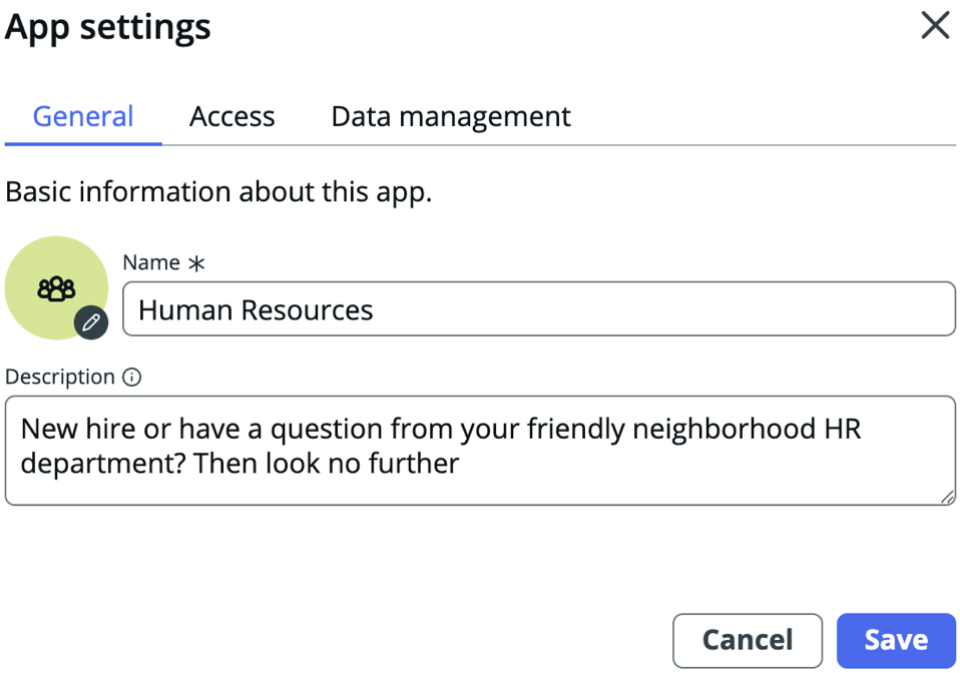
You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the more options icon ( ) and select **App settings** from the menu that pops up.



5. Now it's time to make those changes on the **General** tab, including updating the app's image. You can't make changes to security settings on the **Access** tab after the app has been created.

App settings



6. Once you're happy with everything, select the **Save** button.

Result

And there you have it: With just a few simple steps, you gave your app a whole new vibe. So go ahead, get creative, and make it yours!

Collaboration in Creator Studio

Sometimes you need help building out your app, and that’s ok! And sometimes other people need your help building their apps, which is great! This is where collaboration comes into play.

You can collaborate on apps you don't own by requesting permission to edit them, and invite other people you work with to collaborate on your apps with you.

Collaboration, also referred to as delegated development, builds on the existing delegated development feature set in the ServiceNow AI Platform. It enables developers to invite other developers into apps so they can co-create and develop the app together. Depending on your permissions, you can invite others to collaborate on an app with you, or request to join someone else's app.

Note:
You must have an App Engine Enterprise license to take full advantage of collaboration.

The ServiceNow AI Platform Collaboration app is automatically installed with Creator Studio. For more information about the Collaboration app, see Application collaboration.

Collaboration descriptors: Owners and Editors

So, you realize you either need help with your app, or you want to help someone else building their app. What's next? You need the right collaboration role.

There are two standard types of collaborators when you co-develop an app with other people: Owners and Editors. These two roles are called collaboration descriptors. The collaboration descriptor that someone is assigned determines if they can assign, manage, and monitor delegated development permissions. For example, people who are Owners can do more than people with the Editor collaboration type.

If needed, your admin can define custom collaboration descriptors to select when managing collaborators, either in the Creator Studio Guided Setup or in the Collaboration app. For more information on custom descriptors, see Configure Creator Studio using Guided Setup and Create collaboration descriptors to assign permissions.

Default collaboration descriptors

Descriptor	Description
Owner	Owner of the application. • If you created the app, you're automatically the owner. • Owners can manage other collaborators for the app. • Owners have the delete app permission, which enables them to edit app settings, such as name, description, and icon. • Owners automatically get the delegated_developer role for the app.
Editor	• Editors can invite collaborators. • Editors have limited ability to edit the app.

Collaboration development permissions

You've designated Owners and Editors in your app, now what? You need to invite other collaborators to work in your app.

Collaboration permissions enable you to control who's building apps in Creator Studio. You assign permissions to developers (or users who deploy applications) so that they can develop and deploy applications.

If you invite someone to collaborate on an app and they don't have the Delegated developer (delegated_developer) role, an App Engine admin must approve the collaboration request. For more information, see Delegated development and deployment.

When you add a user or group, a collaboration task is generated behind the scenes, and an approval flow kicks off. If you have App Engine Management Center (AEMC) installed, your admin can review and approve/deny these collaboration request tasks there. If you don't have AEMC installed, admins can navigate to **All** > *App Engine* > **Collaboration** > **Collaboration Tasks**.

The collaboration task that goes to your admin provides information on which application a developer is being added to, and what permissions are granted. Approvers sometimes need to review these task records before they add developers to the application.

What can owners and editors do?

If you're in the Creator Studio users group, you can see all users and groups collaborating on an app, as well as their collaboration descriptors.

If you can't see the users and groups, you may have a different permission with more restrictions. But fear not! You can always contact your admin if you have any questions about who's working on an app.

The following table provides a list of general defaults for what owners and editors can do in Creator Studio.

Note:
For the full list of default Owner and Editor collaborator collaboration type permissions, see Customized app collaboration permissions in Creator Studio.

Collaboration on Creator Studio features

Creator Studio feature	Owner	Editor
Invite collaborators	Yes	Yes
Manage collaborators	Yes	No
Edit app settings, such as name, description, and icon	Yes	No
Create, edit, and delete forms	Yes	Yes
Create, edit, and delete automation	Yes	Yes
Manage the workspace for form submissions	Yes	Yes
Submit app for deployment	Yes	No
Delete app	Yes	No

Collaboration roles and instances on different versions

As admins implement Creator Studio, they may have it installed on a non-production instance while their production instance is on a previous version of the ServiceNow AI Platform that doesn't have Creator Studio. This mis-match of instance versions affects the Collaboration Approval Workflow, which specifies the non-production instance as the source and the production instance as the controller. If the controller doesn't have the version of the collaboration plugin that supports Creator Studio, collaboration is unsupported.

To ensure that users can use the Collaboration Approval Workflow regardless of instance versions, admins must assign the catalog _builder_editor role to Creator Studio user groups.

Ask to work on an app in Creator Studio

Want to work on an app but don't have access to it? No problem. We'll show you how to ask for access.

Before you begin

First things first, you need the right access to get involved. Your system administrator must give you the role of sn_creatorstudio.user or sn_creatorstudio.restricted_user.

About this task

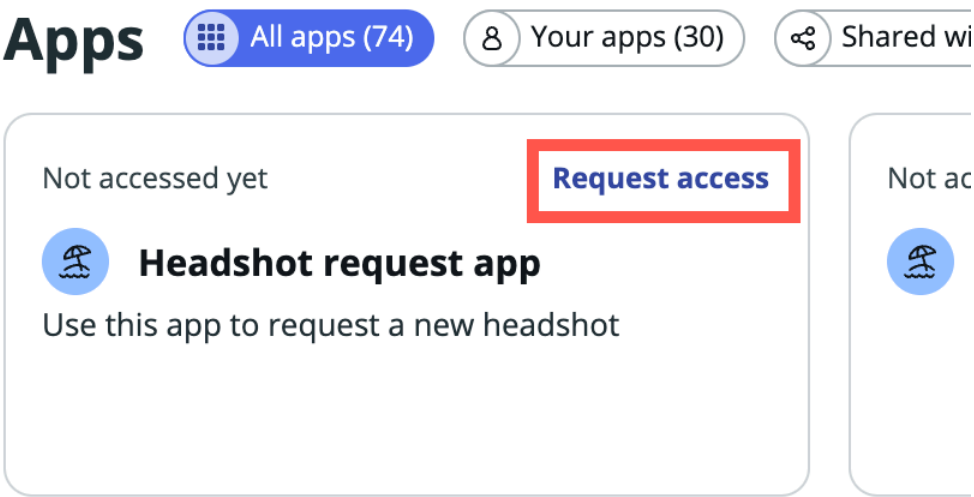
Getting permission to work on an app delegates you to its development, which means you can start pitching in and making things happen. For more information, see Delegated development and deployment.

Note:
Admins don't see the **Request access** option because their permissions enable them to access all apps.

Procedure

1. Head over to Creator Studio by going to **All** > **App Engine** > **Creator Studio**.
   You'll see a list of all the apps built in Creator Studio right there on the home page.
2. Look for the app you want to help with and select its **Request access** link.

Request access



3. **Optional:** Use the **Enter an optional message** field if you want to leave a little note explaining why you're interested in joining the app. Maybe you've got some great ideas, or just want to learn the process.
4. Once you're ready, select the **Request access** button.

Result
With any luck, the App Engine admin will approve your request, and you'll be in! Then, it's time to roll up your sleeves and start working on the app.

Note:
Users with a delegated developer role are automatically approved for collaboration.

What to do next

Important:
After your request to join an app gets approved, you must sign out of the ServiceNow AI Platform and then sign back in to access the app.

If your organization uses App Engine Management Center (AEMC), App Engine admins approve collaboration requests there.

If your organization doesn't have AEMC, admins approve requests to join apps in the Collaboration Tasks table. For more information, see Approve a collaboration request.

Manage collaborators for an app in Creator Studio

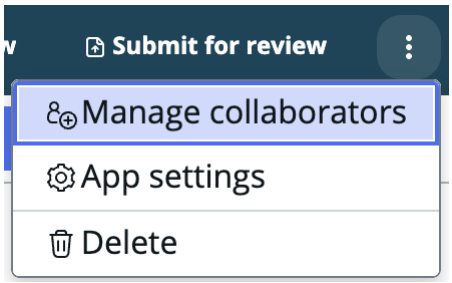
Add or remove people who can work on an app, or change their permissions for the app using the **Manage collaborators** option.

Before you begin
Your system administrator needs to assign you the Manage collaborators permission. Admins can also manage collaborators.

About this task
You can invite collaborators or manage collaborators depending on your permissions for each app. For example, if you have the Invite collaborators permission, you can't change the collaborator type of existing collaborators, such as Editor or Owner.

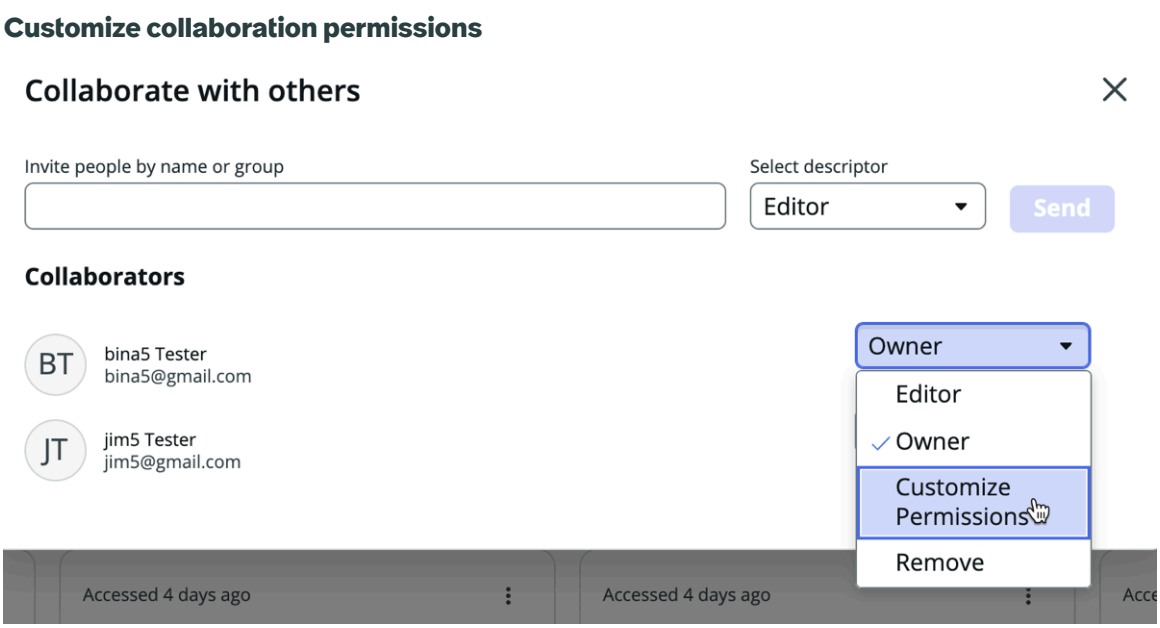
Procedure
1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Find the app you want to manage collaborators for.
3. Access the Collaborators with others modal in one of two ways:

- From the Creator Studio home page, select the app's more actions icon ( ) and select **Manage collaborators** from the menu that appears.
- Open the app, select **Edit**, select the more actions icon in the application header, and select **Manage collaborators** from the menu that appears.

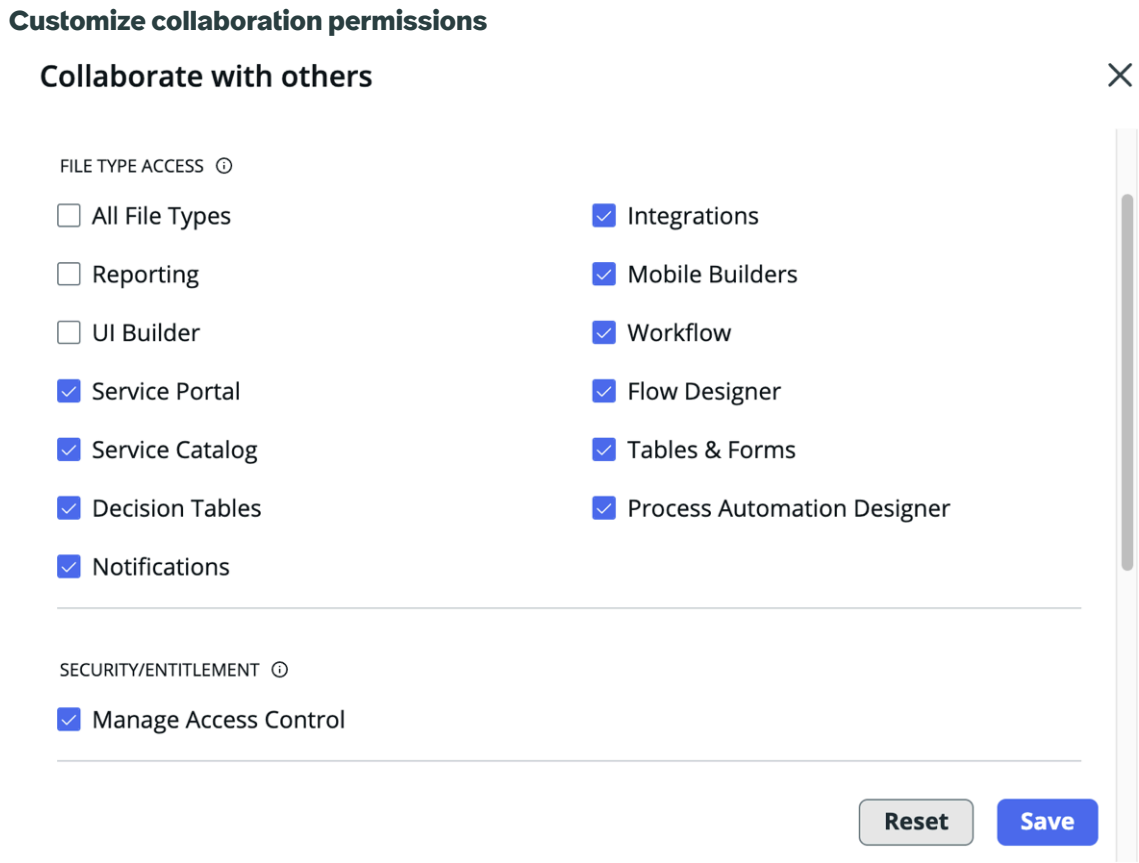


4. Change the permission of existing collaborators in the Collaborator section using the **Select descriptor** field.
   - **Owner** changes the user or group to be an owner for the app.
   - **Editor** changes the user or group to be an editor for the app.
   - **Customize permissions** enables you to select specific things that users can do with the app. For more information, see Customized app collaboration permissions in Creator Studio.
   - **Remove** revokes the user or group's access to work on the app.

   Note:
   The collaboration descriptor must be in the global scope for it to be used by others.



5. Customize collaboration permissions by selecting **Customize permissions** and selecting the permissions the user should have.



6. **Optional:** Add any additional collaborators.
   a. Start typing the people or groups you want to invite in the **Invite people by name or group** field.
   b. Specify the collaboration level in the **Select descriptor** field.
   c. Select the **Send** button to start the approval process.
   You can specify as many users and groups as needed.

Result

Unless you customized permissions, which requires the **Save** button, your changes are automatically saved when you close the Collaborate with others modal.

What to do next

Your App Engine admin must then approve the changes to collaborators.

Working with forms in Creator Studio

To add items or services to your catalog, you must create a different form for each thing being requested.

Summary:

After reading this topic, you'll understand:

- How form templates get you started creating your own forms.
- How to customize the layout and questions on your forms.
- How to make a form (AKA catalog item) appear in your app.
- How to make a form stop appearing in your app!

What is a form?

A form is a list of questions that people answer to make a request. They might ask for a keyboard, a laptop, or permission to take a vacation. The answers to those questions provide the fulfiller with all the information that they need to fulfill the request. For example, if someone asks for a laptop, the fulfiller might need to know the size, the model number, the operating system, and so forth.

In addition to questions, forms can contain images, headings, and text descriptions to explain the questions.

You'll need a different form for every type of item or request that a requester can ask for. Why? Because the questions you must ask for a keyboard differ from the questions that you must ask to give a requester permission to take a vacation. So, there's one form for each type of request that people can make. You can think of a form as representing a catalog item.

Your app will likely have multiple forms, meaning multiple things the requester can ask for. We call a collection of forms a catalog of items.

Key terms:

Item

Something a requester can ask for. Each item requires its own form.

Catalog

A collection of items that represent all the items a requester can ask for.

Some examples of catalogs and items

The following table shows three catalogs. The final column shows the names of the items in each catalog. If you have three items, you'll need three forms.

Example catalogs and their items

Service catalog	Example app	Sample catalog items
HR self-service	Human Resources global people portal app	• Benefits coverage question • Time off request • Change of address needed
Branding resources	Marketing and branding app	• PowerPoint templates • Approved Zoom backgrounds • Logo files
IT fulfillment	Hardware request app	• Mouse device • Headset • Replacement charging cable

Let Creator Studio give you a head start

When you choose a form template to start creating your app, Creator Studio builds a form with questions you might like to use. The questions and their layout make up the form.

Key term:

Form template

A set of questions arranged on a form provided by Creator Studio.

The form template that Creator Studio provides is called the Creator Studio Default Template. It contains three questions to get you started. You can customize, add, and subtract questions on your form.

Your administrator can delete the default template, create additional form templates, and restrict who can fill out the forms. If you're an administrator, you can read more about this topic in Creating catalog templates for use in Creator Studio apps.

To store the forms that requesters fill out, Creator Studio creates a table by default. One row represents one filled-out form.

Your application can contain as many forms as you need. When you create a form, Creator Studio uses the app's request table to hold the submitted answers for each form. All requests from all of an app's forms are stored in the same table.

Types of questions you can ask

You get to pick the questions to ask requesters. Creator Studio provides two types of questions you can ask:

- **General questions** are where you give requesters an empty box to write a short answer (think name or email) or a list of choices to pick from (like a dropdown menu or those little circles you fill in).
- **Reference-based questions** are where you provide a list of choices that you define in a table.

You can add questions to forms that have been preconfigured by your admin, and can't be edited. Adding a **Question set** element to a form enables you to select from the curated options.

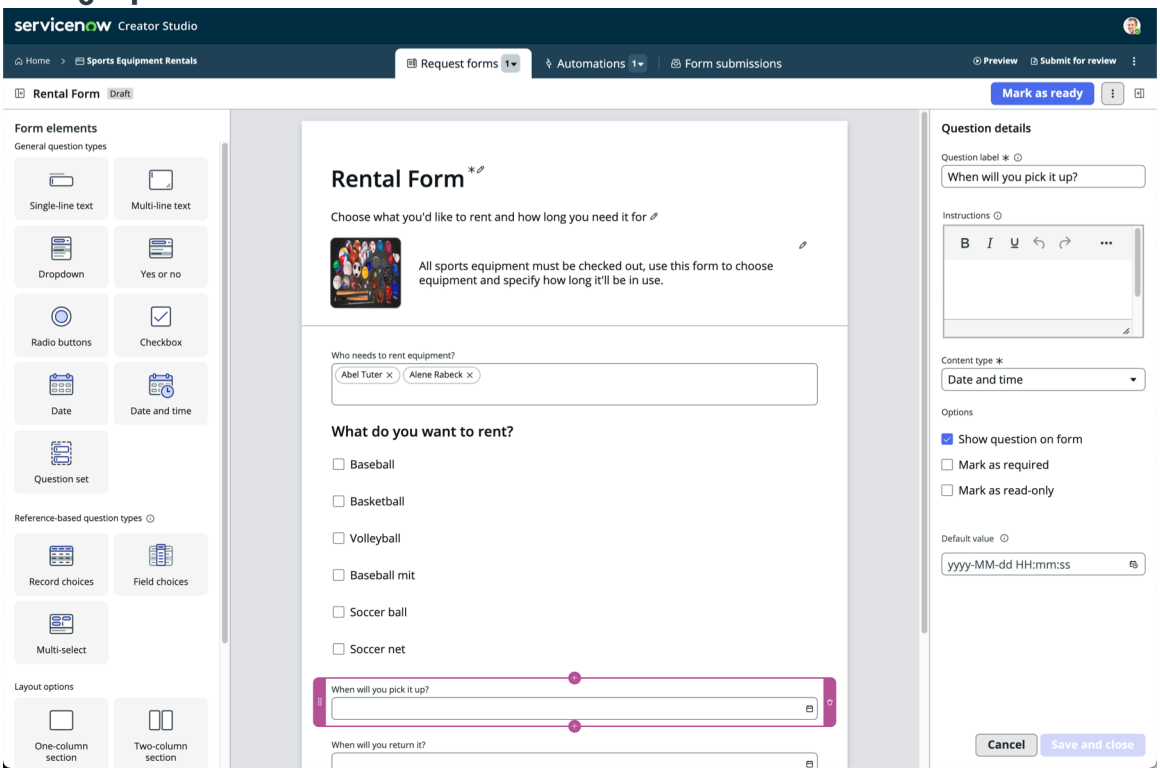
Forms can automatically update how questions appear (or are hidden) based on how users answer a them. We call this dynamic behavior. For example, if a user says they want a T-shirt for an event they're attending, you can make a **T-shirt size** field required.

Laying out the questions

Creator Studio provides an initial layout for its default form. You can use columns, images, divider lines, and more to customize the layout of the questions on the forms. There are two procedures that you can use to customize a form:

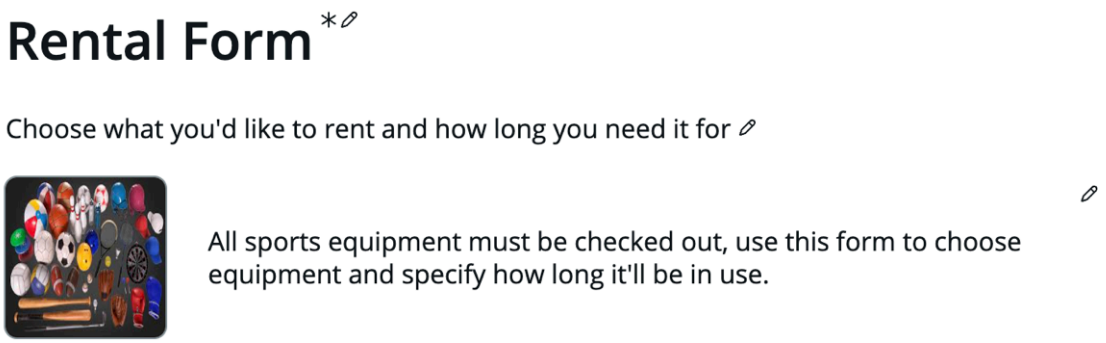
- Changing the questions – Change the number of questions, what they're asking, and adjust where questions appear on a form. Find information on changing questions in Customize your form for an app in Creator Studio.

Editing a question on a form



- Changing the look and feel of the form – Add pictures, write some text to explain the questions, and even move them around to make them flow smoothly. You can read about that in Change the layout of an app's record in Creator Studio.

Edit the look and feel of the app's tile



Where can I see forms?

After deployment, your app lives as forms in the Service Catalog and categories you specified when creating the forms.

Users can access those forms directly in Service Catalog, as well as Service Portal and Employee Center.

If you associate the app's form with one or more topics, the form will appear in the relevant, dynamically created topic pages in Employee Center. Find out more about topics in Associate a

catalog item with a taxonomy topic in Employee Center, and more about taxonomy, which is a categorization method, in Unified Taxonomy for Employee Center.

Making forms available for an app

Forms have two states:

- Draft
- Published

Keeping forms in a Draft state lets you work on them until you're finished. When you're pleased with the form, publish it by selecting the **Mark as ready** button. Only then will it appear in your application (assuming your app has been deployed).

A published form appears as an item in your catalog of offerings. For example, your site might enable requesters to ask for computers, mice, monitors, keyboards, technical support, and so forth. Each item requires a different form, and appears as a different catalog item in your app.

Key terms:
Don't confuse deploying your app with publishing a form.

- Publishing a form means that the form will appear as a catalog item only after your app is deployed. That's why the button label is **Mark as ready**.
- Deploying an app is what system admins do to make your app available for people to use.

Changing published forms

It's inevitable. You deploy your app and find that you need to revise the questions on your form. No problem! Just edit the form as you did when you created it.

Editing a form creates a new Draft version of the form that you can update and then **Mark as ready** when you're finished. Any changes you save to the Draft version won't affect the catalog item in your deployed app. To update the forms in the deployed app, you must redeploy the app.

Undoing form changes

You can select the **Undo all changes** option, available in the form header's more options icon (), to reset a form to the most recently published version.

Deleting published forms

You can't delete a form, whether it's in a Draft or Published state.

- If it's in the Draft state, just don't **Mark it ready**.
- If the form is in a Published state, you can hide it so it no longer appears in the catalog. See Hide a form from use in the ServiceNow AI Platform in Creator Studio for more information on hiding forms.

Customize your form for an app in Creator Studio

Forms help people ask for things they need, such as a new keyboard or permission to take time off. The default form that's added when you create an app in Creator Studio needs some changes to fit your needs. For example, you must add question labels to gather information about the request.

Before you begin
To customize a form, you must be given permission to work on the app.

About this task

You can add additional forms to your application if you need them. Additional forms are stored in the app's same table. No need to worry about those tables right now, but if you want to know how to add more forms besides the form that was generated when the app was created, check out Add more forms to an app in Creator Studio.

Summary:

After finishing the following steps, you'll know how to:

- Open the form that was automatically generated when the app was created.
- Add questions and images to the form.
- Arrange the questions and images that you added.

Let's open a form and customize it.

Note:

For example, you can create a form that people use to register for a company event, with questions like dietary preference and accessibility needs.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio**.
   All the apps built in Creator Studio appear on the home page.
2. Open the application that contains the form you want to customize.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

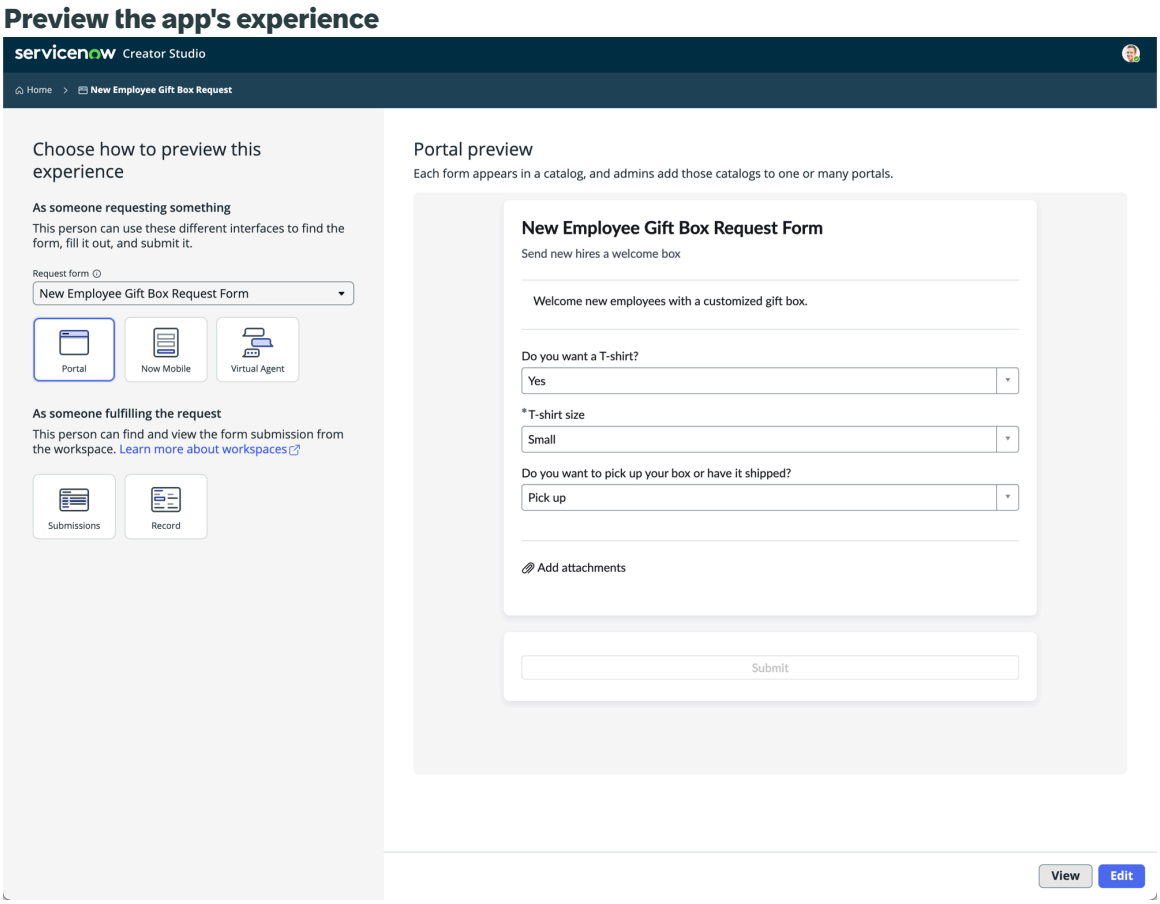
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

     Note:

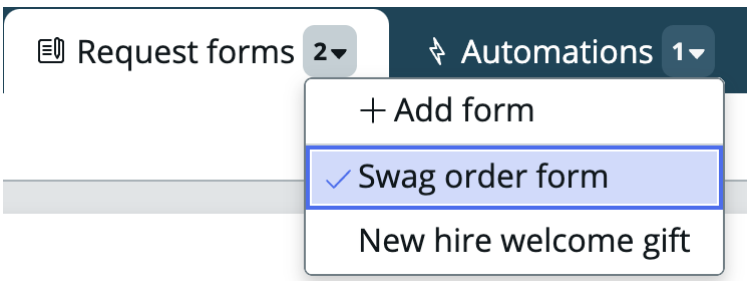
     Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Request forms** tab to make sure you're editing the correct form in your app.
   If you're not, select the form you want to edit.



5. Let's customize the form by doing one or more of the following optional steps.

   a. Add or modify the image that appears on your form by selecting the add image icon ( ) and then selecting an image.

   b. Change the form's title, short description, and other text by selecting those parts of the form and typing in your changes.
   You can enhance how the longer **Description** appears using rich text, such as font changes and sizing.

6. Next, add questions to the form your fulfiller needs to evaluate the request. To add and customize a question, complete the following steps.

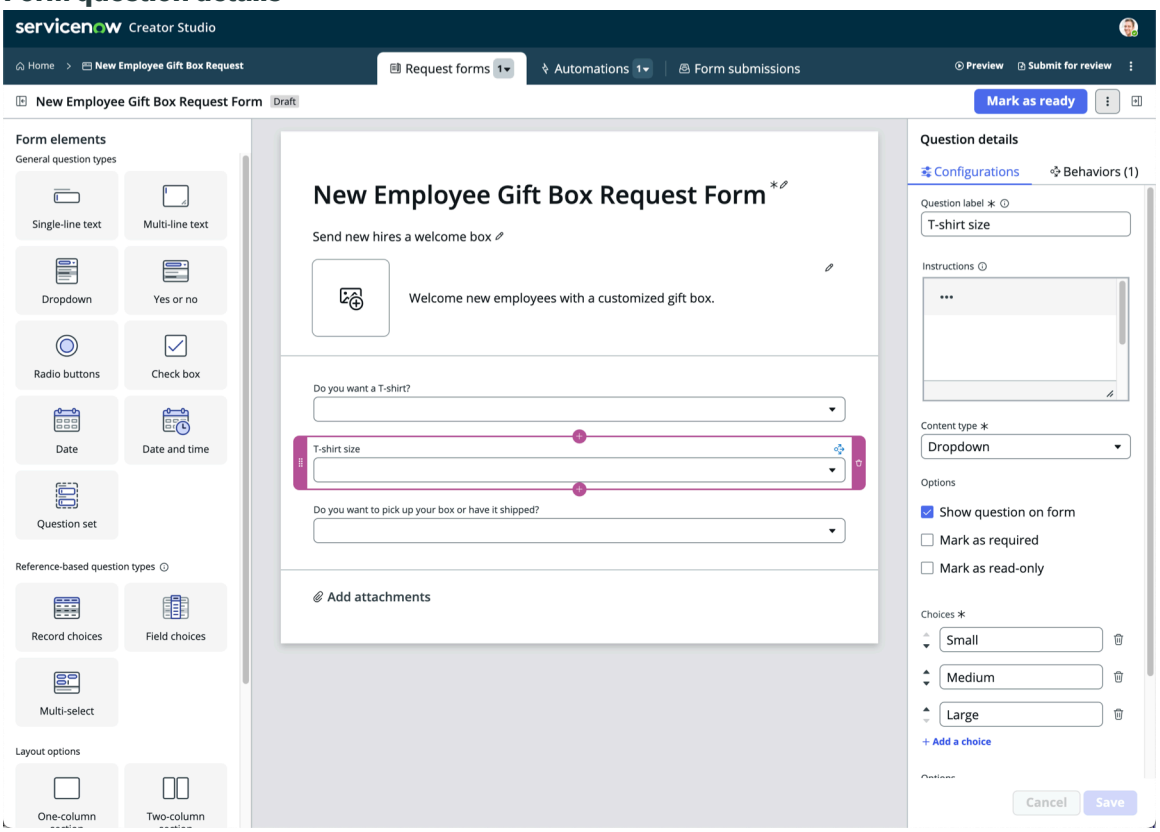
   a. In the Form elements panel, drag the type of question you want onto the form and drop it on the canvas where you'd like it. You can also add questions by selecting the add icon (+) that appears when you click on an existing question on the form.
   If you're adding a pre-configured **Question set**, you must select the question set from the modal that appears when you drag it onto the form.

   For a description of question types and how they're used, see Available question types in Creator Studio.

   b. Select the question.
   When you select a question, it's highlighted on the form so you know what you're working on.

   c. In the **Configurations** tab of the Question details panel, specify information about the question you added, such as whether a question must be answered for the requester to submit the form.
   The details vary by question type. For example, if you add a **Dropdown** question, you must supply the options to choose from.

Form question details



   d. **Optional:** Make the form's appearance change based on how users answer questions by adding dynamic behavior to it on the **Behaviors** tab.
   For example, if a user says they want a T-shirt for an event they're attending, you can make a **T-shirt size** field required. Get the details on adding dynamic in Make a form change based on responses in Creator Studio.

   e. Select **Save and close** when you finish modifying the question.

f. Revise or add more questions to the form using this procedure. To change a question type, select the question and then select the new question type in the **Content type** field of the Question details panel. Selecting a new type may introduce new values you must supply. Keep these specifications in mind as you create your questions:

- You can't change an existing question into a question set. To include a question set on a form, you must newly add it to the form.
- If you put two checkbox questions side-by-side on a form, they make a section. You can't add other types of questions to that section.

7. Now, let's arrange the questions and images that you've added to the form.

a. From the Form elements panel, drag the layout option you like onto the form and drop it where you want it to appear, for example, a **Divider line**.
Don't worry if you don't like the layout, just try another one by dragging it onto the form's canvas.

b. Revise the form layout you chose using the Section details or Question details panel (depending on the layout you're working on). You can do things like make text bold, add links, and so forth.

> **Note:**
> To edit or delete a section, you must hover over the section name and then select **Section** to see the section details in the properties panel, as well as the delete icon.

Selecting a section

For more information, see Layout options for forms in Creator Studio.

c. Select **Save** in the Section details/Question details panel when you're done revising the form's layout.

Result

> **Congrats:**
> Congratulations! You've customized the default form that came with your app.

What to do next

You must then mark the form as ready to publish it, which makes it available for the app to use. Find out how to publish a form in Publish a form for your app in Creator Studio.

You can select the **Undo all changes** option, available in the form header's more options icon (···), to reset a form to the most recently published version.

Make a form change based on responses in Creator Studio

Make a form update based on how users answer a question using dynamic behavior. For example, if a user says they want a T-shirt for an event they're attending, you can make a **T-shirt size** field required.

Before you begin

You must name the form before you can add dynamic behavior to it.

To add dynamic behavior to a form, you must be given permission to work on the app.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the application that contains the form you want to add dynamic behavior to.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

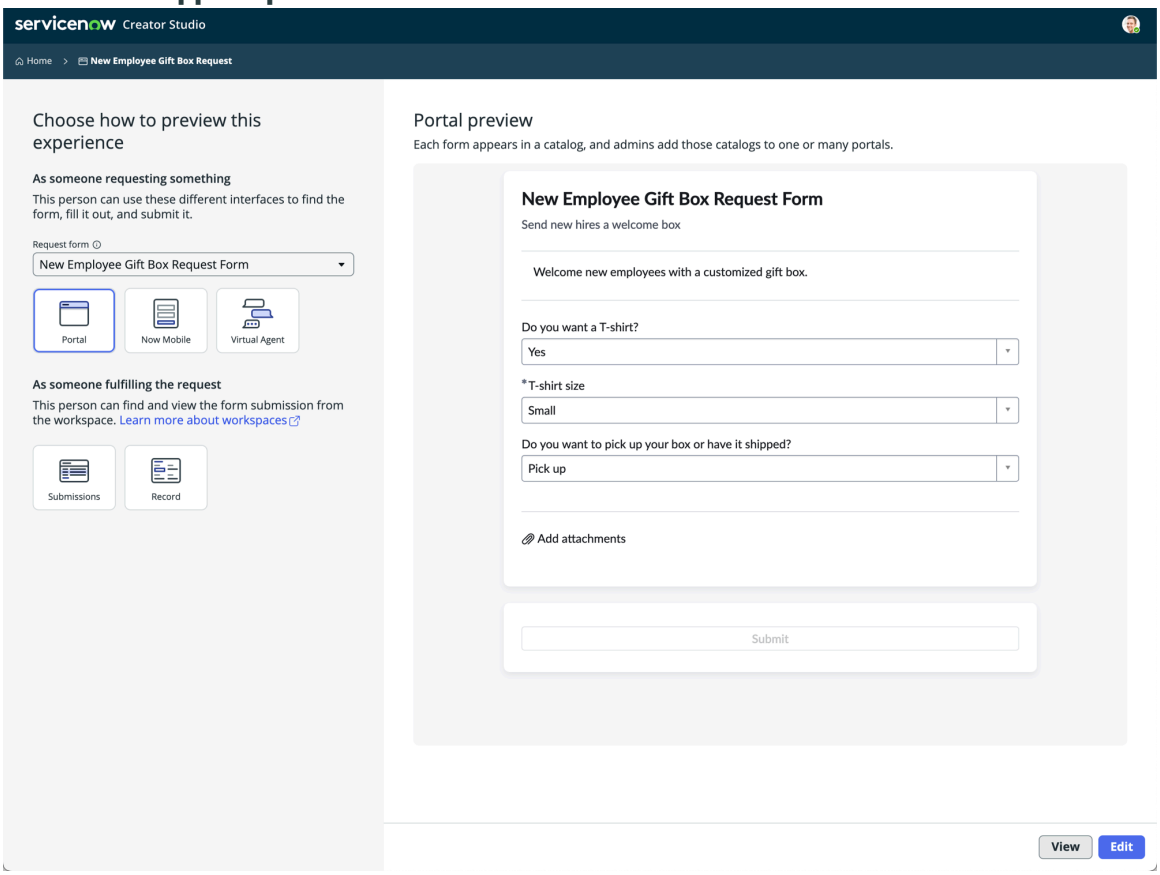
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

   > **Note:**
   > Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

Preview the app's experience

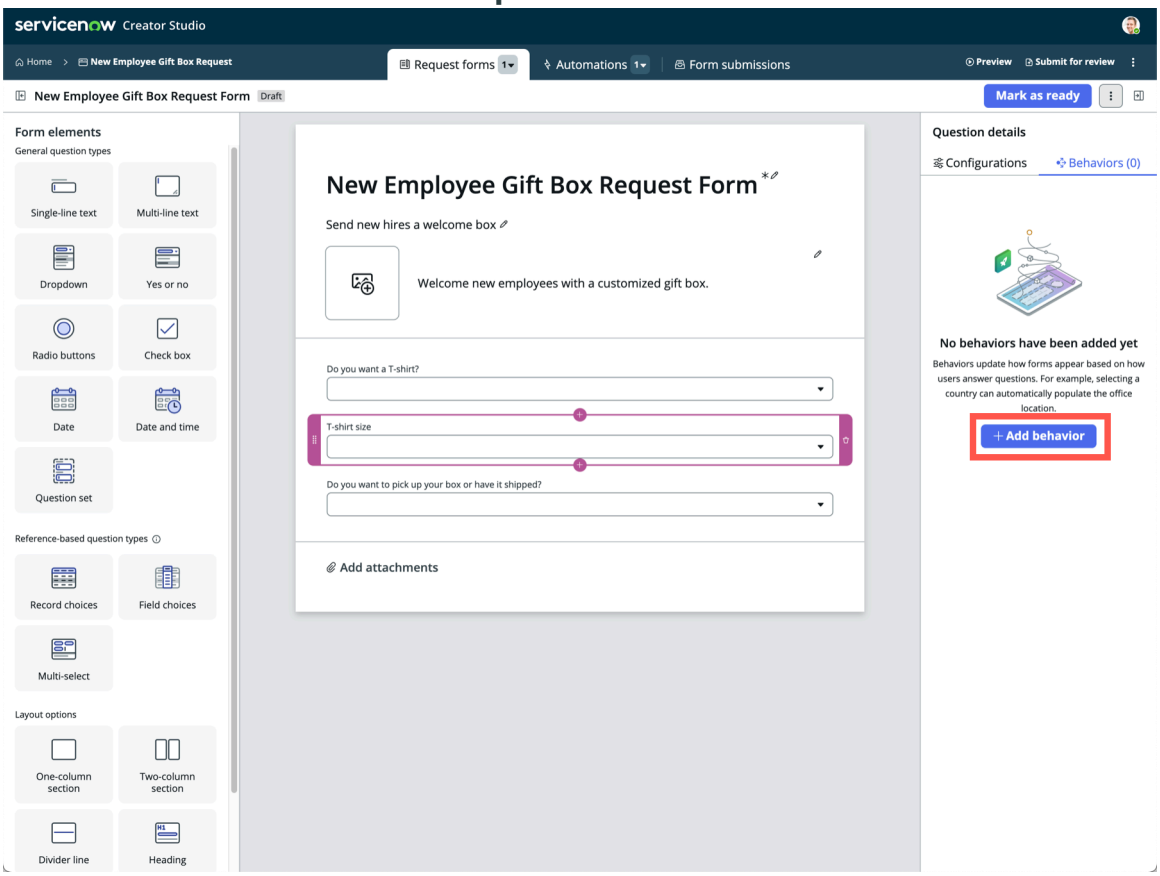


You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Check that you're editing the correct form in your app by selecting it from the **Request forms** tab.

5. Select the question that should be affected by how users answer one or more previous questions.
   For example, select the **T-shirt size** field to make it required if a user answers `Yes` to the **Do you want a T-shirt?** question.
6. Select the **Behaviors** tab of the settings panel.

Add behavior button on the Behaviors panel



7. Define the conditions that will make the form change in the Conditions section.

   a. Select **Questions** in the **Field** field to specify that you're choosing questions from the form. The name of the form automatically appears in the **Item** field.

   b. Choose the question whose answer triggers the dynamic behavior in the **Question** field. For example, select the **Do you want a T-shirt?** question.

   c. Select the condition **Operator** to determine the status of the selected field that will trigger the dynamic behavior.
   For example, you could select **Is** as the operator for the **Do you want a T-shirt?** field.

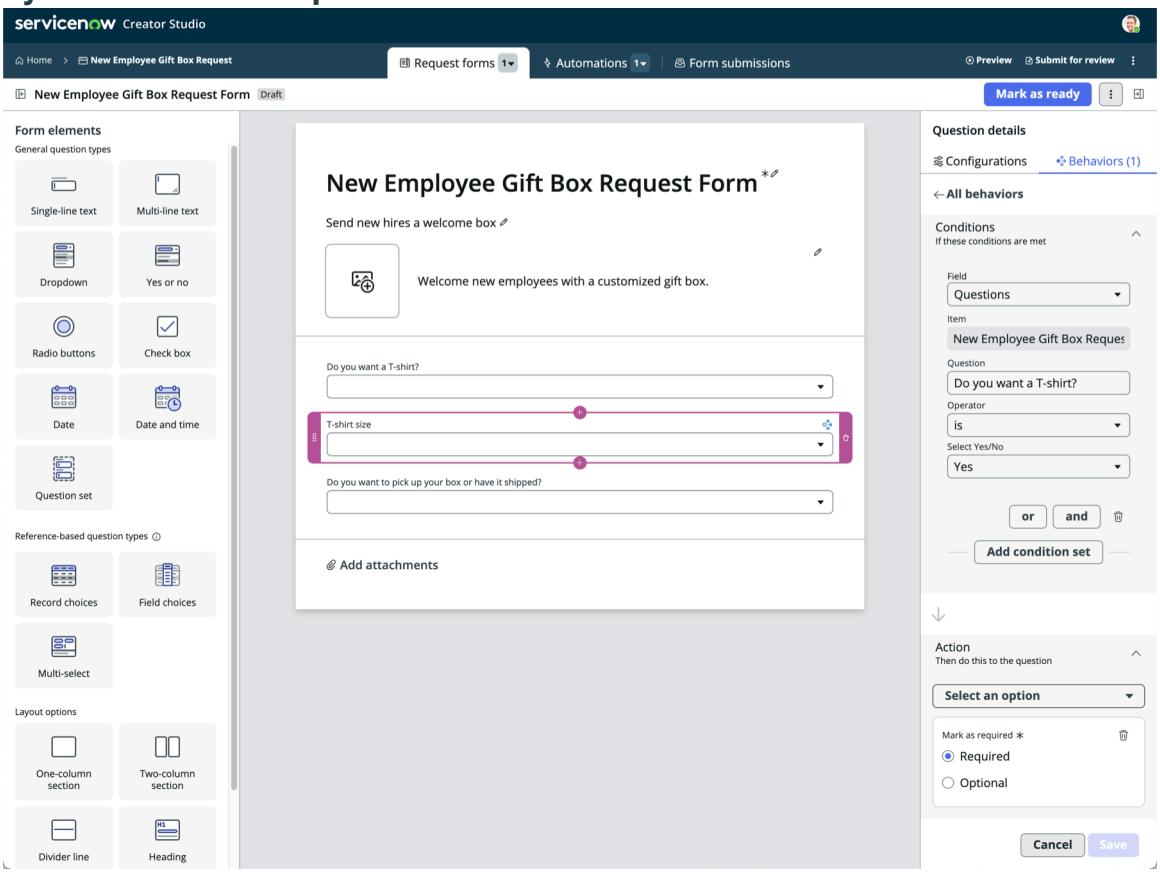
   For more information, see Condition builder.

   d. Enter or select the value for the user's response to the trigger question.
   For example, the value could be **Yes** if they want a T-shirt.

   e. **Optional:** Specify another field and value to create a more complex set of triggering conditions by selecting the **or** button or the **and** button.

   f. **Optional:** Build another set of conditions by selecting the **Add condition set** button and repeating steps a-e.

Dynamic behavior example



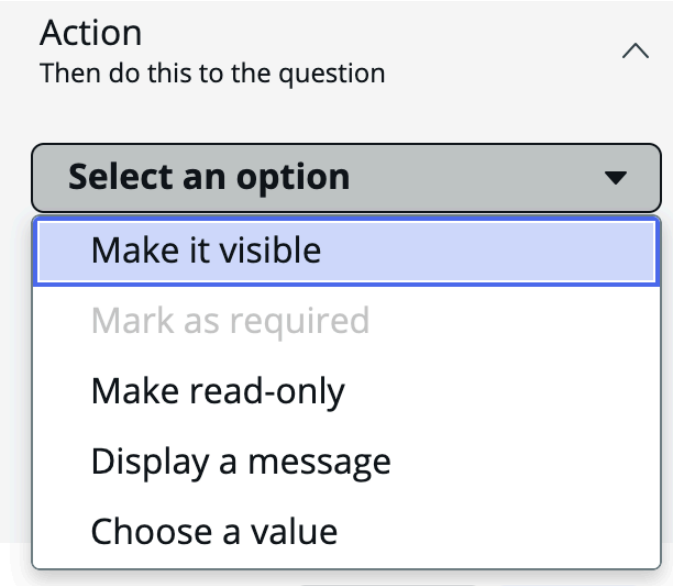
8. Define how the form changes in response to the conditions that you defined in the Action section of the **Behaviors** panel.

   a. Choose what the form does when the triggering conditions are met in the **Select an option** field.
   Depending on the conditions you created, the following actions may be available:

   - **Make it visible**
   - **Mark as required**
   - **Make read-only**
   - **Display a message**
   - **Choose a value**

   You can add multiple actions, but you can add only one of each type of action for each set of dynamic behavior.

Dynamic behavior actions



b. Choose what happens for each action that you select.
For example, if you chose to **Display a message**, you can make it an **Info**, **Warning**, or **Error** message, and then you must enter the message that appears.

9. Select **Save**.

Result

The question appears with a dynamic icon ( ) to indicate that it has dynamic behavior.

The dynamic behavior appears in a card on the **Behavior** tab of the settings panel. You can select the card at any time to view or edit its details, or select the **Add behavior** button to add another dynamic behavior.

Development states of a form in Creator Studio

Think of your app as an online store that contains a catalog of items. To request something, a person needs to fill out a form. It's your job to create and customize forms for every item in your catalog using Creator Studio.

Summary:
Forms have two states:

Draft

The state of a form when you're editing it. It can be a new form or a copy of a published form. Forms in this state do not appear in your catalog.

Published

The state of a fully-designed form that appears in your catalog.

You want to publish forms only when they're ready—not some half-baked version you're working on. That's why forms have states.

Draft state for forms

Forms in the Draft state don't appear in your catalog of items. That means you can work on them without affecting your online catalog.

The forms you edit can be new, or copies of forms that have already been published. You can revise forms as much as you like in this state, but the revisions are kept behind the scenes until you publish the form. So, no worries about leaving a form unfinished when it's in Draft state!

Published state for forms

When you finish editing a form, you mark it as ready, which puts it in the Published state. This version of the form then appears in its specified catalog of items.

> **Note:**
> The published form appears only on the non-production instance where you're developing your app. You must request that your admin deploy the app to production for the form to appear in the catalog on the production instance.

If you edit a form that's been published, Creator Studio creates a copy of the form, and puts the copy in the Draft state so you can work on it. The published version of the form remains available in your online catalog until you submit the revised copy of the form, which will replace the published form.

Add more forms to an app in Creator Studio

Add as many forms as you need to create catalog items for your app. For example, you could have an office booking app with three different forms: one for requesting parking spaces, one for requesting a desk, and one for reserving conference rooms.

Before you begin

To add forms to an app, you must be given permission to work on the app.

About this task

> **Note:**
> An IT fulfillment app could already have one form for requesting hardware accessories (such as a mouse or headphones), and you can add a second form for requesting software applications.

Procedure

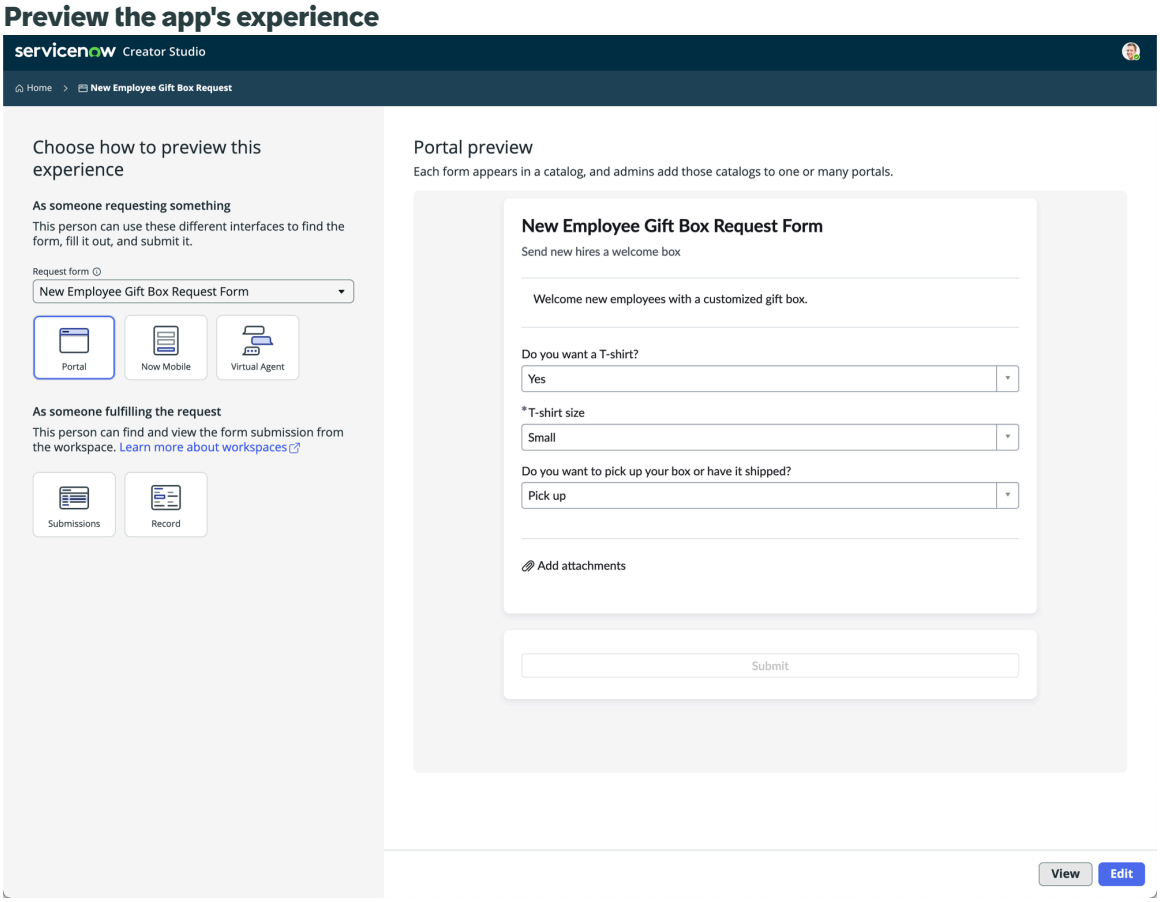
1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that you want to add more forms to.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

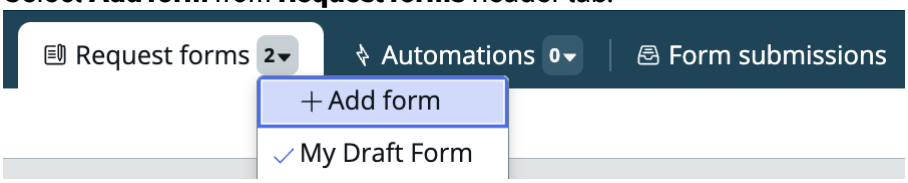
     > **Note:**
     > Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select **Add form** from **Request forms** header tab.



5. Select the catalog template that you want to use to build the form from the list.
6. Select the **Apply this template** button.
   You can select the **Preview** button to see a quick mockup of what the form will look like without any modifications.
7. Add a form to a catalog to specify its business area.

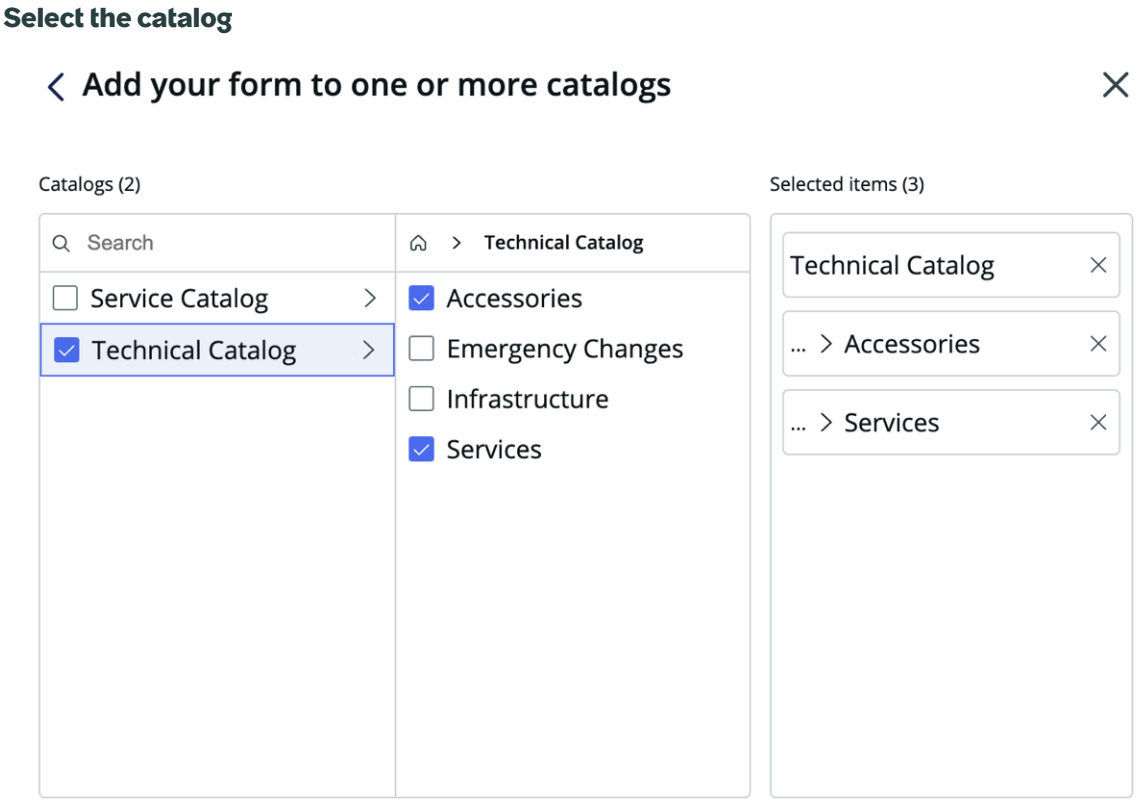
> **Note:**
> You can skip this step for now and revisit it later when you have a better idea of where it fits.

The available catalogs are configured by your admin, contact them if you don't see the one you want.

a. Select the **Edit button** (Edit) for the Catalogs and categories card.

b. Select the catalog that represents the business area the app will use.

For example, choose a service catalog that contains software and laptop cables. Expand the carat for each catalog to see its sub-catalogs.



c. Select as many items in the catalogs as you need.

d. Select **Apply**.

You can edit any of the app's basic settings any time after you finish creating the app. For more information, see Creator Studio form settings.

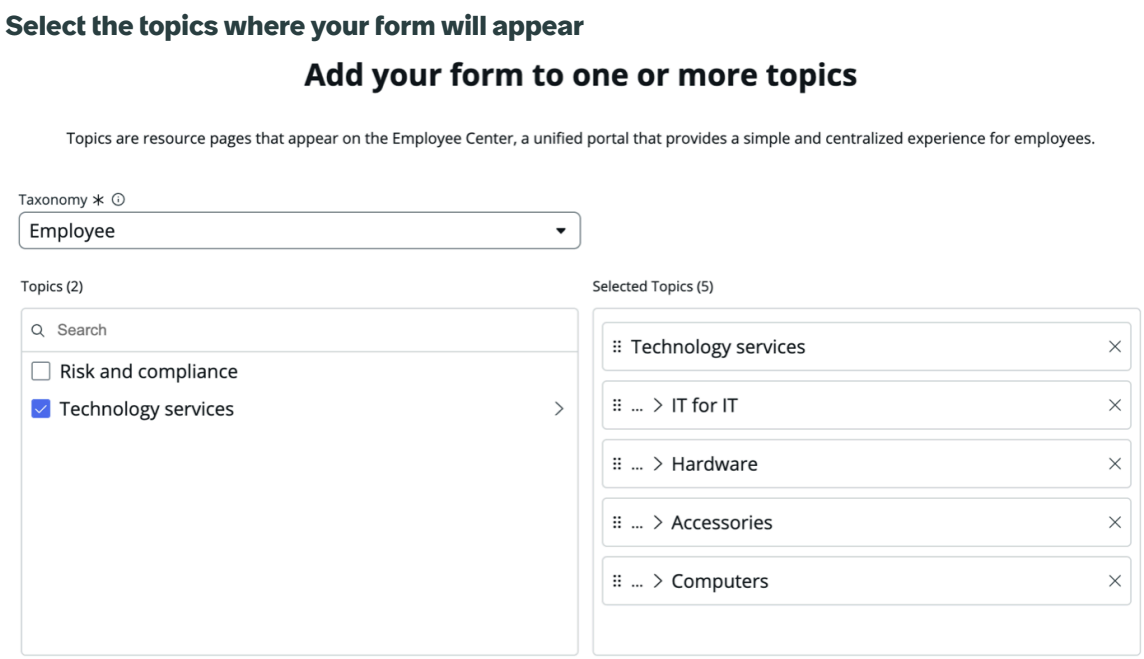
8. Next, choose one or more topics to specify where the form will appear.
Find out more about topics in Associate a catalog item with a taxonomy topic in Employee Center, and more about taxonomy, which is a categorization method, in Unified Taxonomy for Employee Center.

> **Note:**
> You can skip this step for now and revisit it later when you have a better idea of where it fits.

a. Select the **Edit button** (Edit) for the Topics card.

b. Choose the **Taxonomy** page where you want the form to appear, such as **Employee**.
Taxonomy is a method of categorization that Employee Center uses.

c. Select the topic(s) that represent the Employee Center areas where you want the form to appear.
For example, choose a topic that contains technology services, and expand its carat to see each of its sub-topics.



d. Select the topics where you want the form to appear, as many as you need.

e. Select the **Apply** button to save your changes.

9. Select the **Save and continue** button.

Result

Congrats:
Hooray! You've added another form to the app.

What to do next

Next, you can edit the form to populate its questions and customize its layout. For more information, see Customize your form for an app in Creator Studio.

Edit the settings for a form in Creator Studio

Edit form settings if you need to change its basic attributes, such as its associated image or attachments are allowed.

Before you begin

To edit the settings for a form, you must be given permission to work on the app.

About this task

Note:
After you create the app, its **Template** can't be edited. If you want to change it, create a new form using a different catalog template. The **Record submission table** is automatically generated, and can't be changed.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.

2. Open the application that contains the form you want to edit.

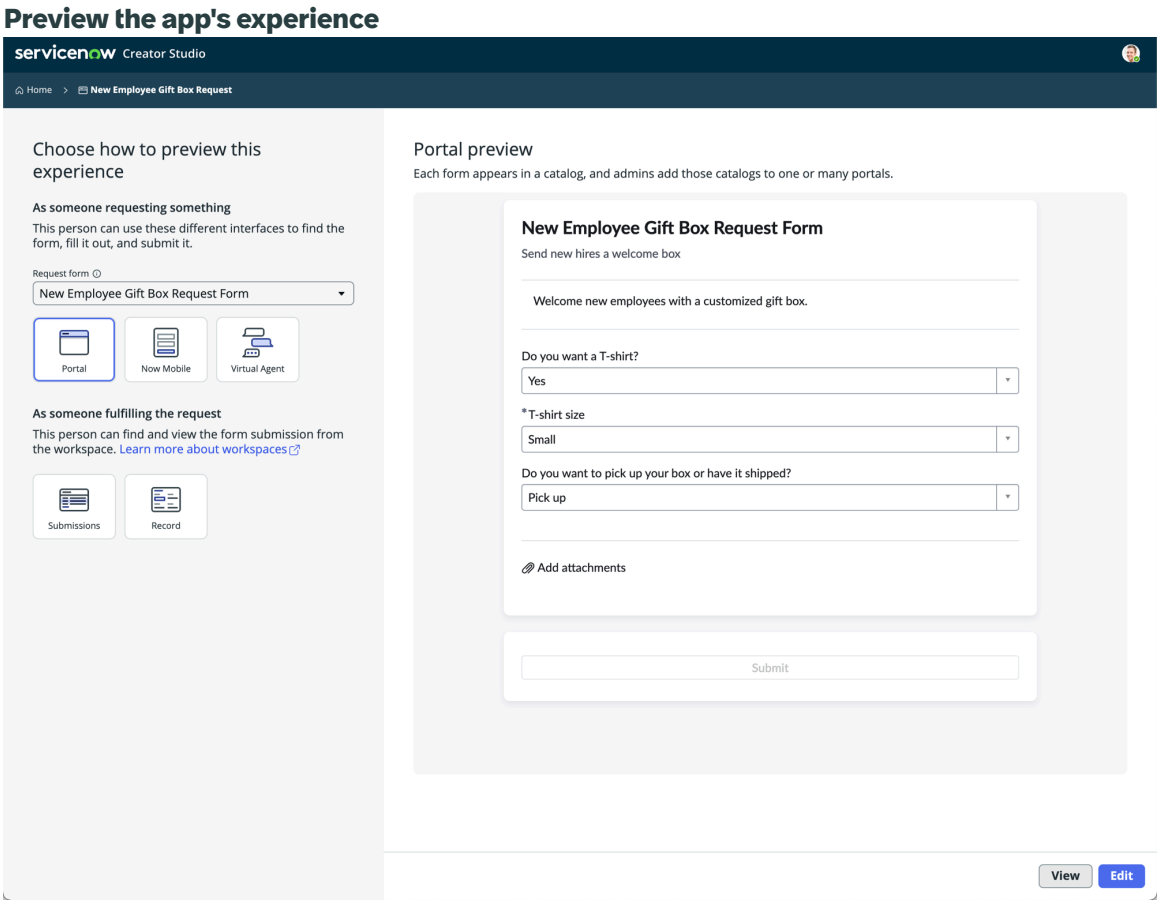
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

- **Portal** (see a preview of how it'll appear on a desktop website)
- **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
- **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

> **Note:**
> Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.



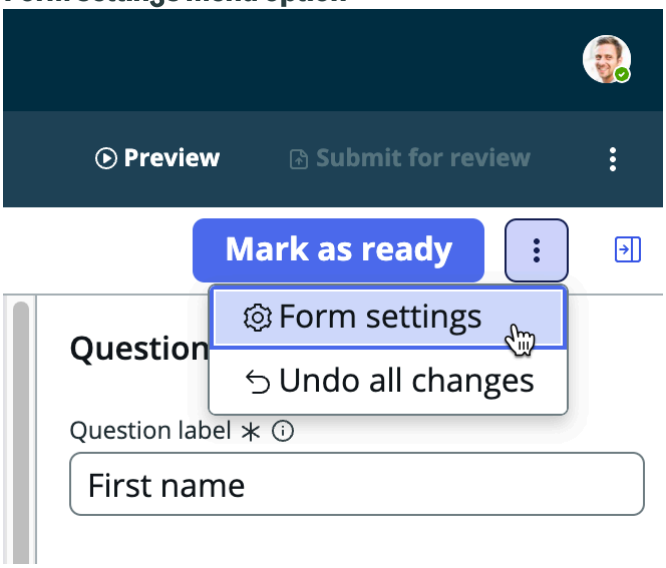
You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Check that you're editing the correct form in your app by selecting it from the **Request forms** tab.

5. Select the more actions icon ⋯.

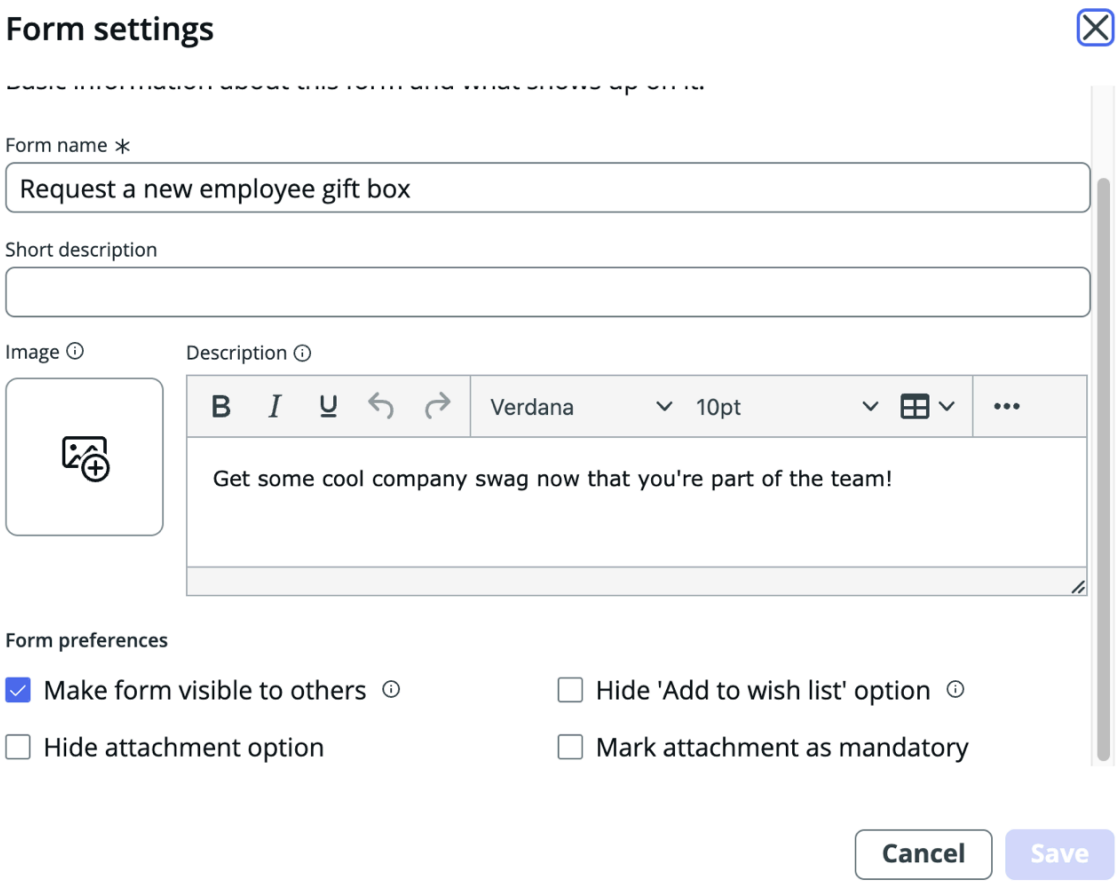
6. Select **Form settings**.

Form settings menu option



7. Update settings on the **General** tab.
For details on specific form settings, see Creator Studio form settings.

Form settings modal



8. Select the **Location** tab to update where the form appears in a catalog, how it's categorized, and which topics it appears for.

a. Select the edit icon (Edit) for the **Catalogs and categories** card.

b. Select the catalog that represents the business area the app will use.
For example, you could select a service catalog that contains software and laptop cables for an IT fulfillment app.

c. Expand the carat for each catalog to see its sub-catalogs.

d. Select items in the catalogs, as many as you need.

e. Select the **Apply** button to save your changes.

f. Select the edit icon (Edit) for the **Topics** card.

g. Choose the **Taxonomy** page where you want the form to appear, such as **Employee**.
Taxonomy is a method of categorization that Employee Center uses.

h. Select the topic(s) that represent the Employee Center areas where you want the form to appear.
For example, choose a topic that contains technology services, and expand its carat to see each of its sub-topics. Find out more about topics in Associate a catalog item with a taxonomy topic in Employee Center, and more about taxonomy, which is a categorization method, in Unified Taxonomy for Employee Center.

i. Select the topics where you want the form to appear, as many as you need.

j. Select the **Apply** button to save your changes.

9. Select the **Access** tab to change which roles and groups can view and submit the form.

a. Select the edit icon (Edit) to edit users that the form should be **Available for**.

b. Select the roles and groups that should have access to the form.

c. Select the **Apply** button to save your changes.

d. Select the edit icon (Edit) to edit users that the form should be **Not available for**.

e. Select the roles and groups that shouldn't have access to the form.
Work with your admin to restrict or provide access to the roles and groups for this setting in non-production and production environments. For more information, see Administering user access for deployed Creator Studio apps.

f. Select the **Apply** button to save your changes.

10. Select **Save all settings**.

Publish a form for your app in Creator Studio

Publishing forms once they're ready makes them available as catalog items in the production instance for published apps.

Before you begin
To publish a form, you must be given permission to work on the app.

About this task
After you publish a form, it's available in the specified catalog on the instance you're working on. However, you still need to deploy your app to the production instance for users to access the form. For more information, see Deploying your Creator Studio app.

Procedure
1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the application that contains the form you want to publish.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

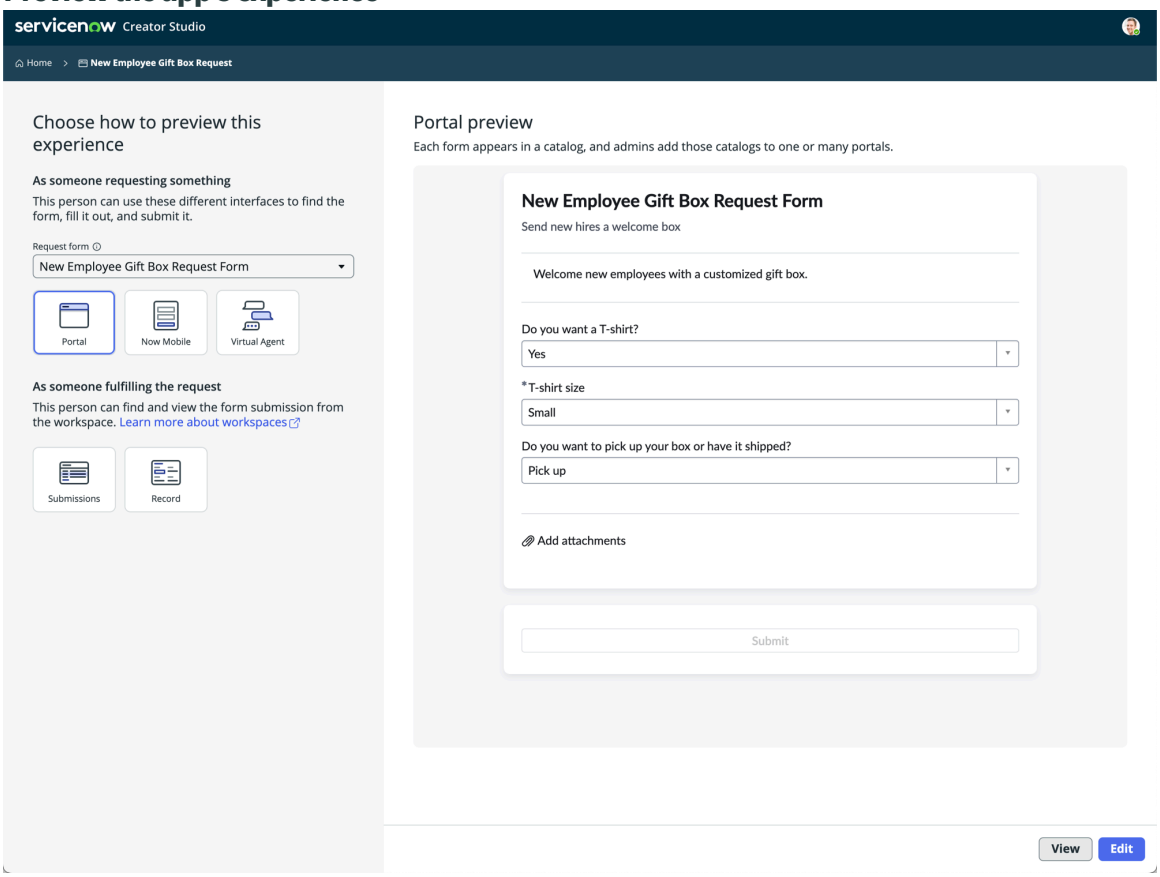
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

   Note:
   Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

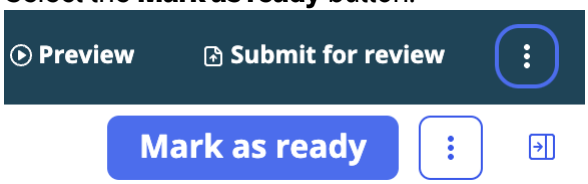
   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

Preview the app's experience



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Check that you're working on the correct form in your app by selecting it from the **Request forms** tab.
5. Select the **Mark as ready** button.



Result

The catalog item for the form is ready to be deployed with your app. Once the app is deployed, the form is available in the associated service catalog. If the form's app hasn't been deployed, you need to deploy it.

Hide a form from use in the ServiceNow AI Platform in Creator Studio

Hiding a catalog item for your app's form effectively makes the form inactive. Hidden forms are unavailable in both the app and the catalog it belongs to.

Before you begin

To hide a form, you must be given permission to work on the app.

About this task

Creator Studio doesn't let you delete forms. Instead, hide forms to ensure that workflows stay intact.

You can hide only forms that have already been published. For more information, see Deploying your Creator Studio app.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the form you want to hide.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

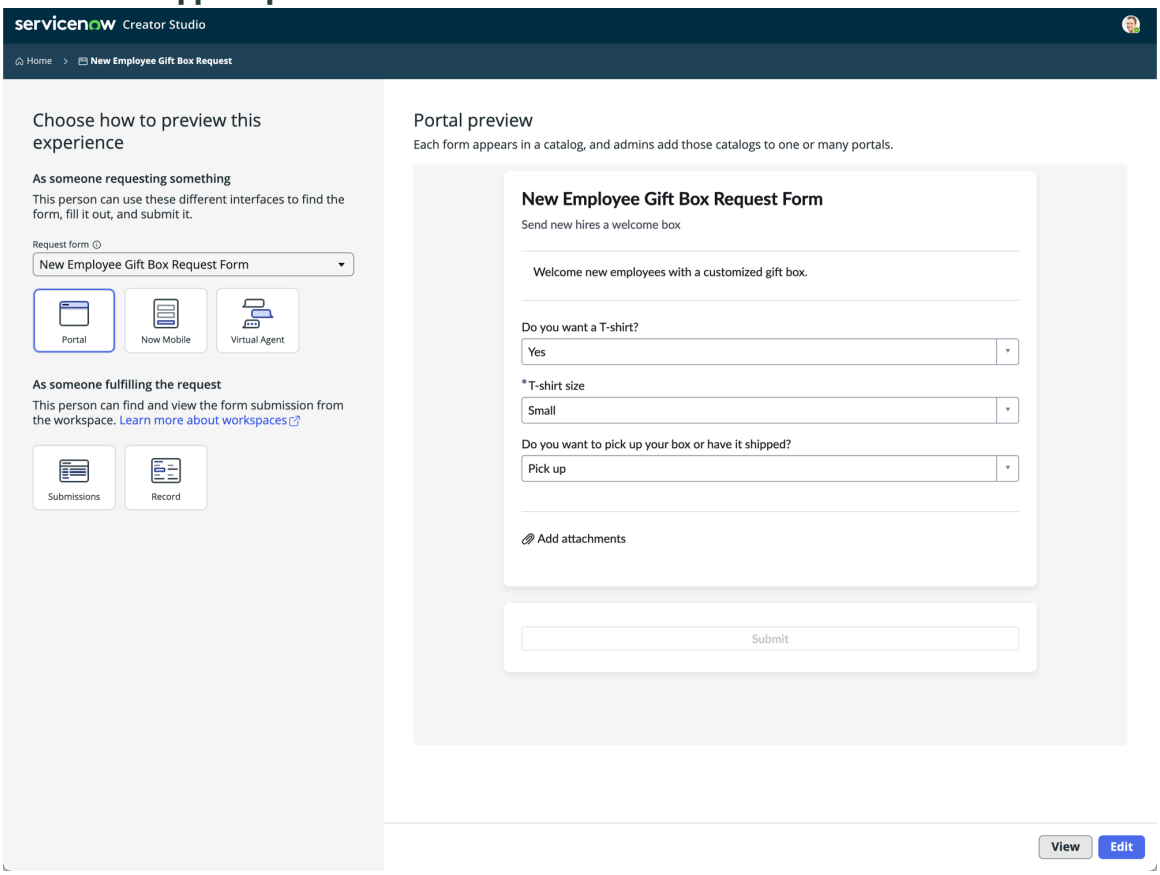
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

   Note:
   Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

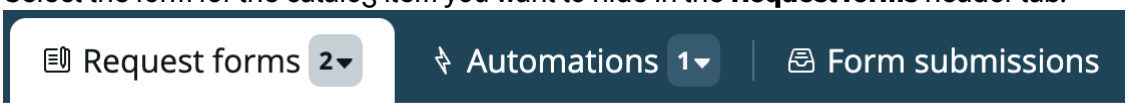
   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

Preview the app's experience



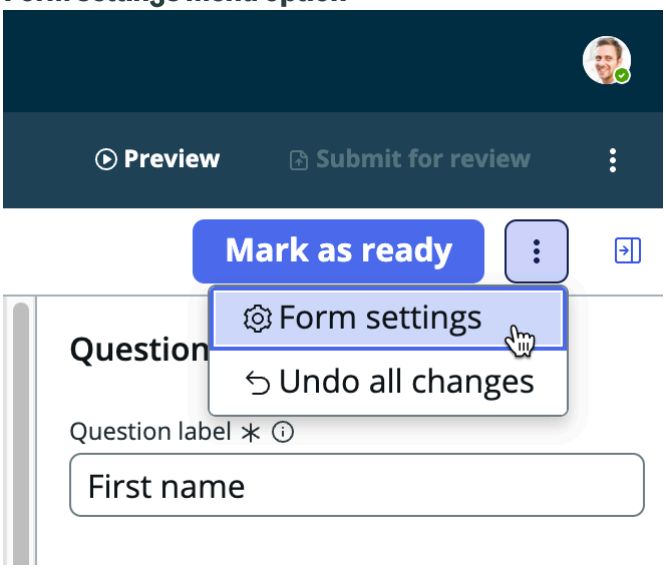
You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the form for the catalog item you want to hide in the **Request forms** header tab.



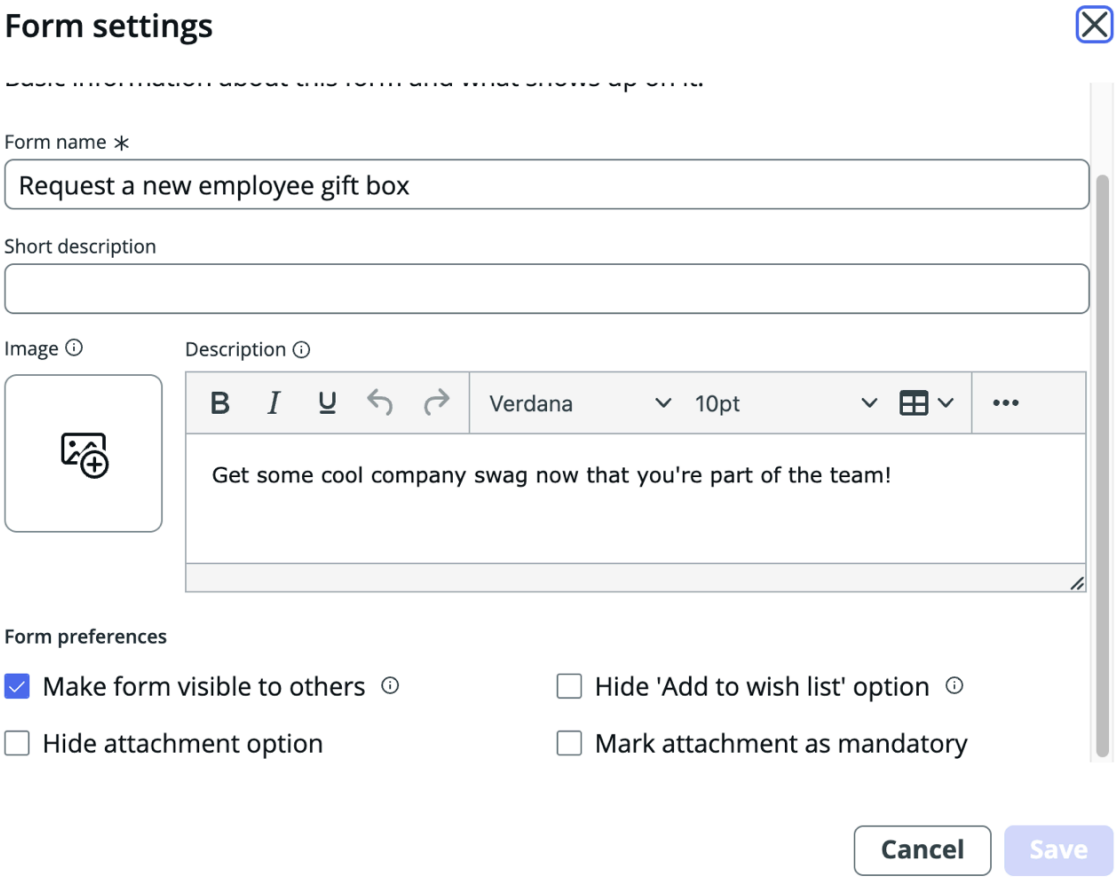
5. Select the more actions icon ( ).
6. Select **Form settings**.

Form settings menu option



7. De-select the **Make form visible to others** option.

Form settings modal



8. Select **Save**.

What to do next

You must re-deploy the app to hide the inactive form (catalog item) on the production instance.

Working with automation in Creator Studio

How do you simplify and streamline tasks for fulfillers? Use automation! For example, you can have a task assigned automatically to someone based on what type of request it is. We'll use playbooks to accomplish adding this automation to your app.

Summary:
After reading this section, you'll understand how to use automation to do the following things:

- Use playbooks to guide agents and fulfillers through processes from start to finish, improving the customer experience and the way tasks are carried out.
- Create a consistent process for your tasks or activities to be carried out.
- Consolidate business decisions and processes across the organization.

Note:
Because playbooks are based on forms, you should finish building out your app's forms before you start working on automation.

Playbooks are a set of activities that occur in order based on a trigger. Each activity is more like a step in a chain of steps. You can add multiple playbooks to an app if you need to.

Key terms:

Activity

Activities in a playbook complete automated tasks, such as obtaining approvals, assigning records, and sending email.

Automation

Tasks or requests moving through a workflow without manual intervention.

Playbook

A set of actions based on an initiating action called a trigger.

Record

A task or request that must be acted on. All rows in ServiceNow tables are individual records. Records have different states depending on what actions have been taken.

Trigger

The event that starts making the playbook run.

Workflow

The path a record takes from creation to completion.

What is automation?

Automation in Creator Studio enables you to create simplified playbooks, which are basic, automatic workflows. Simple playbooks consist of a trigger (such as a record changing states, or a user choosing a specific answer for a question) and an activity (such as the record being assigned to a new person). Playbooks enable you to add the following:

- Activities to your automation to define the workflow's path.
- Decisions to create different paths, or branches, depending on the decision's outcome.
- Custom activities defined by your admin or ServiceNow AI Platform owner.

Playbooks take the guess work out of assigning tasks and help users focus on the tasks and information that matter to their job. A well-designed playbook can do these things:

- Automatically start or trigger changes for different types of records, such as task assignments.
- Reuse activities and business decisions from other workflows you've already created.
- Clearly show the next steps that fulfillers must take to move to the next step of completing the record's life cycle.

There are several types of activities that you can add to playbooks that, when paired with decisions, remove the manual thinking behind what to do next when addressing tasks. Depending on your configuration, some of the activities you can add include:

- Request approval for a task
- Assign the task to someone
- Create a new task as a next step
- Send an email
- Choose a custom action configured by your admin

Note:
In addition to creating custom activities, your admin can also hide standard activities, so you may not see all of the above options.

You can also use a placeholder activity for more complex activities that must be configured outside of Creator Studio, or for another activity to be swapped in later. Complex activities are configured in a different tool called Workflow Studio.

What goes into building automation?

Automations in Creator Studio include the following parts.

Automation parts

Concept	Description
Playbook	An automatic workflow that guides users through tasks based on the record's status changes. Find out more about creating playbooks in Add an automated playbook to an app in Creator Studio.
Trigger	The event that specifies when the playbook should start running, such as when a form is submitted or updated to a specific status. Check out Edit the trigger for a playbook in Creator Studio for more information.
Activity	The thing that happens during the playbook, such as assigning a task or requiring approval.

Automation parts (continued)

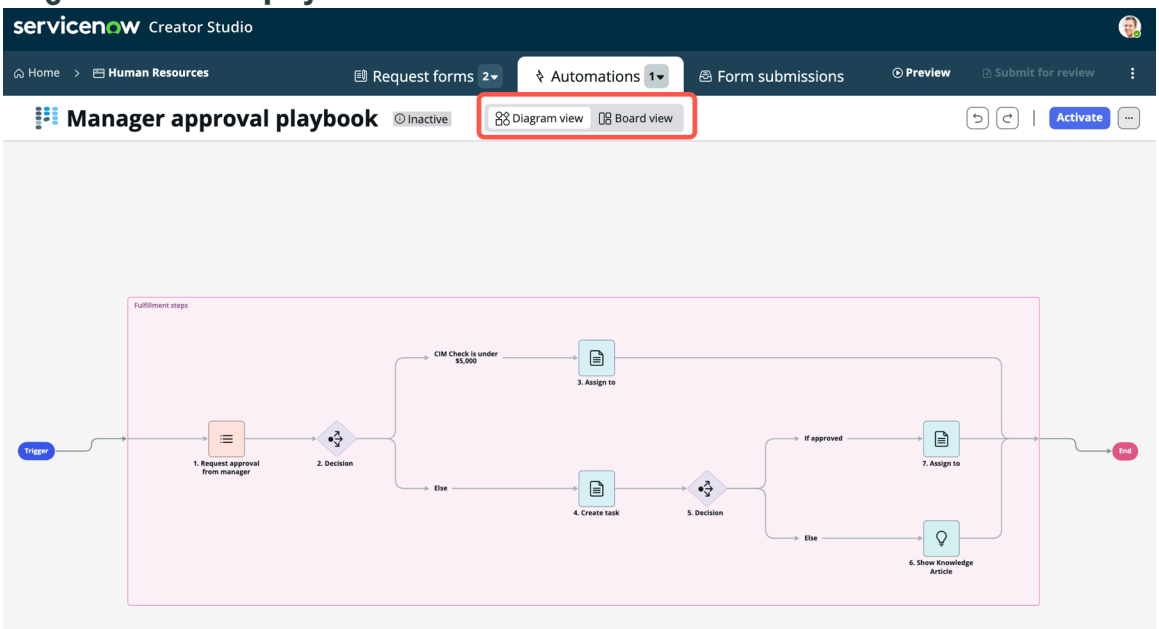
Concept	Description
	Note: If you're already familiar with optional activities in playbooks, that's great! However, to keep automations streamlined and simple, Creator Studio doesn't support optional activities. If you must create optional activities, you can add a placeholder activity in Creator Studio and then edit the placeholder in Workflow Studio or swap it out for another activity later in Creator Studio. Additionally, admins can create custom activities for you to include in a playbook. Learn how to add activities in Add activities to an app's playbook in Creator Studio.
Decision	A series of inputs, conditions, and results that create different branches, or conditional paths that users follow based on the decision's outcome. Get the scoop on adding decisions in Add a decision to an app's playbook in Creator Studio.
Branch	A distinct path from a decision's outcome, or what happens for each of the decision's possible choices. For more information, see Add a decision to an app's playbook in Creator Studio.
Parallel process	Branches for activities and stages that run in parallel to another branch of activities. Learn about parallel processes in Add a parallel process to an app's playbook in Creator Studio.
Connectors	Lines that connect activities and decisions in a playbook that enable you to add more activities, decisions, and parallel processes. You can drag connectors between activities to create parallel activities.

Board and Diagram views of a playbook

Creator Studio offers two ways of working with playbooks: the Board view and the Diagram view.

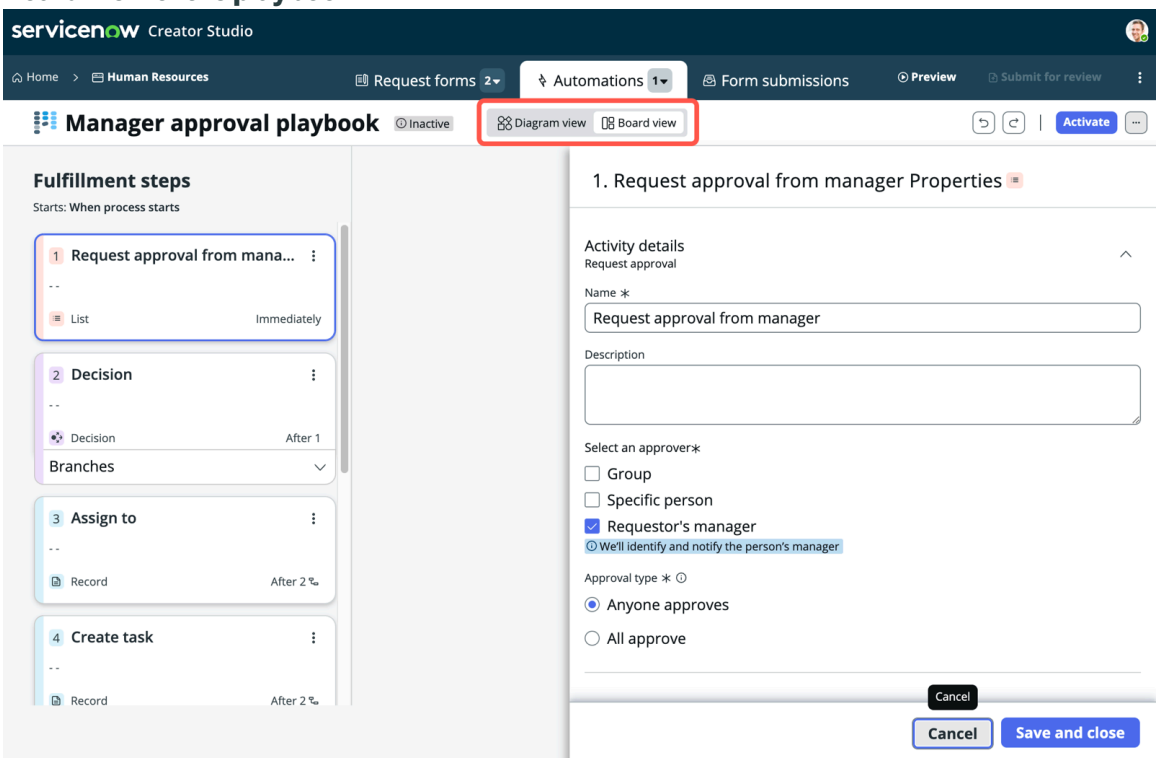
The Diagram view is a visually intuitive diagram that shows you the order that activities occur in the playbook. When you select an activity, such as the trigger or a decision, a modal pops up where you can edit its details. You must use the Diagram view to add decisions.

Diagram view of the playbook



The Board view is a more traditional layout, which shows each activity as a card in the navigation panel. When you select an activity, for example a task, its editable details appear in the canvas for you to work with them. You must use the Board view to rearrange activities.

Board view of the playbook



This help document will primarily give instructions using the Diagram view, with alternate directions on how to complete automation tasks in the Board view.

Where do playbooks run in the ServiceNow AI Platform?

Users work on requests in the Request App Workspace on the ServiceNow AI Platform, which lists a record for each request. The playbooks run on the specified records in the app's workspace category.

Specifically, the **Catalog tasks** tab of the record details walks users through the activities they must do to complete the task and close the record.

Workflow for creating automations in Creator Studio

Provide your organization's fulfillers with guidance when addressing records generated from the app by adding automated playbooks to it.

Before you begin

Note:
Because playbooks are based on forms, you should finish building out your app's forms before you start working on automation.

To work with automations, you must be given permission to work on the app.

Procedure

1. Create an app.
   If you have an existing playbook that's close to what you want to create, you can duplicate and edit it, which may be faster than creating an entirely new playbook. For more information, see Copy an automated playbook in Creator Studio.
2. Add an automated playbook to the app.
   For more information, see Add an automated playbook to an app in Creator Studio.
3. Add activities to the playbook.
   For more information, see Add activities to an app's playbook in Creator Studio.
4. Create different branches of the playbook by adding decisions to it.
   For more information, see Add a decision to an app's playbook in Creator Studio.
5. **Optional:** Create any parallel processes to run at the same time in the playbook.
   For more information, see Add a parallel process to an app's playbook in Creator Studio.
6. Make any additional changes to the playbook's trigger.
   For more information, see Edit the trigger for a playbook in Creator Studio.
7. Make any additional changes to the playbook settings.
   For more information, see Edit a playbook in Creator Studio.
8. Activate the playbook when it's ready.
   For more information, see Activate a playbook in Creator Studio.

Add an automated playbook to an app in Creator Studio

Create an automated playbook, which is a process that runs whenever a record is created or updated by the app's specified form.

Before you begin

Playbooks run on the records generated by catalog items you create for the app, so you must add forms to the app before you can add automation. For more information, see Working with forms in Creator Studio.

To add a playbook, you must be given permission to work on the app.

About this task

Note:
You can build a travel request app where people can ask for permission for different types of travel, which automatically routes the request to a manager for approval.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that you want to add automation to.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

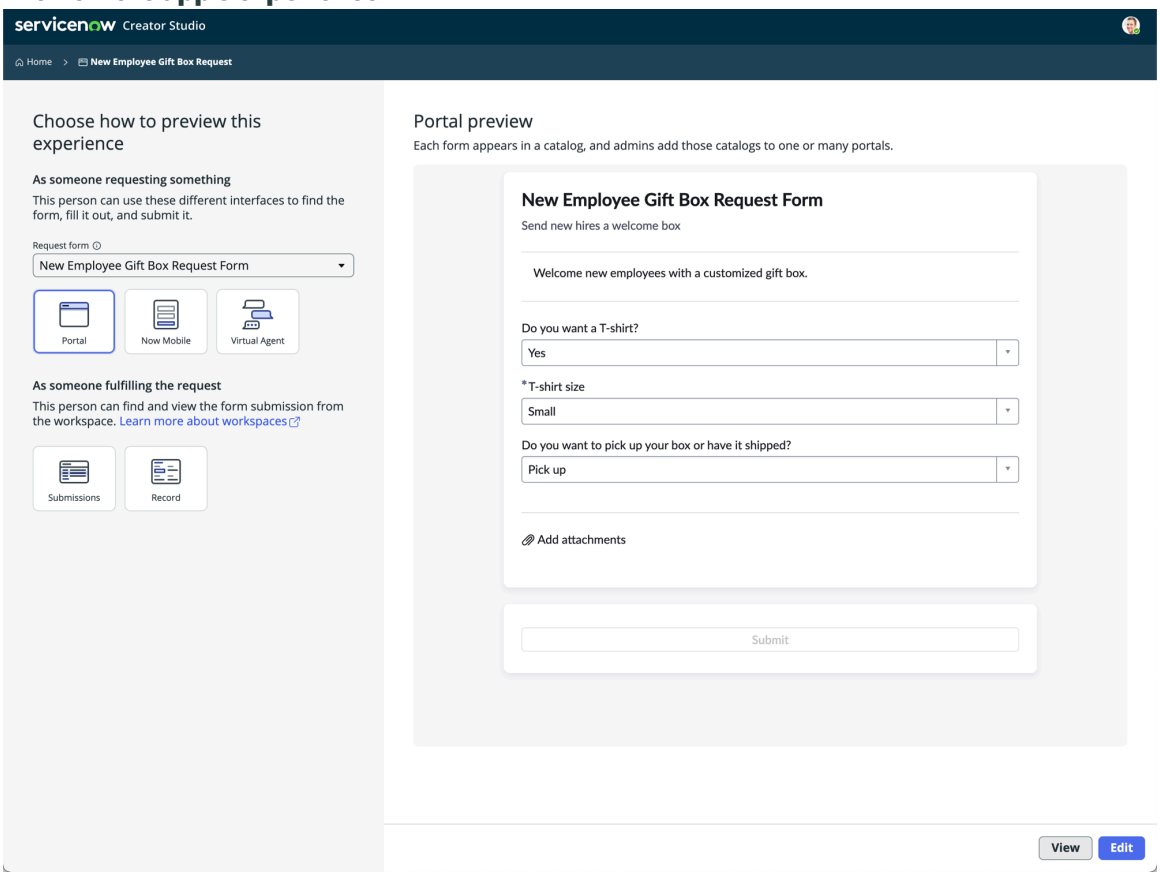
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

   Note:
   Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

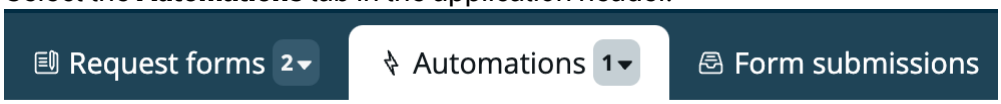
   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

Preview the app's experience



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Automations** tab in the application header.

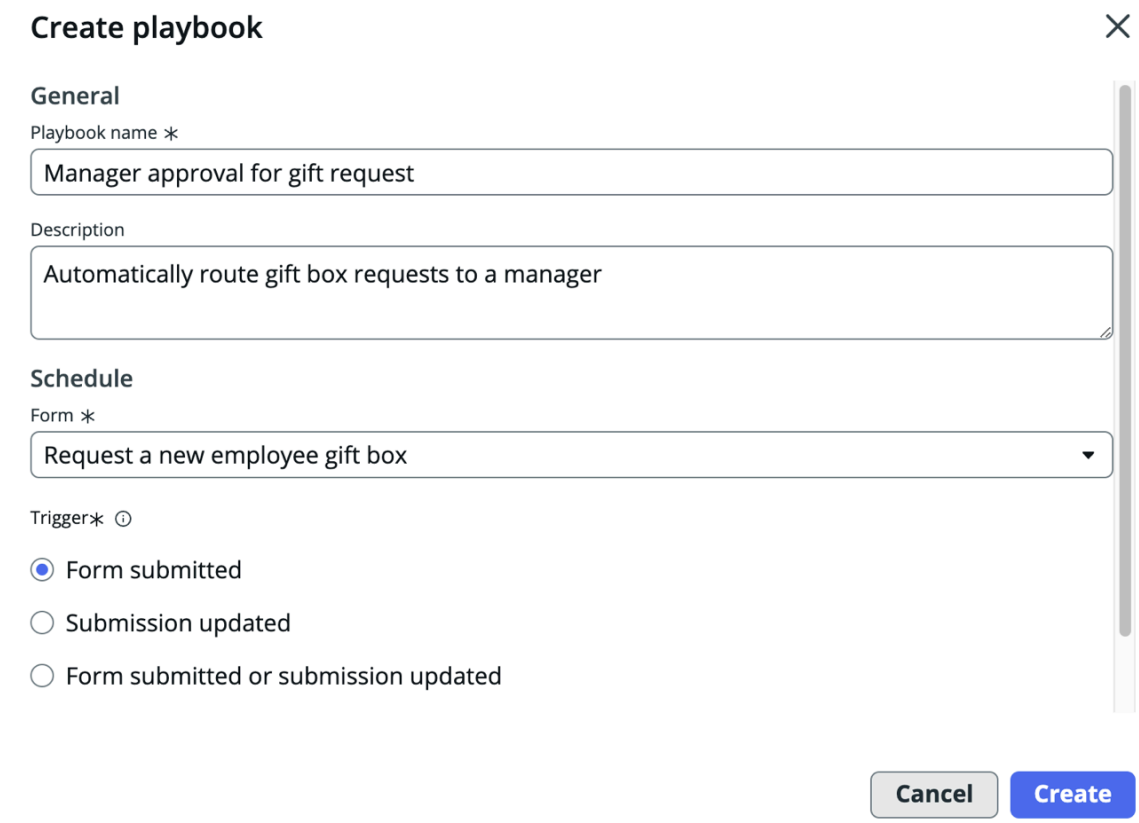


5. Select the **Create a playbook** button.
   If an automation already exists, select the **Automations** tab and then select **Add new playbook**.
6. Specify the playbook's General attributes in the Create playbook modal.

General playbook definition fields

Field	Description
Playbook name	Descriptive name for the playbook you're creating.
Description	Brief explanation of what the playbook does, for example, the end goal for the record type.

Create a playbook



7. Specify the Schedule for the playbook.

   a. Select the **Form** whose catalog item generates the record type that you want the playbook to run on.
   The table for the form you choose is always the app's task table, which is specified when the app is created.

   b. Select the type of **Trigger** that initiates the playbook.

Playbook trigger options

Trigger	Description
Form submitted	Start running the playbook when a user submits the form you chose.
Form updated	Start running the playbook when a user updates the form you chose.
Form submitted or updated	Start running the playbook when a user submits or updates the form you chose.

Note:
You can't change an playbook's trigger type after you finish creating the playbook. Instead, create a new playbook with a different trigger.

However, you can edit the trigger condition, such as making the playbook run conditionally based on a specific answer to a question. For more information, see Edit the trigger for a playbook in Creator Studio.

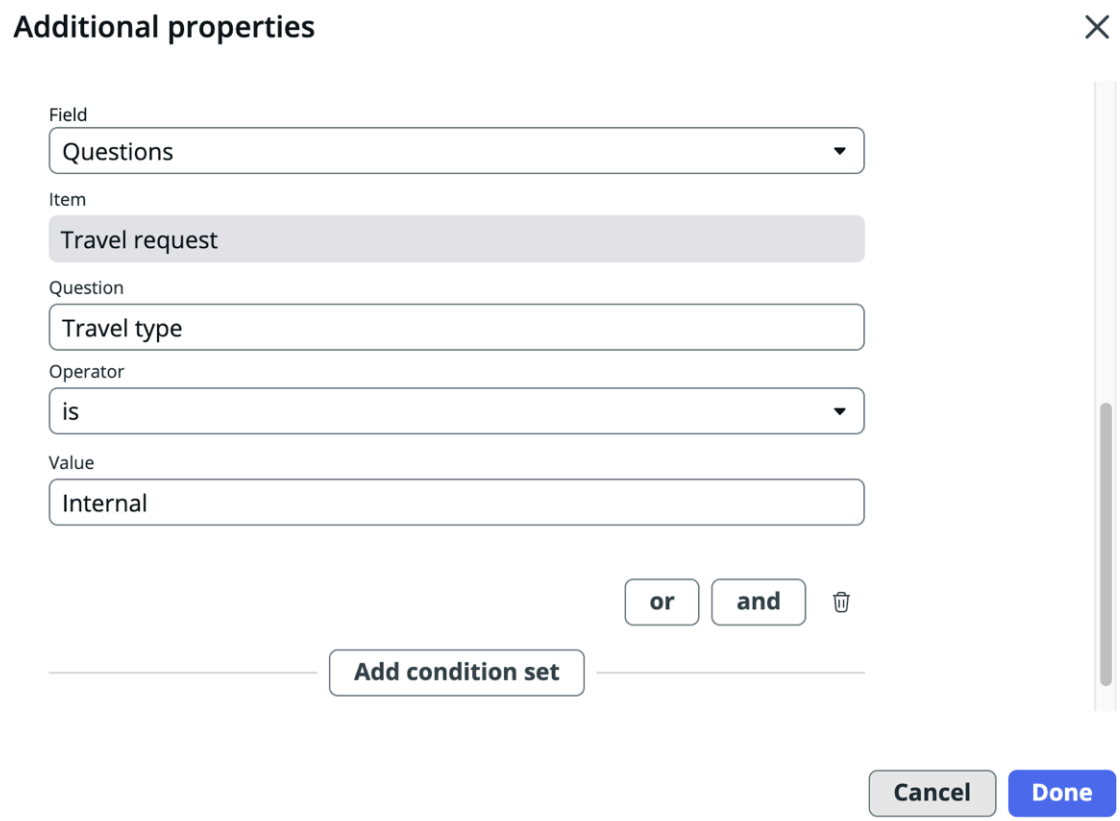
c. If you chose a trigger that includes a form being updated, specify how often that app should **Run your playbook**.
The options are:

- **Once**
- **For each unique change**
- **Only if not currently running**
- **For every update**

d. Specify the conditions that must be met for the playbook to begin running by selecting **Add condition set**.

- If you want to trigger the playbook based on the value of a column in a table, select the **Field** that you want to be the trigger, as well as its condition **Operator** and the specific trigger **Value**. For example, when a **Start date** is **after** the **Date** needed.
- If you want to trigger the playbook based on the response from a form, select **Questions** as the trigger **Field**. Then select the question you want in the **Question** field, the condition **Operator** and the answer's **Value**.

Question answer as trigger for an automation



Add as many conditions as you need. For more information, see Create a condition statement using the condition builder.

8. Select **Save**.

Result

Congrats:
Congratulations, you've created an automation!

What to do next

Next, you need to add activities to the playbook. For more information, see Add activities to an app's playbook in Creator Studio.

Copy an automated playbook in Creator Studio

Duplicate an existing playbook and make small changes to it, which may be faster than creating an entirely new automated playbook.

Before you begin

Because playbooks are based on forms, you should finish building out your app's forms before you start working on automation.

To copy a playbook, you must be given permission to work on the app.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app with the playbook you want to copy.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

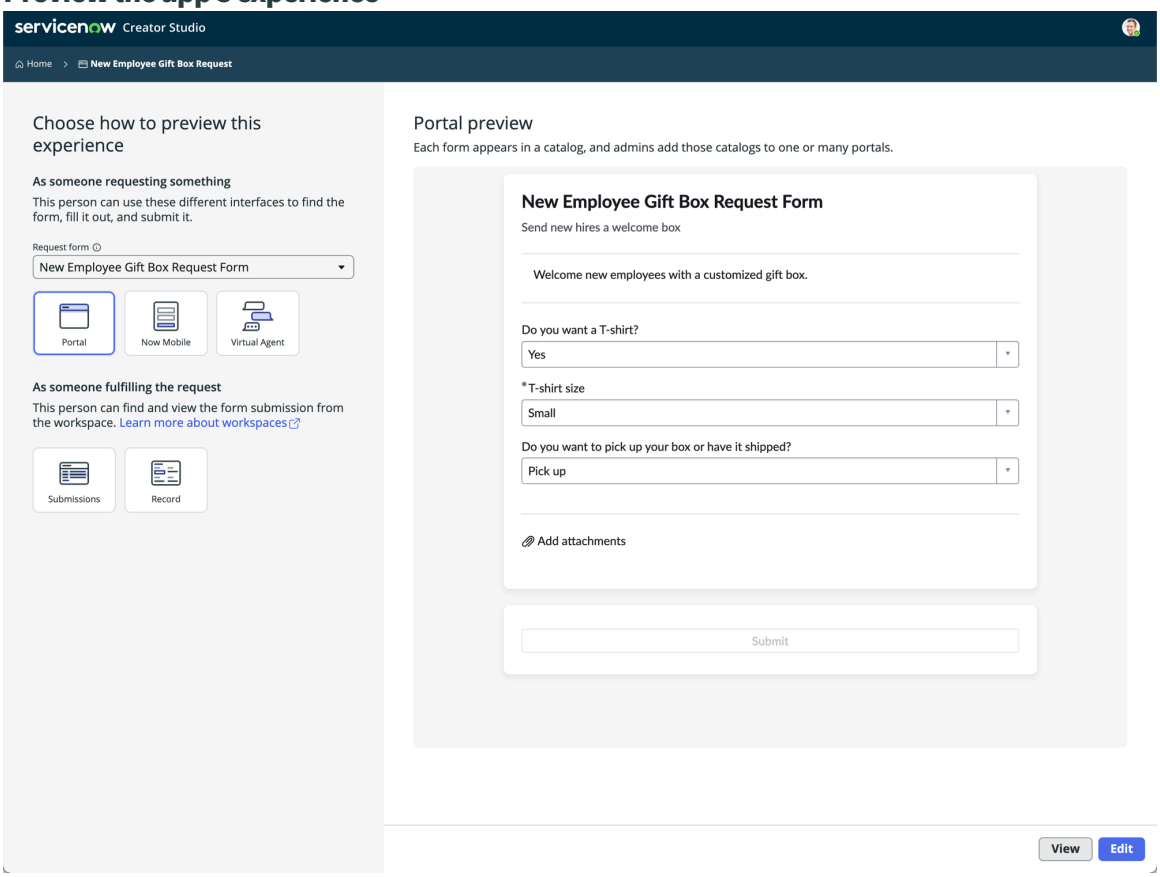
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

   Note:
   Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

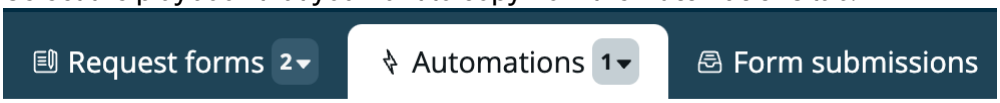
   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

Preview the app's experience



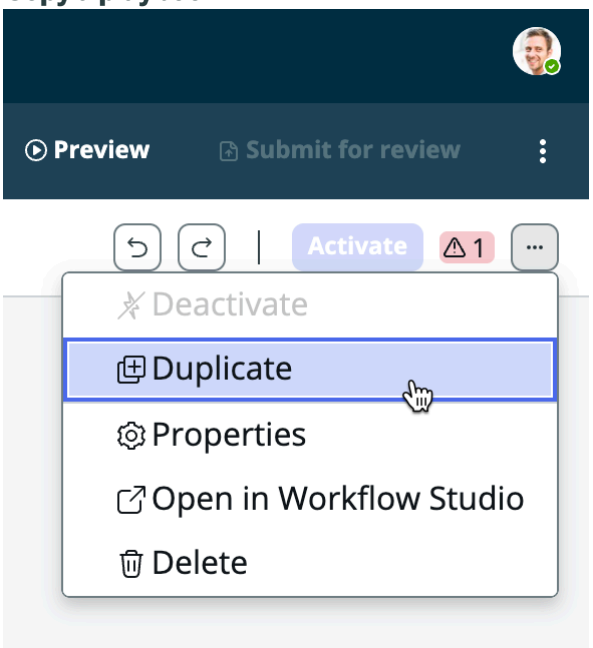
You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Automations** tab in the application header.
5. Select the playbook that you want to copy from the **Automations** tab.



6. Select the more actions menu icon ( ).
7. Select **Duplicate**.

Copy a playbook



8. Choose the **Duplicate** button in the confirmation modal.

Add activities to an app's playbook in Creator Studio

Add activities to an app's playbook to specify what the automation does to the designated record type. An activity defines what actually happens when a playbook executes.

Before you begin

To add activities to a playbook, you must be given permission to work on the app.

About this task

The activities that you can add to a playbook are limited by your administrator to ensure you use the correct ones for your organization. Your admin may have created some custom activities for you to choose from, or hidden some standard activities.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the playbook you want to add an activity to.

3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

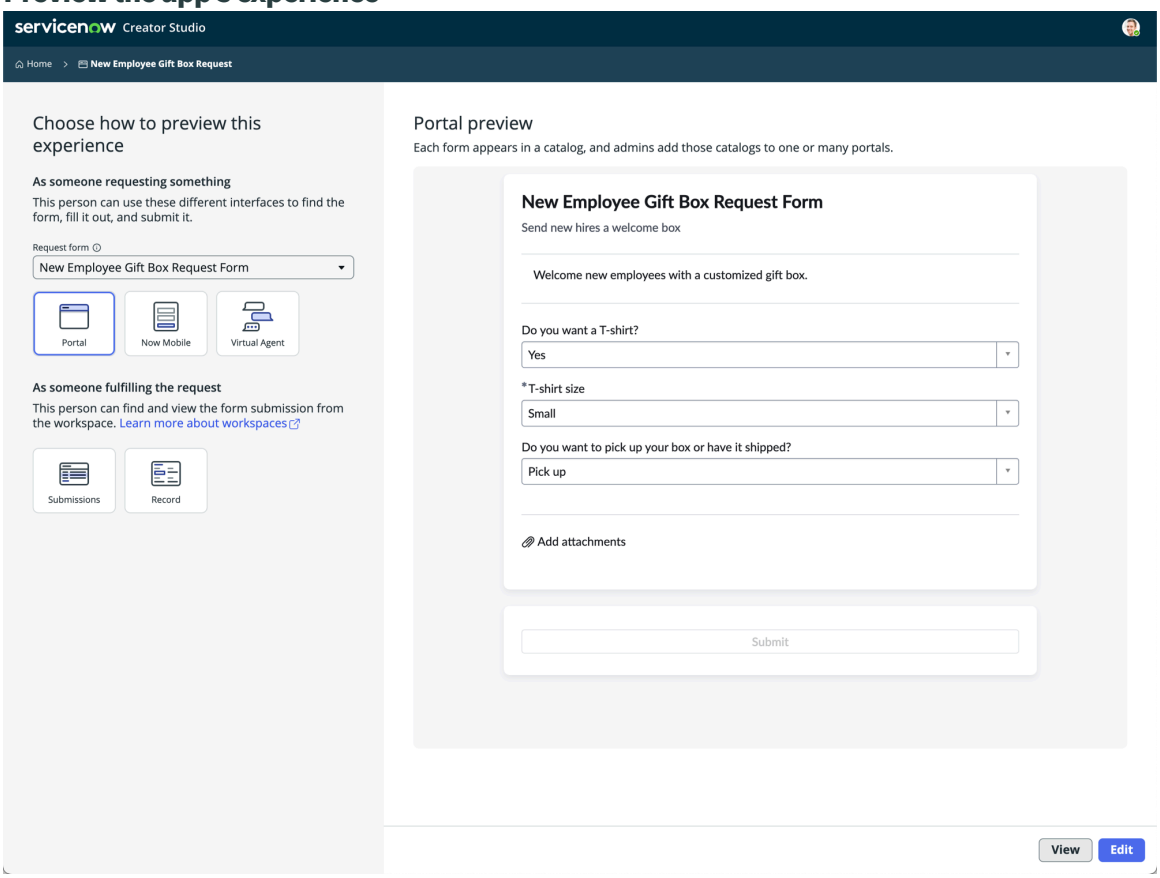
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

   > **Note:**
   > Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

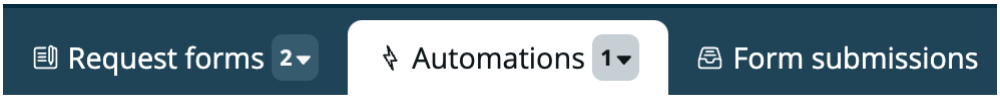
   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

   Preview the app's experience

   You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Automations** tab in the application header.
5. Check that you're editing the correct playbook in your app by selecting it from the **Automations** tab.



6. Select the add icon on the connector where you want to add an activity and choose the square **Add an activity** icon ( ) in the menu that pops up.

How to add an activity in Board and Diagram views

View	Steps
Diagram	**a.** Select the **+** icon in the stage. **b.** In the mini-picker, select the square icon ( ) to add an activity.
Board	Select **+ Add activity**.

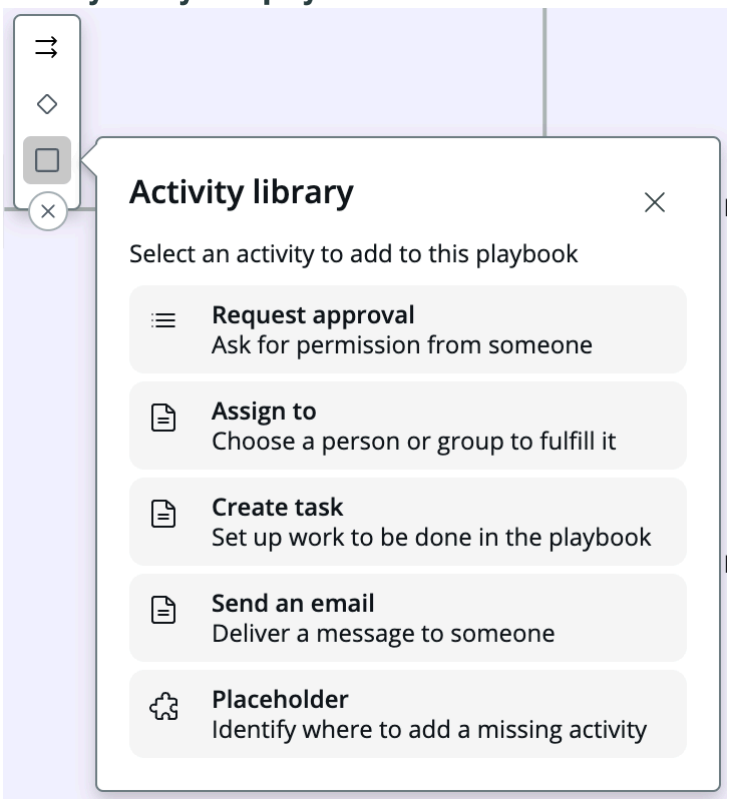
7. Choose the type of activity that you want from the Activity library pop-up.

Note:
In addition to the following standard activities, you may see some custom activities that your admin created.

Types of activities

Activity type	Description
Request approval	Ask someone for permission to accomplish a task.
Assign to	Choose a person who should fulfill the task.
Create task	Specify a process that must be done as part of the playbook.
Send an email	Send an email to one or more people. You can specify images and enrich text for the email that gets sent automatically.
Placeholder	Set an undefined activity to be specified later, or a more advanced activity such as an email notification, when an activity is completed. Placeholder activities don't have any logic assigned to them yet, and must be edited in Workflow Studio. Or, you can swap them out later for another type of activity in Creator Studio.

Activity library for a playbook



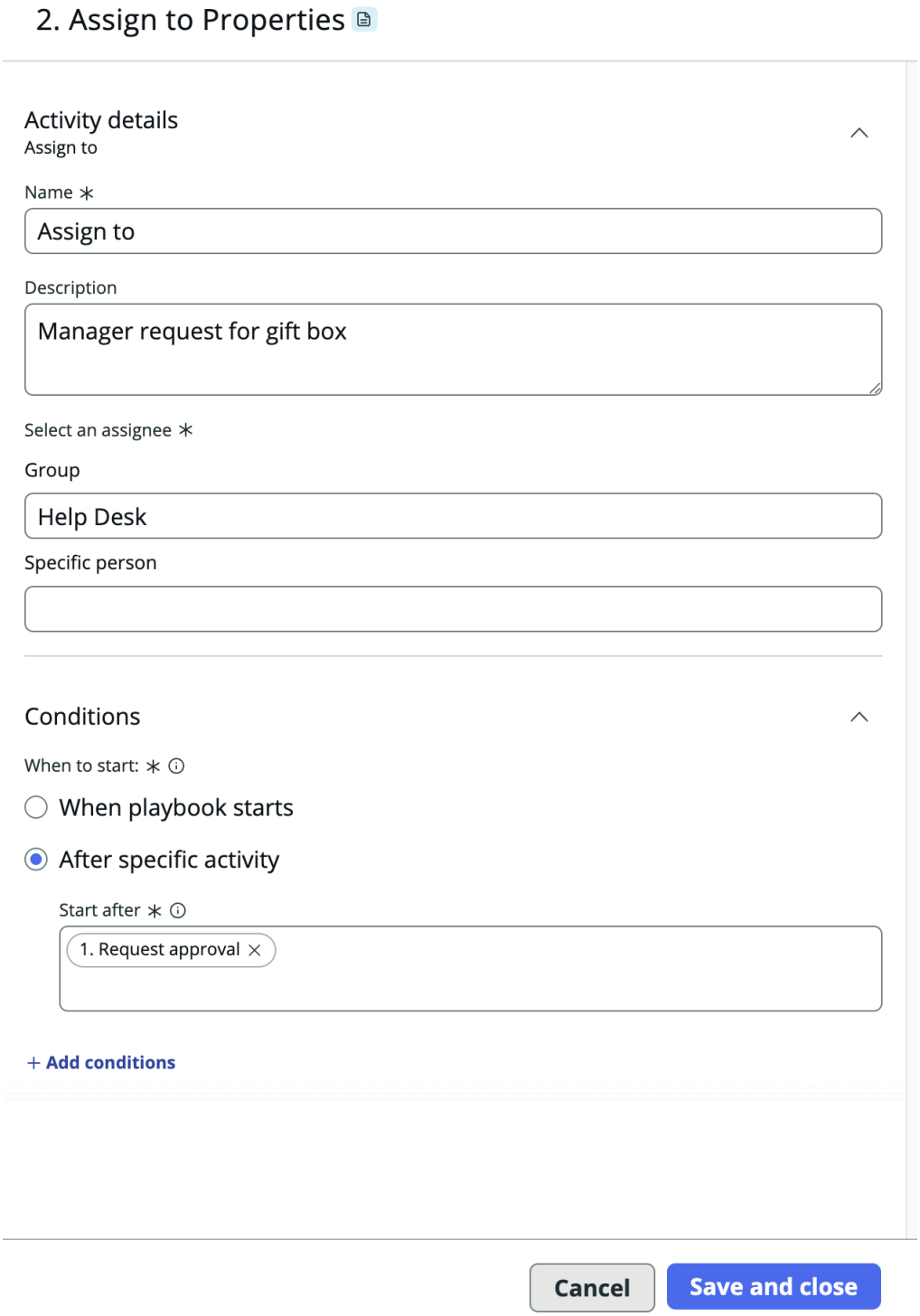
Your new activity and its Activity properties panel appear, where you define what the activity is.

8. Enter the basic details for the activity.

General activity details

Field	Description
Name	Unique, user-facing name for your activity, which appears to agents and fulfillers while the playbook is running.
Description	Optional details about what the activity accomplishes.

Activity details panel



9. If needed, specify the approver or assignee in the **Select an approver** or **Select an assignee** fields.
   Approvers and assignees are the person or people who must approve the task, or who the task gets assigned to. The options are:

- **Group**: Enter one or more user groups in the field that appears. For example, `App engine Admins`.
- **Specific person**: Enter the person's name as it appears on the ServiceNow AI Platform. If you entered a group, you can select only from users in the specified group.
- **Requester's manager**: (Request approval activities only) Automatically routes the approval to the manager of the person who's making the request.

10. Enter details for specific types of activities.

a. For Request approval activities only, specify the **Approval type**.
Approval type defines whether a single approver is needed or if everyone must approve the request. The options are:

- **Anyone approves**: Only one of the specified approvers must approve the request.
- **All approve**: Every one of the specified approvers must approve the request.

  If you select this option and specified a group, all members of the group must approve the request.

b. For Create task activities, specify the **Priority**.
The options are:

- **Critical**
- **High**
- **Moderate**
- **Low**
- **Planning**

c. For Send an email activities, specify the recipients and contents of the email.

- Enter the name (for ServiceNow AI Platform users) or email address for people who should receive the email in the **To** and **Cc** fields.

  Select the **Requester** option for the **To** or **Cc** fields to include the person who made the request in the email's recipients.

- Enter the email's subject in the **Subject** field.
- Enter the body of the email in the textbox. Select the additional toolbar buttons icon () to display more formatting options. For example, you can add images, tables, and links to the email.

11. Select at what point the activity should start running in the **When to start** field of the Conditions section.
The options are:

- **When** *`playbook`* **starts**: The approval request runs when the trigger occurs, which is the form's record being created or updated.
- **After specific activity**: The approval request runs after one or more activities run, which you must specify in the **Starts after** field. You can select only activities that occur sequentially before the activity you're working on.

  Note:
  This option isn't available for the first activity in a playbook.

12. **Optional:** Define any extra conditions that must be met for the activity to happen by selecting **+ Add conditions**.
    Use the condition builder that appears to specify any other things that must happen for the activity to run. For more information, see Create a condition statement using the condition builder.
13. Select the **Save and close** button to finish setting up your activity.

What to do next

Continue adding activities and decisions, as well as connectors and parallel processes if needed, to finish creating your playbook. For more information on decisions, see Add a decision to an app's playbook in Creator Studio.

Add a decision to an app's playbook in Creator Studio

Add decisions, which are if/then conditions, to define branches, or different paths of an automation's playbook in Creator Studio.

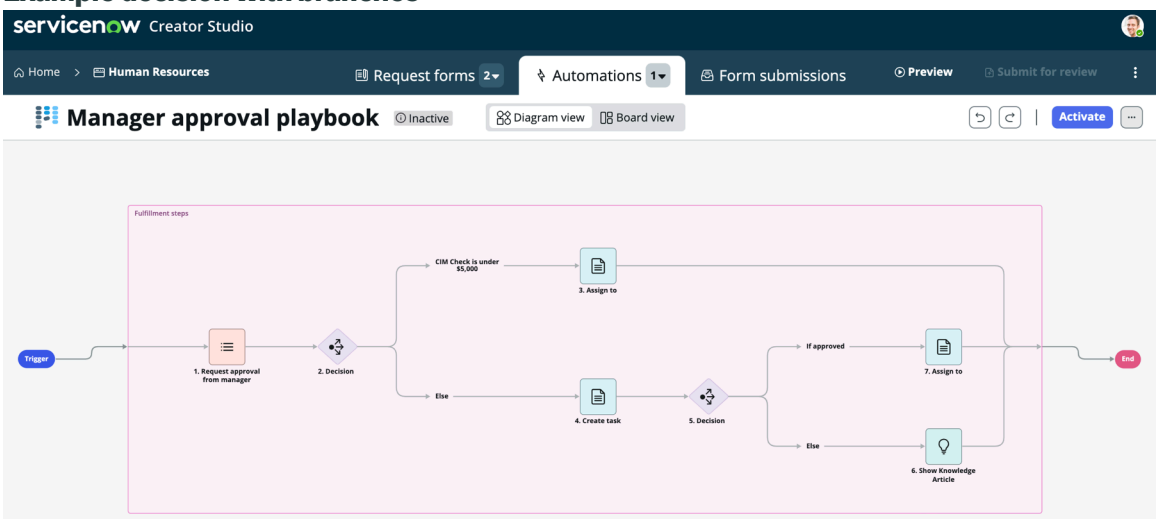
Before you begin

To add decisions to a playbook, you must be given permission to work on the app.

About this task

Each decision should have at least two branches, with each branch representing a possible outcome for the decision. The second branch can be the ELSE branch, which is required. The ELSE branch will be followed when all other conditions on other branches are false. Think of the else branch as the catch-all case for the decision.

Example decision with branches



Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the playbook you want to add a decision to.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

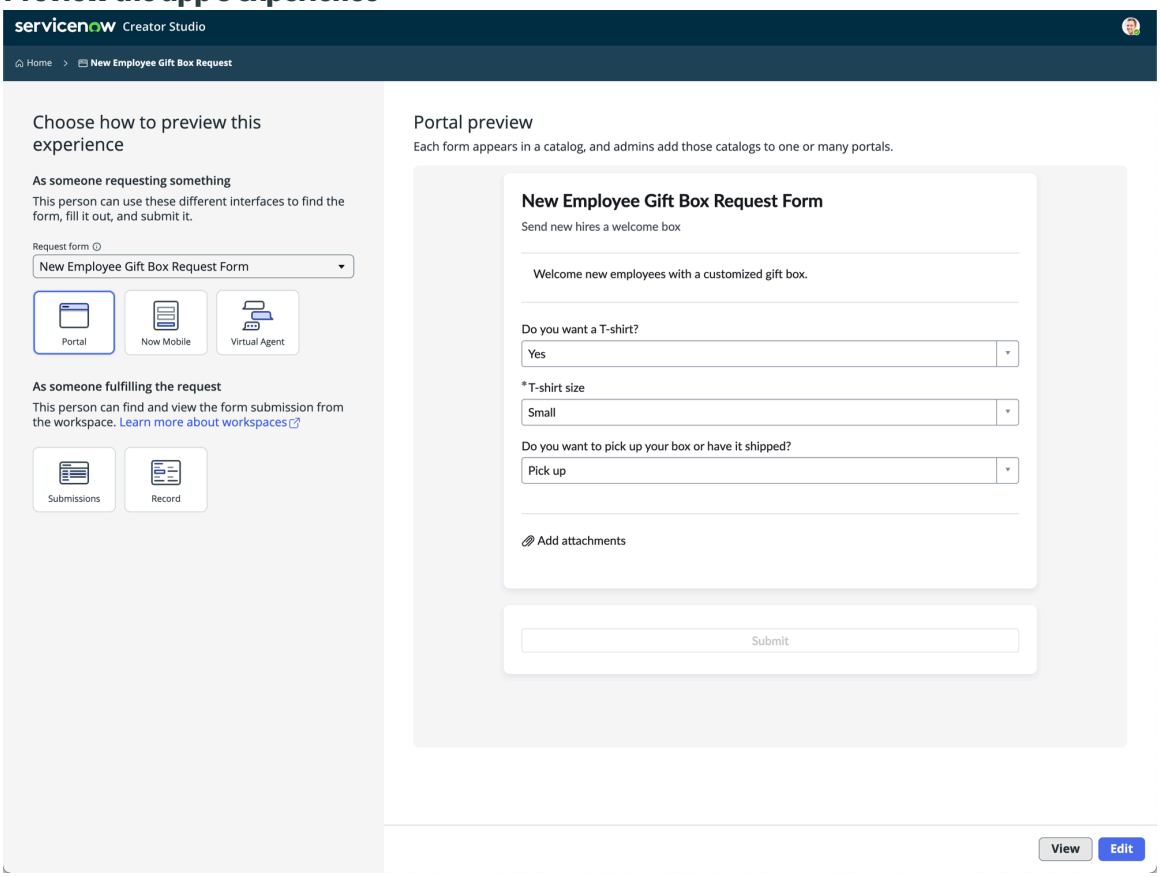
   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)

- **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

> **Note:**
> Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

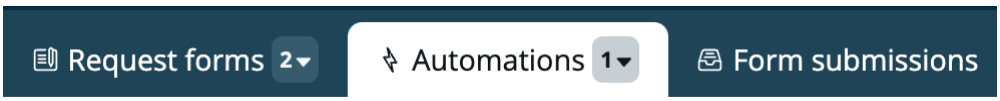
The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

Preview the app's experience



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Automations** tab in the application header.
5. Check that you're editing the correct playbook in your app by selecting it from the **Automations** tab.



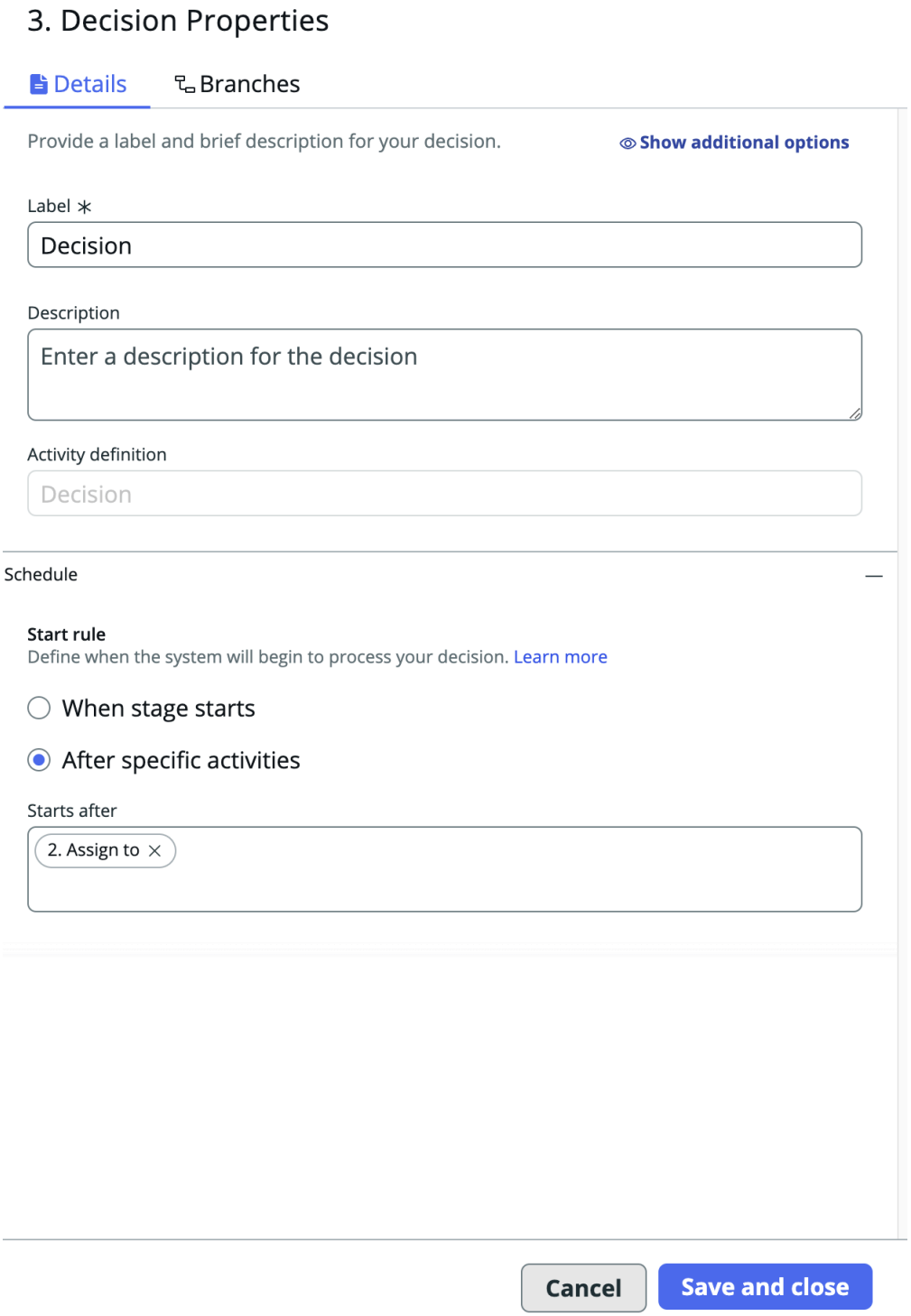
6. Select the add icon ( ) on the connector here you want to add a decision and choose the diamond-shaped **Add a decision** icon ( ) in the menu that pops up.
   You must be in the Diagram view to add a decision.

7. Specify the decision's basic attributes on the **Details** tab of the Decision properties panel that appears.

Decision basic options

Field	Description
Playbook name	Unique, user-facing name for the decisions, which appears to agents and fulfillers while the playbook is running.
Description	Optional details about what the decision accomplishes.

Playbook decision properties



8. Define the decision's schedule.

Decision schedule options

Field	Description
Start rule	Choose when you want your activity to start running. The options are:

Field	Description
	○ **When stage starts**: The decision starts running as soon as the stage starts running, which is when your playbook is triggered. ○ **After specific activities**: The decision starts running after the specified activities have finished running.
Starts after	Select the activity that must finish running for the decision to happen. This field is editable only if you selected to start the rule **After specific activities**. You can choose only activities that happen before this decision in the playbook.

9. **Optional:** Specify more scheduling conditions for the decision, such as whether there's a delay, by selecting **Show additional options**.

Decision schedule additional options

Field	Description
Display order	Order in which this activity occurs during a playbook run.
Start with delay	Toggle to specify that the ServiceNow AI Platform waits for a duration of time before running the decision after the start rule is met. For more information on how to specify the delay duration, see Start with delay input properties.
Restart rules	What the decision does when a playbook is restarted. The options are: ○ **Skip on restart**: Skip this decision when the playbook run is due to a restart. ○ **Run always**: Always run this decision, including first runs. ○ **Skip on first run**: Skip this decision during the first run. For more information, see Restart a playbook.

10. Create the conditions for each branch, or possible outcome for the decision on the **Branches** tab of the Decision properties panel.

    a. Enter a name for the branch in the **Branch label** field.

    b. Select the **Add condition** button and specify what conditions should be met for the branch to take effect.
    For more information, see Create a condition statement using the condition builder.

    c. **Optional:** Select the **Add new branch** button and add as many branches as needed.

d. If you add two or more branches, select how you want your decision activity to run the different branches.

Branch processing instructions

Option	Description
Process any & all branches that are true	The app processes all branches with conditions that are met.
Process only the first one that is true	The app processes only the first listed branch with conditions met. If you select this option, drag the branch that you want processed first to the top.

e. **Optional:** Adjust the order of branches as needed by dragging them into a new position.

11. Select **Save and close**.

Add a parallel process to an app's playbook in Creator Studio

Create parallel processes in Creator Studio to make things happen in your app at the same time.

Before you begin

To add a parallel process to a playbook, you must be given permission to work on the app.

About this task

Activities on a process's parallel branches run at the same time as other specified parallel activities. Unlike decision activities, the activities and stages on a parallel can run on the same conditions as other branches.

Procedure

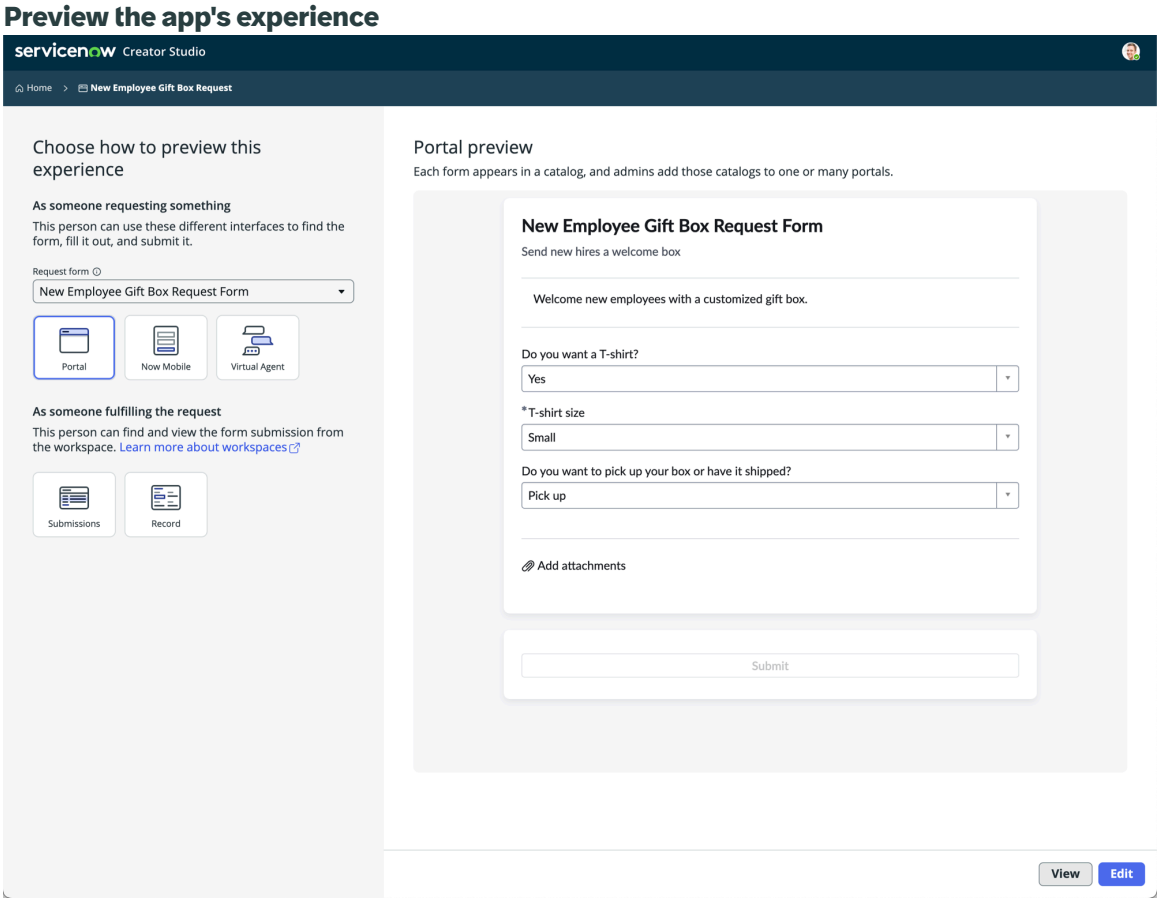
1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the playbook you want to add a parallel process to.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

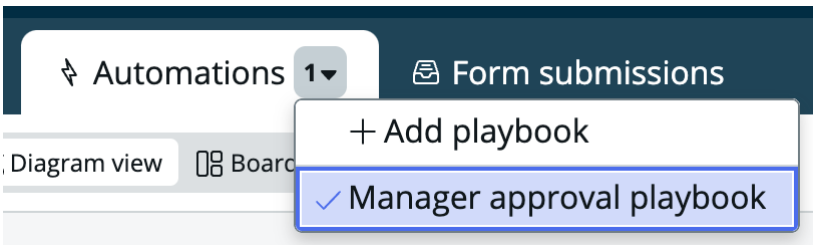
   Note:
   Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.



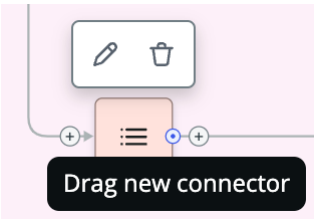
You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Automations** tab in the application header.
5. Check that you're editing the correct playbook in your app by selecting it from the **Automations** tab.



6. Choose one of the following ways to add a parallel branch in the diagram view.
   - Select the add icon ( ) and choose the **Add a parallel path** icon ( ) in the menu that pops up.
   - Select the Drag new connecter dot icon ( ) for the parallel process's starting activity and drag it to the activity or decision that should be the end point of the parallel process.

Note:
If you can't see the Drag new connector dot icon, you might need to hover over the activity to see it.



7. Add activities to the parallel branch as needed.
   For more information, see Add activities to an app's playbook in Creator Studio.
8. If you want to have several activities happen in a parallel process as soon as the playbook is triggered, complete the following steps.
   a. Display the Board view.
   b. For each activity that should be happen in parallel when the playbook is triggered, select the activity's card and set the **When to start** field to **When playbook starts**.
   c. Select the **Save and close** button.

Edit the trigger for a playbook in Creator Studio

Define the trigger for a playbook in Creator Studio to specify what makes the playbook start running.

Before you begin

The trigger initiates a playbook to run on one of the app's forms. For example, when a user selects a specific answer to a question, the playbook begins to run. The form you select must already be configured for you to select it for the trigger.

To edit the trigger for a playbook, you must be given permission to work on the app.

About this task

Note:
You can't change a playbook's trigger (whether the form is submitted or updated to initiate the playbook) or how frequently it should run after you create the playbook. Instead, create a new playbook with a different trigger.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the playbook you want to add actions to.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

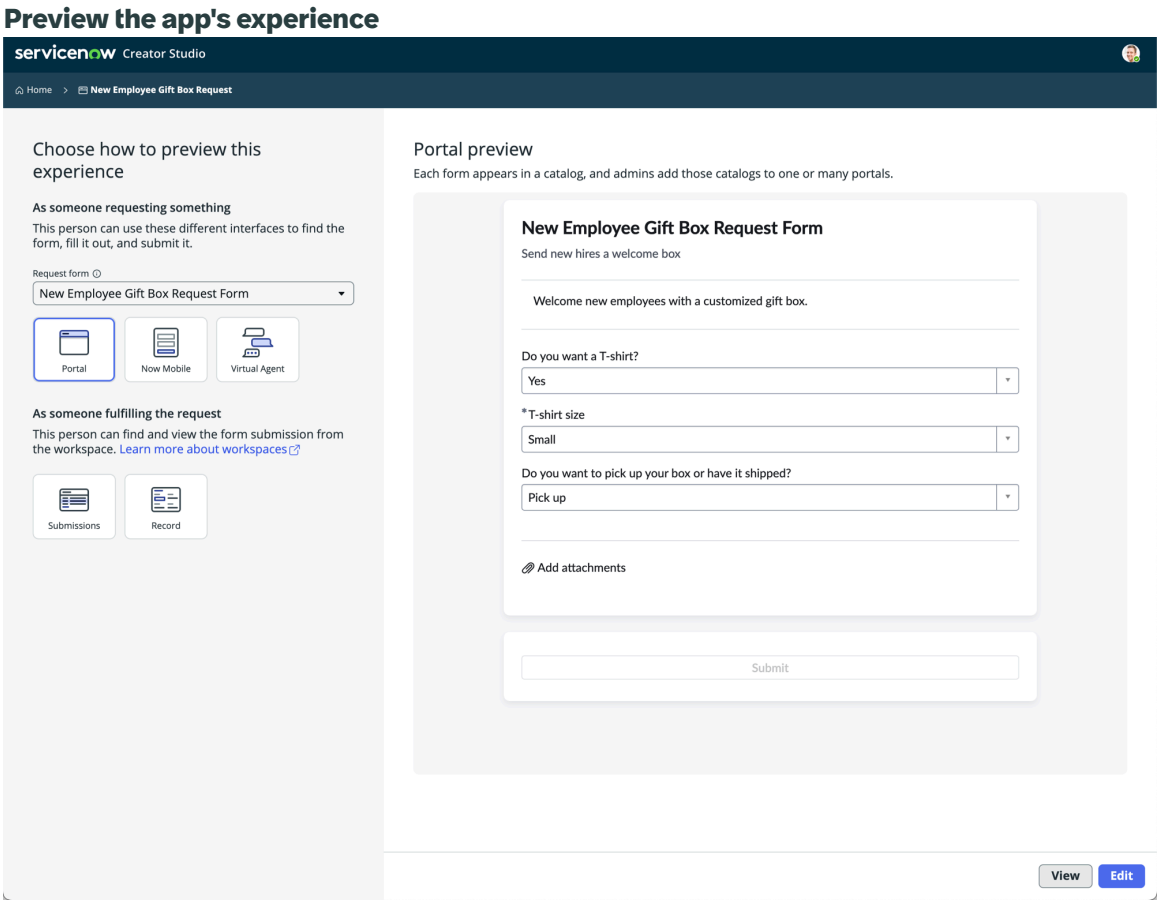
   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)

- **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

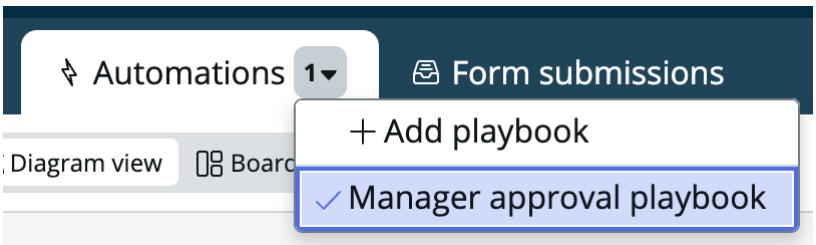
> **Note:**
> Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Automations** tab in the application header.
5. Check that you're editing the correct playbook in your app by selecting it from the **Automations** tab.



6. Select the **Trigger** (Trigger).

The Additional properties modal appears, where you can edit some of the trigger's settings.

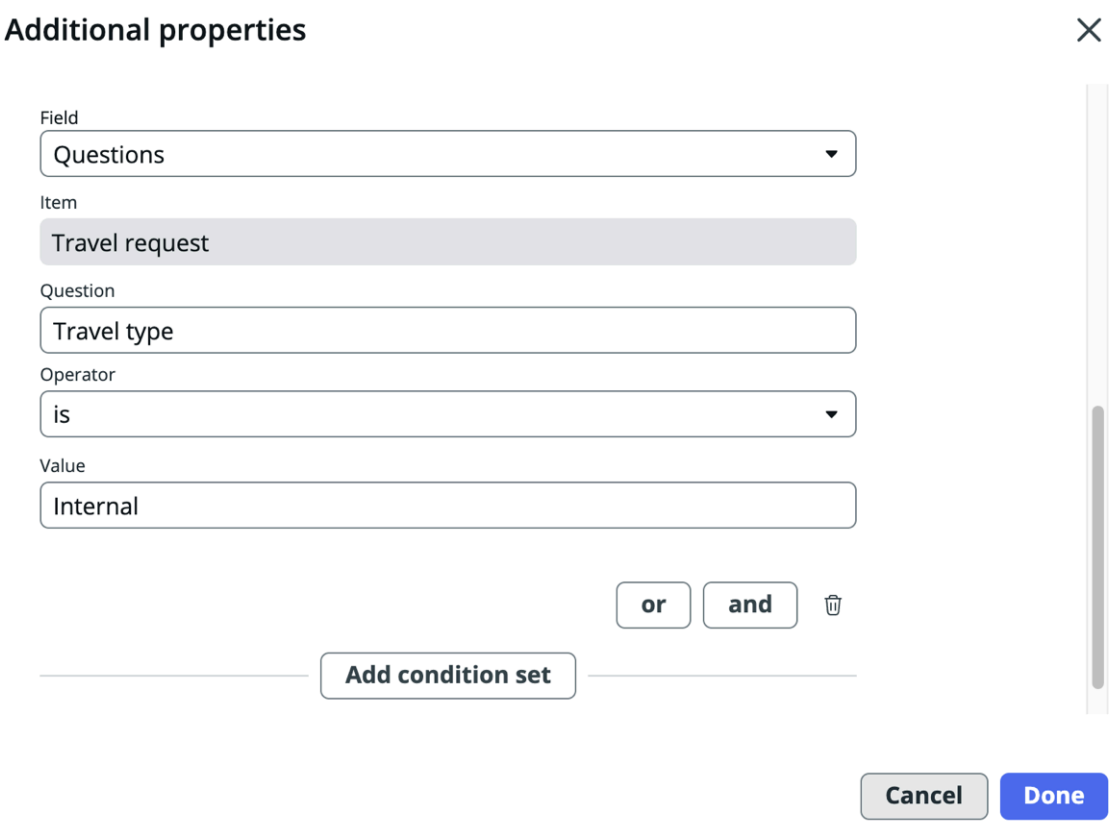
7. Make any changes to the trigger's general attributes.

General playbook definition fields

Field	Description
Playbook name	Descriptive name for the playbook you're editing.
Description	Brief explanation of what the playbook does, for example, the end goal for the record type.

8. If necessary, change the **Form** whose catalog item generates the record type that you want the playbook to run on.
9. Change the conditions that must be met for the playbook to begin running by selecting **Add condition set**.
   - If you want to trigger the playbook based on the value of a column in a table, select the **Field** that you want to be the trigger, as well as its condition **Operator** and the specific trigger **Value**. For example, when a **Start date** is **after** the **Date** needed.
   - If you want to trigger the playbook based on the response from a form, select **Questions** as the trigger **Field**. Then select the question you want in the **Question** field, the condition **Operator** and the answer's **Value**.

Question answer as trigger for an automation



Add as many conditions as you need. For more information, see Create a condition statement using the condition builder.

10. Save your changes by selecting the **Done** button.

Edit a playbook in Creator Studio

Update a playbook to change its settings, or rearrange or remove tasks, to make it work for your app.

Before you begin

To edit a playbook, you must be given permission to work on the app.

About this task

Note:

Not all items that can be added to playbooks are supported in Creator Studio, for example, notifications.

If you want to add complex, unsupported items to a playbook (such as optional activities, multiple stages, or data pills from previous activities), you must open and edit that playbook in Workflow Studio. After you edit a playbook in Workflow Studio, you can't edit it in Creator Studio. However, you can still work on the rest of the app in Creator Studio.

Procedure

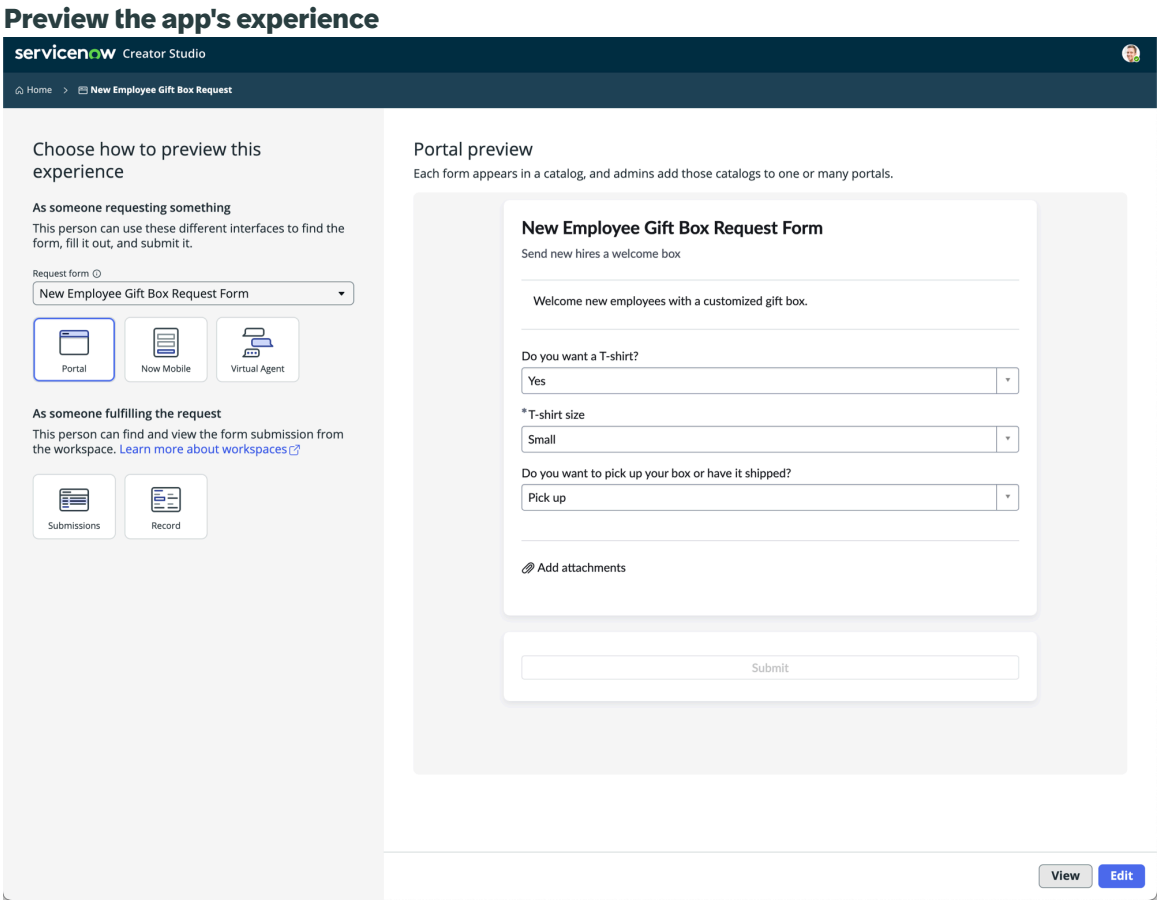
1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the playbook you want to edit.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

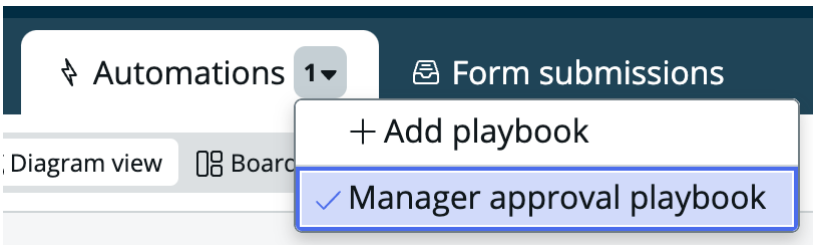
     Note:
     Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

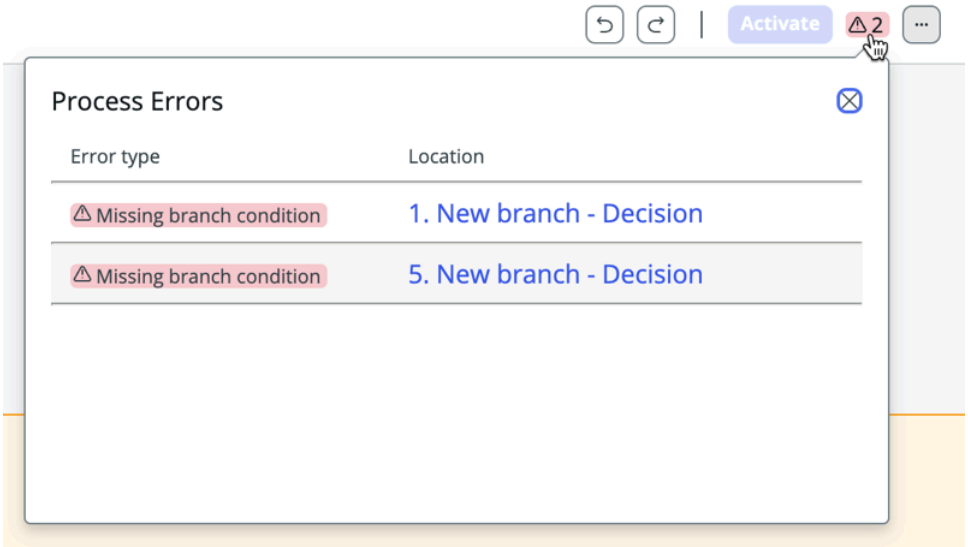


You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Automations** tab in the application header.
5. Check that you're editing the correct playbook in your app by selecting it from the **Automations** tab.



6. Address any errors in the playbook's logic by selecting the error message icon.
In the error tray that appears, you can open the properties to fix each error by selecting the **Location** link.



7. Edit the playbook's settings.

   a. Select the more actions icon (···) and select **Properties**.

   b. Update any of the settings.

   > **Note:**
   > You can't change a playbook's trigger once it's created. Instead, add a new playbook with a different trigger.

   c. Select the **Done** button.

8. Rearrange the order of tasks as needed.

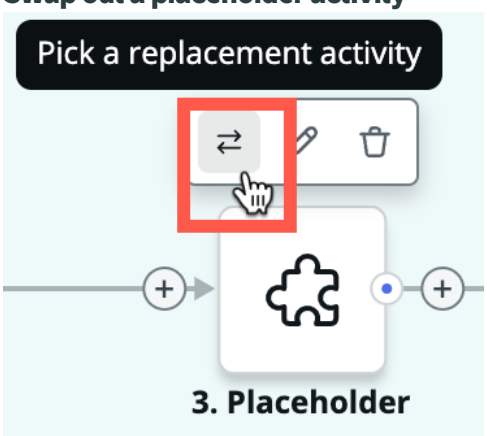
   a. Display the **Board view** for the playbook.

   b. Drag the cards for each activity into the order that you want.
   Creator Studio automatically saves the changes.

9. Swap a placeholder activity if you want to replace it with another activity type.

   a. Hover over the placeholder activity and select the replace activity icon (⇄) to directly open the activity picker.

Swap out a placeholder activity



b. Select the new activity from the activity picker.

c. Optional: Update the **Label** and **Description** as needed, or any of the other properties of the activity.
For more information, see Add activities to an app's playbook in Creator Studio.

10. Delete any unnecessary activities by hovering over the activity and selecting the delete icon ( ).

11. Make any advanced edits in Workflow Studio by selecting the more actions icon ( ) and selecting **Open in Workflow Studio**.

Activate a playbook in Creator Studio

Activating a playbook means that it will run when its related form is created or updated on your non-production, development instance. However, the app must still be deployed to production.

Before you begin

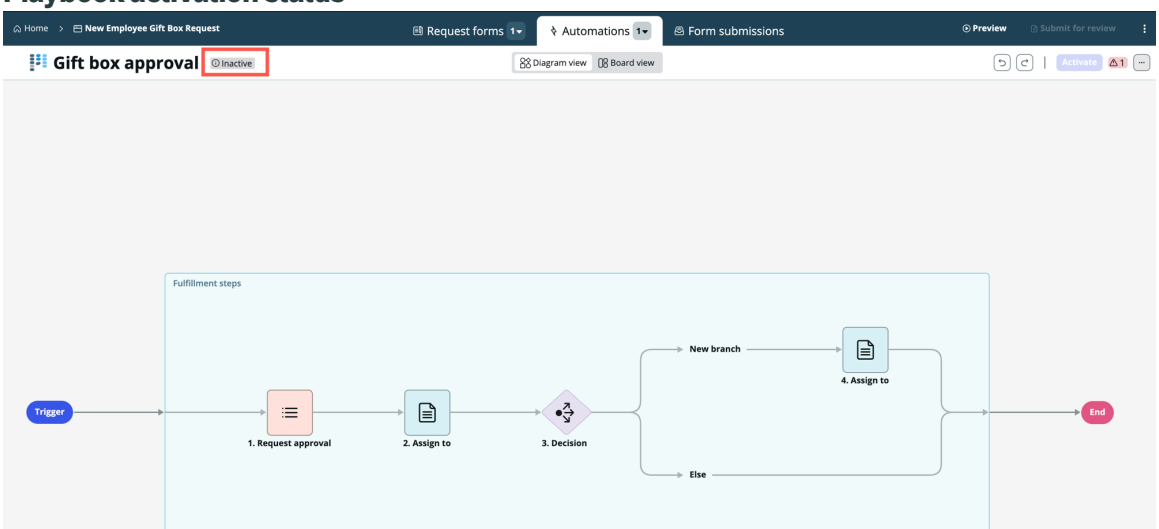
You must resolve any errors in the playbook before you can activate it.

To activate a playbook, you must be given permission to work on the app.

About this task

You can check the activation status of a playbook at any time in the **Automations** tab header.

Playbook activation status



If you don't activate a playbook and its app is deployed to production, the automation won't run on the applicable records. However, your App Engine admin can activate the deployed playbook for you.

Procedure

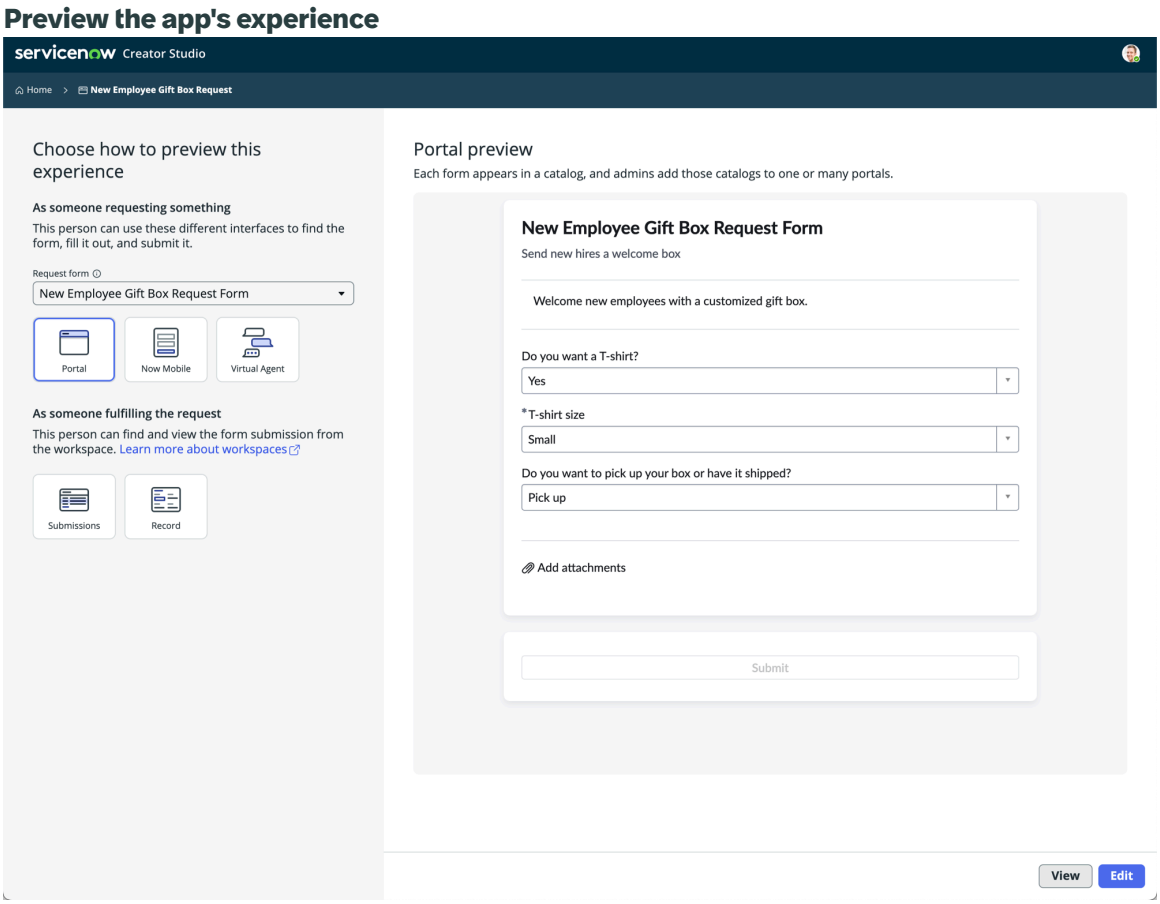
1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the playbook you want to activate.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

- **Portal** (see a preview of how it'll appear on a desktop website)
- **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
- **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

> **Note:**
> Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

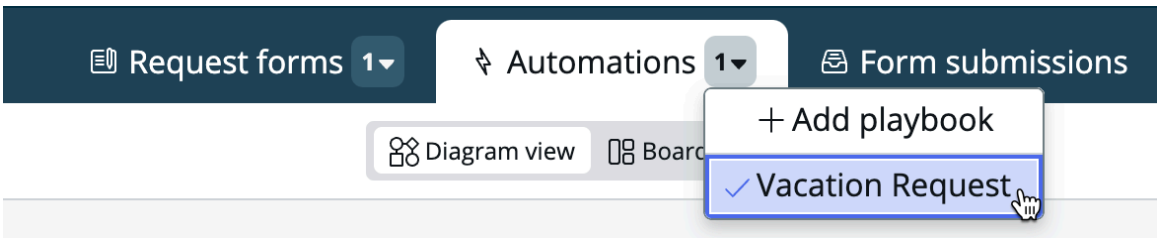
The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

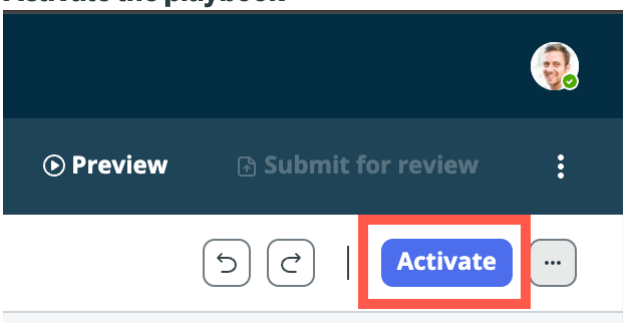
4. Select the **Automations** tab in the application header.
5. Check that you're activating the correct playbook in your app by selecting it from the **Automations** tab.

Select automation



6. Select the **Activate** button.

Activate the playbook



Result

The status displayed in **Automations** tab header updates from **Inactive** to **Saving** to **Active**, informing you of when changes are made.

The playbook is ready to be deployed with your app. Once the app is deployed, the form is available in the associated service catalog. If the form's app hasn't been deployed, you need to deploy it.

Delete a playbook in Creator Studio

Delete a playbook to remove it from the app completely.

Before you begin

To delete a playbook, you must be given permission to work on the app.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the playbook you want to delete.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

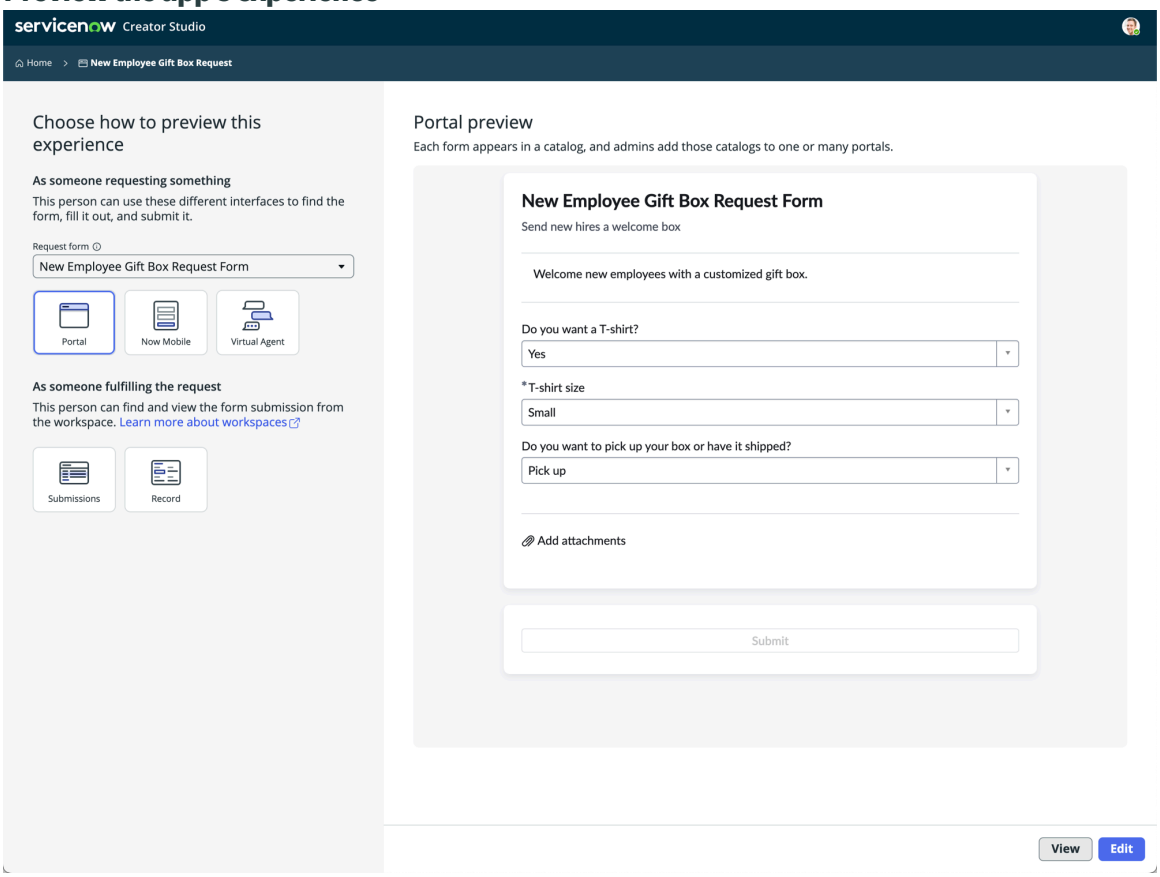
   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)

- **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

> **Note:**
> Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

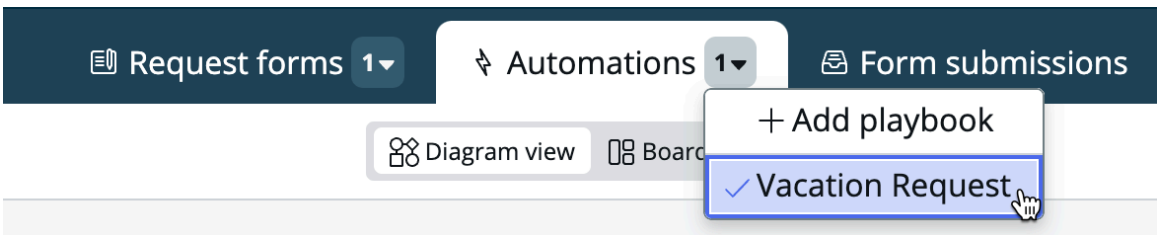
Preview the app's experience



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

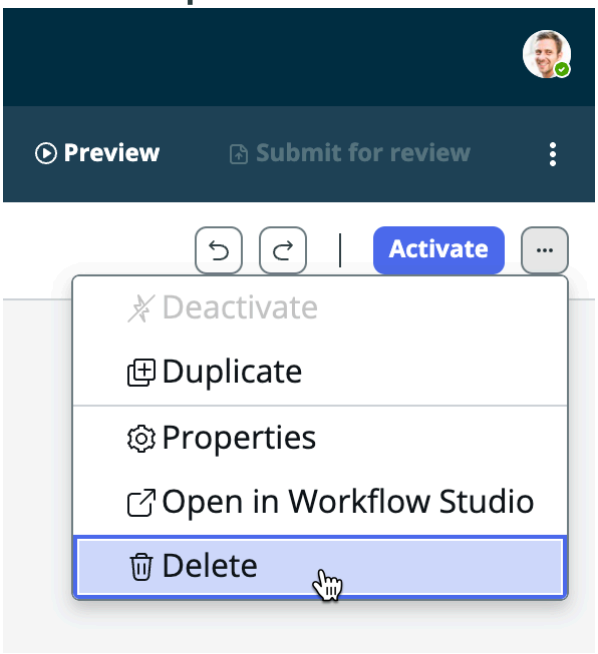
4. Select the **Automations** tab in the application header.
5. Check that you're deleting the correct playbook in your app by selecting it from the **Automations** tab.

Select automation



6. Select the more actions icon ( ).
7. Select **Delete**.

Delete menu option



8. On the confirmation modal, select the **Delete** button.

Working with form submission workspaces in Creator Studio

So, you've built your app, and people are submitting requests through it, that's great! However, you might be wondering where those requests go after they're submitted. The short answer is they live in a form submission Workspace, which you can customize in Creator Studio to suit your needs.

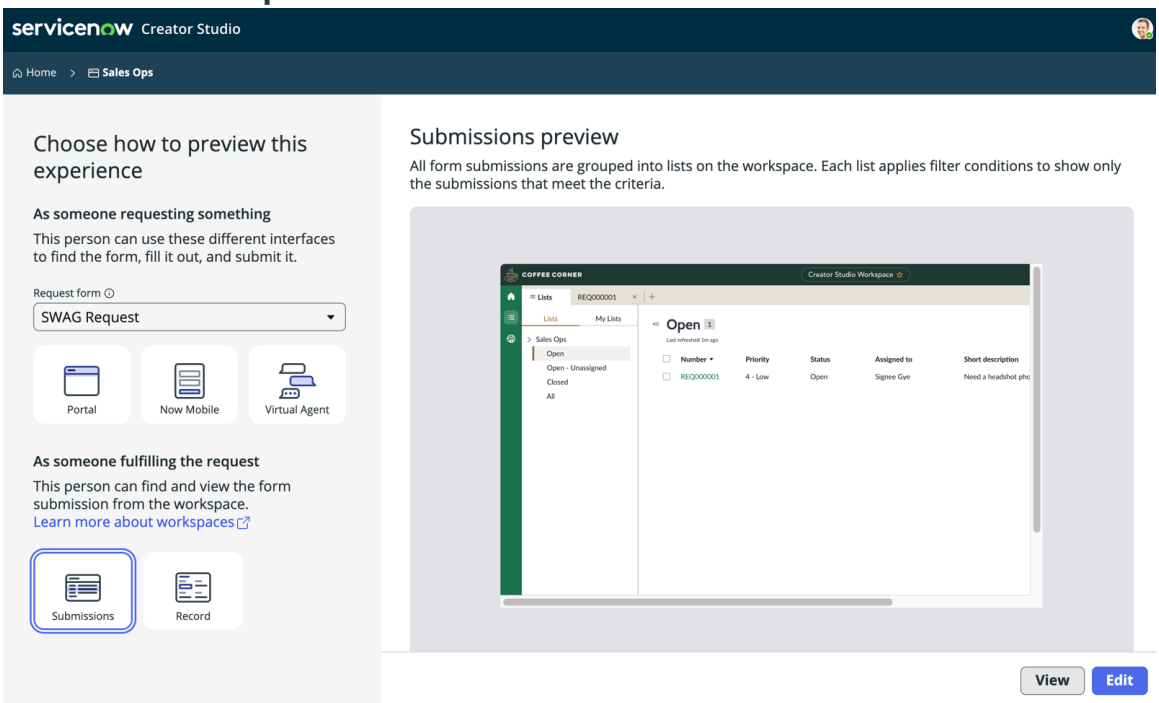
A little more about workspaces

The Request App Workspace holds form submissions that come in from each of your apps. Fulfillers (remember, these are the people who work on requests) can access the workspace through the ServiceNow AI Platform and see submissions for all the apps they're assigned to.

Each app that you build in Creator Studio has its own category in the Request App Workspace, regardless of how many forms or automations it has.

As the app owner, you're responsible for the look and feel of the workspace that the app's fulfillers use. Thankfully, the workspace should be ready to use out-of-the-box, but you can also customize it on the **Form submissions** tab of your app in Creator Studio.

Form customization preview



When you open the **Form submissions** tab, you'll see some filtered lists that appear in the default workspace. These filtered lists (**Open**, **Open – Unassigned**, **Closed**, and **All**) are intended to give fulfillers a quick view of all the form submissions. You can add more filtered lists as needed, for example, you could add a filtered list for each of the app's forms or for canceled requests.

If it would help your fulfillers to have additional categories, you must ask your admin to add them using the List Category [sys_ux_list_category] table on the ServiceNow AI Platform.

Creating lists to support multiple forms

When fulfillers open the workspace to view an app's form submissions, the default "Open" list displays all open requests made through the app. However, if you have multiple forms in your app, fulfillers might find it easier to have separate "Open" lists for each form in the app. In that case, you should create different lists for each separate form and name them after the forms. Find out how in Add a filtered list to a workspace in Creator Studio.

Alternatively, fulfillers can use the "My lists" feature in Workspace to create custom lists that fit their needs. Tell them how to by checking out Create My Lists in workspace.

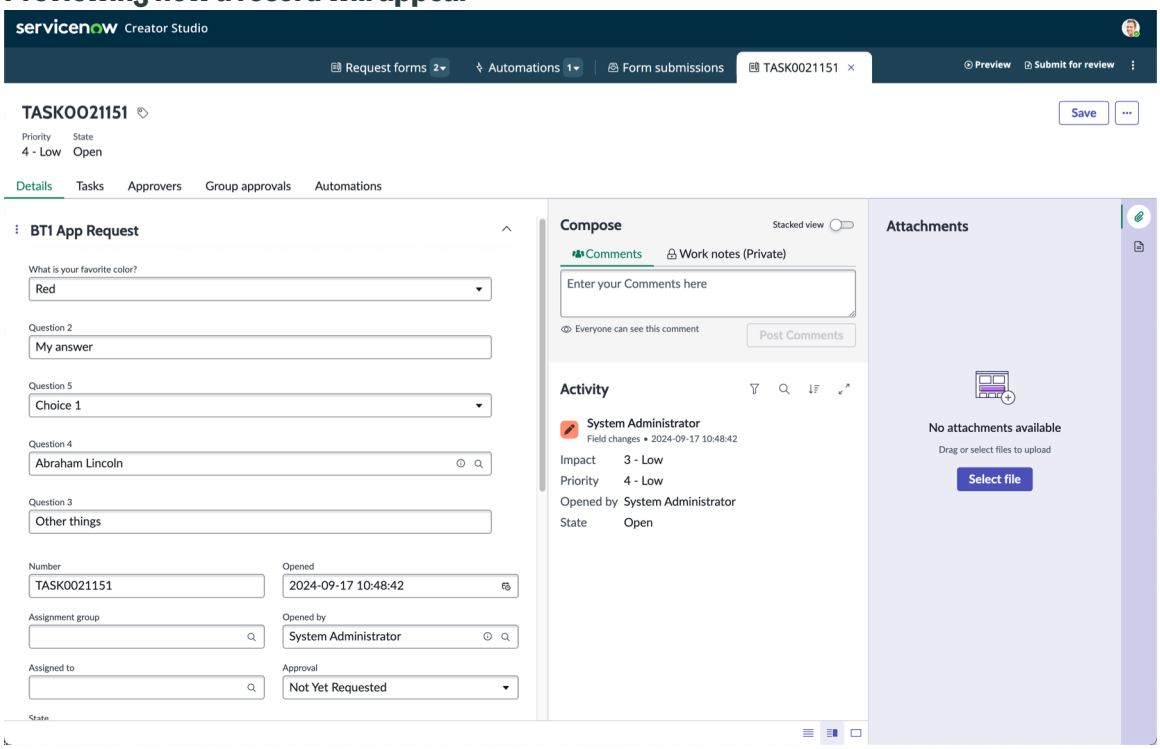
How fulfillers access submitted forms in Workspace

Fulfillers access the default workspace on the ServiceNow AI Platform by going to **All** > **App Engine** > **Request App Workspace**. The Request App Workspace displays a category for each app with filtered lists of records, both default and any extra that you added. Fulfillers then select a record to view the details of the request and make updates as needed.

Note:
You can't access the Request App Workspace as the owner of an app, though you can access it if you also have an admin or the applicable agent role. The app's agent role is <app scope>.agent, for example, x_snc_app_name.agent.

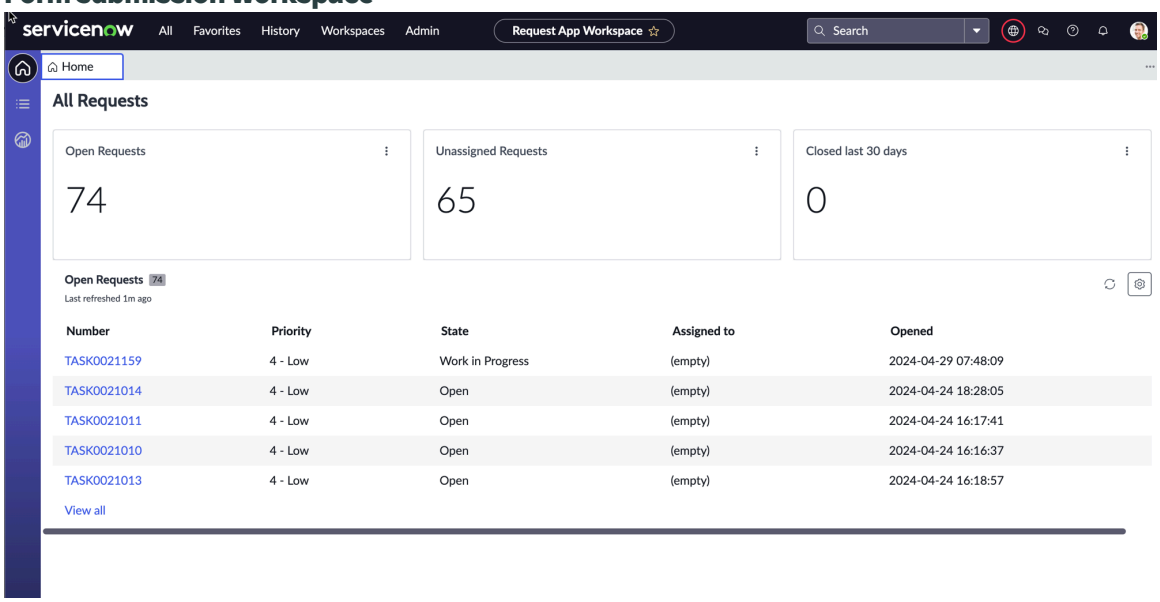
You can also preview and interact with how records will appear in the Request App Workspace directly within Creator Studio.

Previewing how a record will appear



What fulfillers can see in the Request App Workspace depends on their permissions. For example, admins can see workspace categories for all apps built in Creator Studio. However, most users see only the workspace category for their particular app.

Form submission workspace



When fulfillers view a record for an individual request in the workspace, the **Catalog tasks** tab displays the playbook that you configured in Creator Studio.

Fulfillers or admins may also want to monitor the app's performance as requests come in. This is where the Analytics Center in the Request App Workspace is helpful! On the **Analytics** tab, each app has its own analytics dashboard that shows applicable data, like how many open requests you have or what tasks are related to the app. There's a lot of helpful information here. Find out more about working with data in Analytics Center.

Add a filtered list to a workspace in Creator Studio

Create custom filtered lists in an app's workspace to view records that meet specific conditions. For example, if your app has multiple forms, you can create a list for each form by filtering on the record type.

Before you begin

To add a list to a workspace, you must be given permission to work on the app.

About this task

Note:
You can create a filtered list to display open records assigned to a specific user, or all requests that have been cancelled.

Procedure

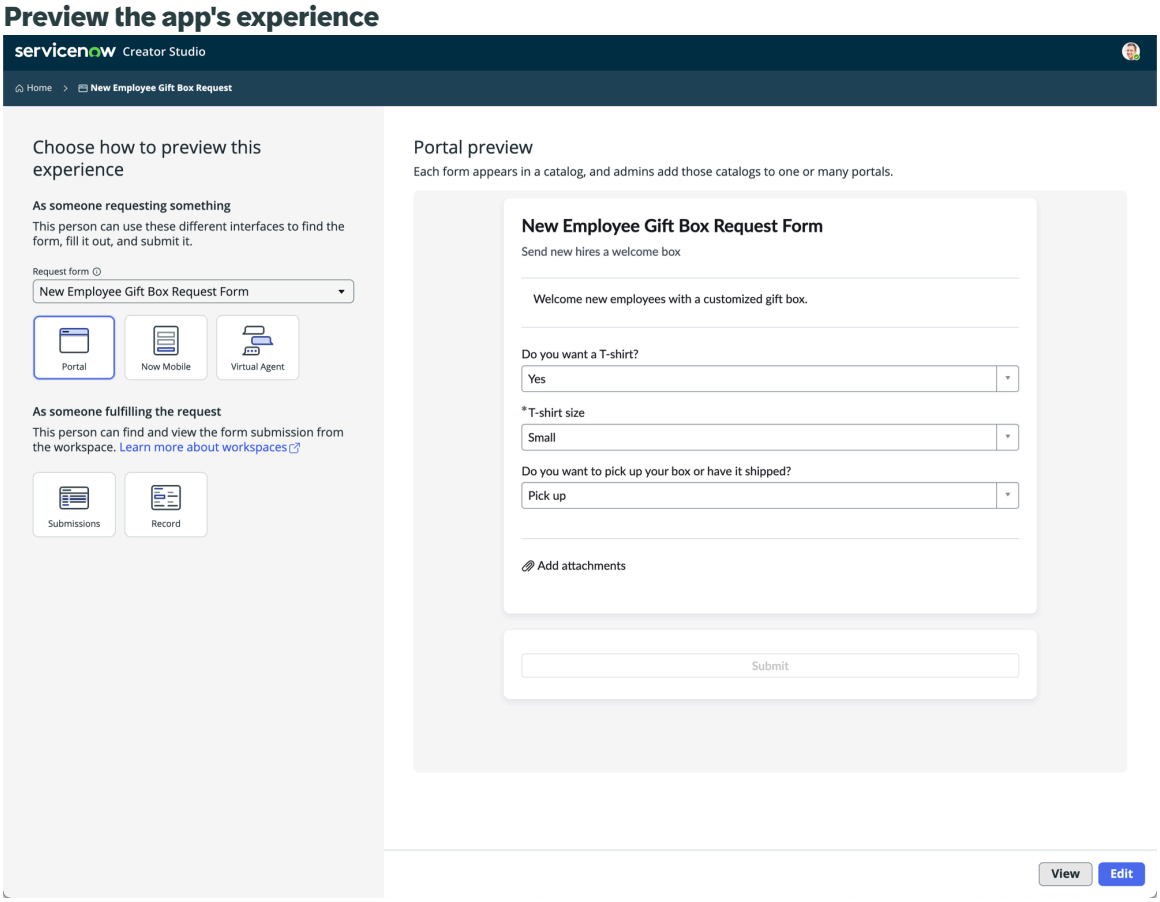
1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the workspace you want to add a list to.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

     Note:
     Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

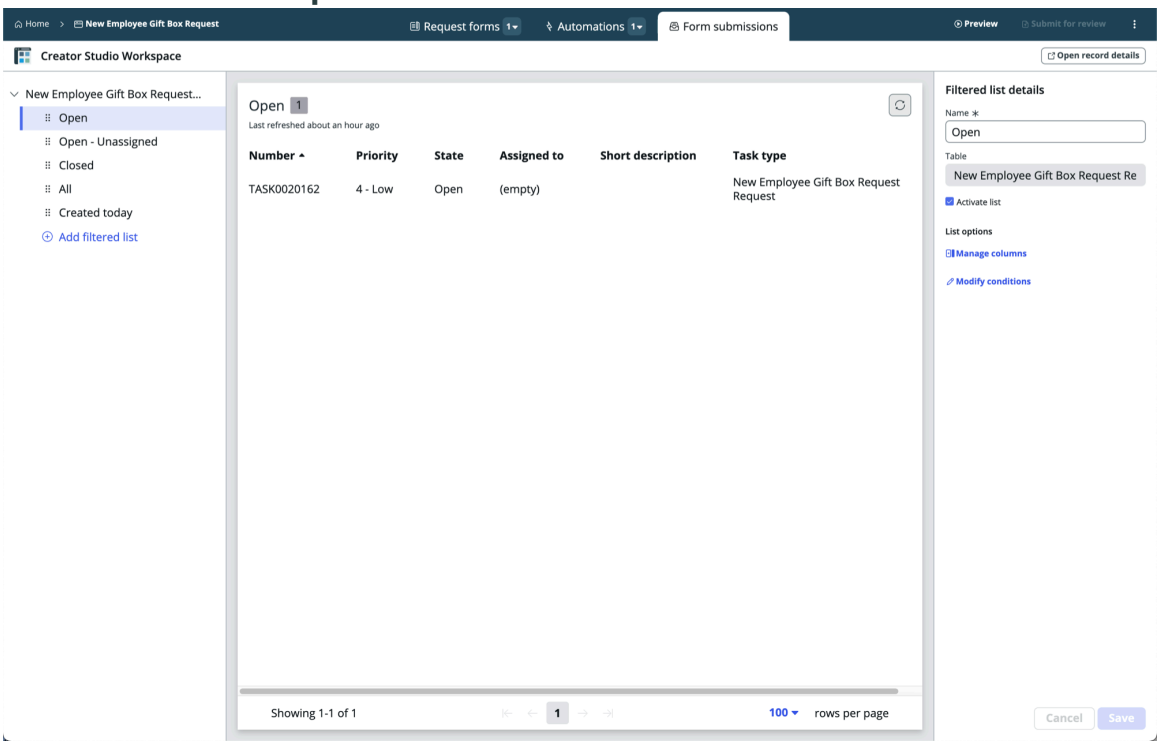
   The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

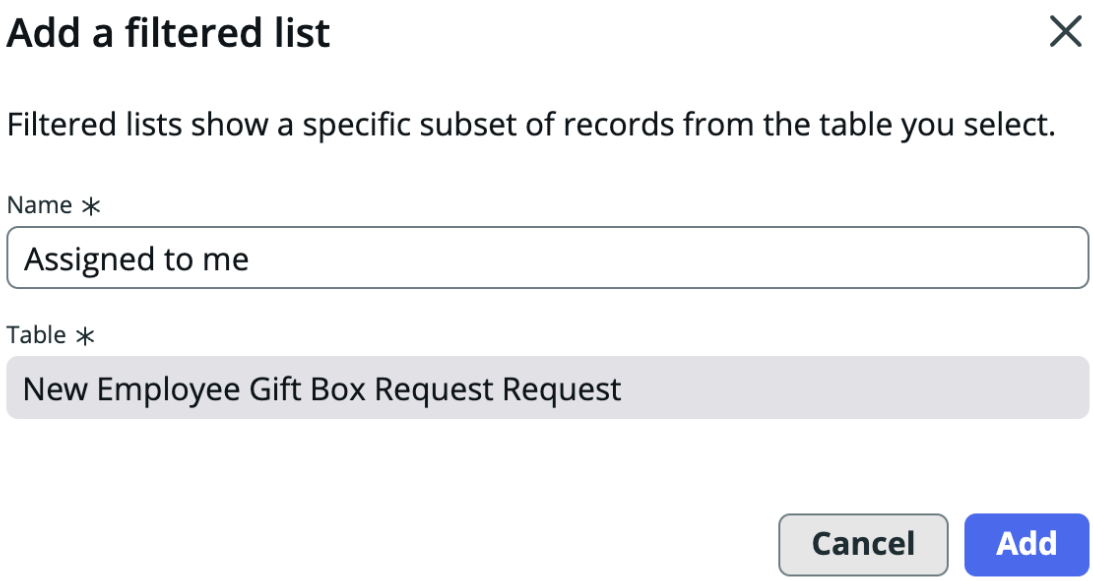
4. Select the **Form submissions** tab in the application header.

Form submission workspace in Creator Studio



5. Select **Add a filtered list**.

Add a filtered list



6. Enter a **Name** for the new list in the modal that appears.

   Choose a name to help users easily identify their category in the workspace, such as `HR timeoff requests`.

   Note:
   You can't change the table for the workspace, as it's linked to the app.

7. Select the **Add** button.
   The new list appears under the category, with its details appearing in the Filtered list details panel.
8. Select **Apply conditions** in the Filtered list details panel and use the condition builder to specify what types of records the list should contain, selecting **Apply** when you're done. For example, you can select **Urgency** as the field and **High** as the value to have the list show only urgent records. For more information, see Create a condition statement using the condition builder.

   For example, the following conditions specify open and closed requests:

   - `Active is true` specifies only open records.
   - `Active is false` specifies only closed records.
9. **Optional:** Change which columns appear and the order they're displayed in by selecting **Manage columns**.
   a. Move columns from **Available columns** to **Selected columns** in the Manage columns dialog box.
   b. Drag to rearrange the columns.
   c. Select **Apply**.
10. Ensure that the **Activate list** option is selected.

Result
Your new list will appear in the Request App Workspace after your admin deploys it!

Customize an app's workspace in Creator Studio

Make changes to the app's workspace category where fulfillers review submitted requests from apps built in Creator Studio to adjust how it appears. For example, you can rearrange columns in its lists and customize how the record generated by the request app looks.

Before you begin
To customize how the form submission workspace appears, you must be given permission to work on the app.

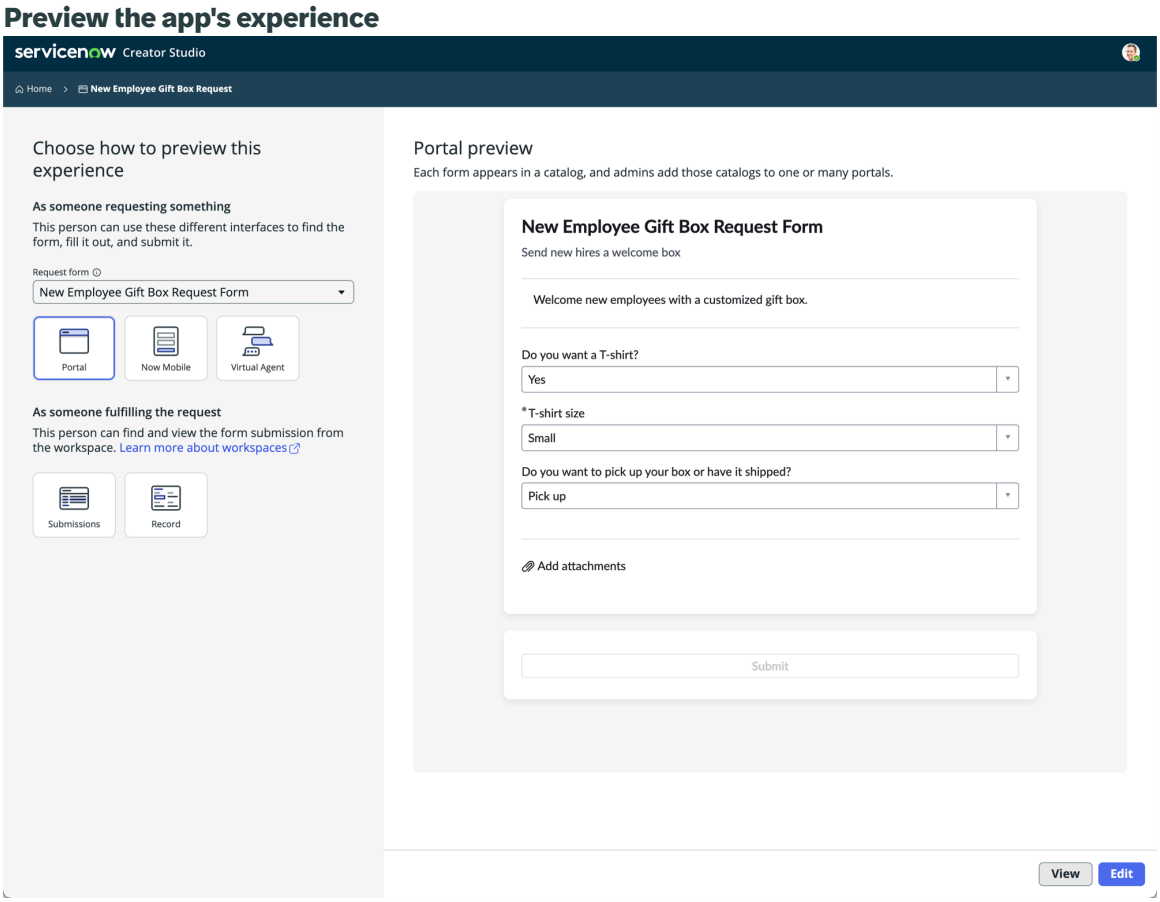
Procedure
1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the form submission workspace you want to edit.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)
   - **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

   Note:
   Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

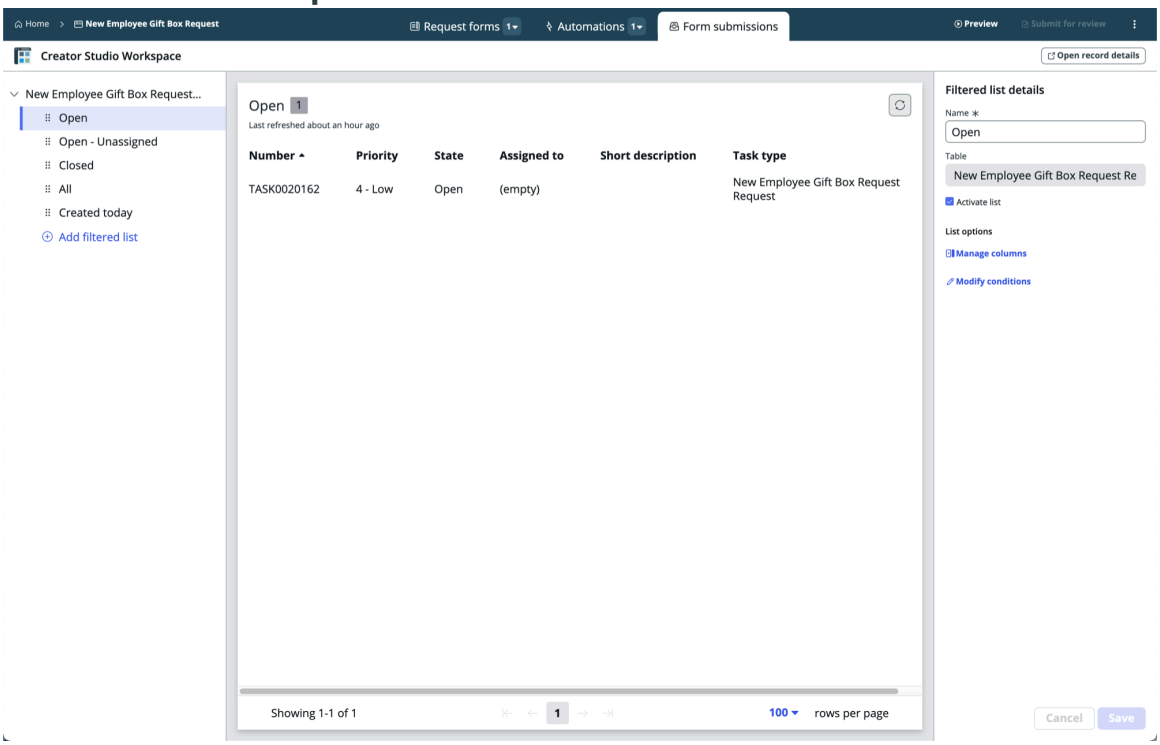
The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Form submissions** tab in the application header.

Form submission workspace in Creator Studio



5. Make the edits that you need to customize the workspace.

Edits you can make to a workspace

Update	Process
Edit the name of a filtered list	**a.** Select the list in the list navigation panel. **b.** Update the **Name** of the list in the Filtered list details panel that appears. **c.** Select **Save**.
Manage columns to change which columns appear and the order they're displayed in	**a.** Move columns from **Available columns** to **Selected columns** in the Manage columns dialog box. **b.** Drag to rearrange the columns. **c.** Save the update by selecting the **Apply** button.
Update the types of records that appear in a list	For example, you can select **Urgency** as the field and **High** as the value to have the list show only urgent records. For more information, see Create a condition statement using the condition builder.

Update	Process
	a. Select **Modify conditions** in the Filtered list details panel. **b.** Use the condition builder to update what types of records the list should contain. **c.** Save the update by selecting the **Apply** button.
Delete a list	**a.** Hover over the list you want to delete in the list navigation panel. **b.** Select the delete icon that appears ( ). **c.** Confirm the deletion in the modal that appears by selecting the **Delete** button.

6. Select the **Save** button in the Filtered list details panel to ensure all updates are saved.

Change the layout of an app's record in Creator Studio

Adjust how the records that your app generates will look, such as the order in which fields appear.

Before you begin

To change the layout of an app's record in the form submission workspace, you must be given permission to work on the app.

About this task

All columns of the app's Task table are available as fields on the record page, though you may want to adjust them. For example, you may want to move the **Priority** field to appear more visibly on the record so fulfillers can tell how important the request is. You could then build an automated playbook that runs when the priority changes to high.

Note:
You can't edit UI policies or make more advanced edits to the form in Creator Studio. To make advanced edits, open the record in Table Builder. For more information, see Forms in Table Builder.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the form submissions workspace you want to edit.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

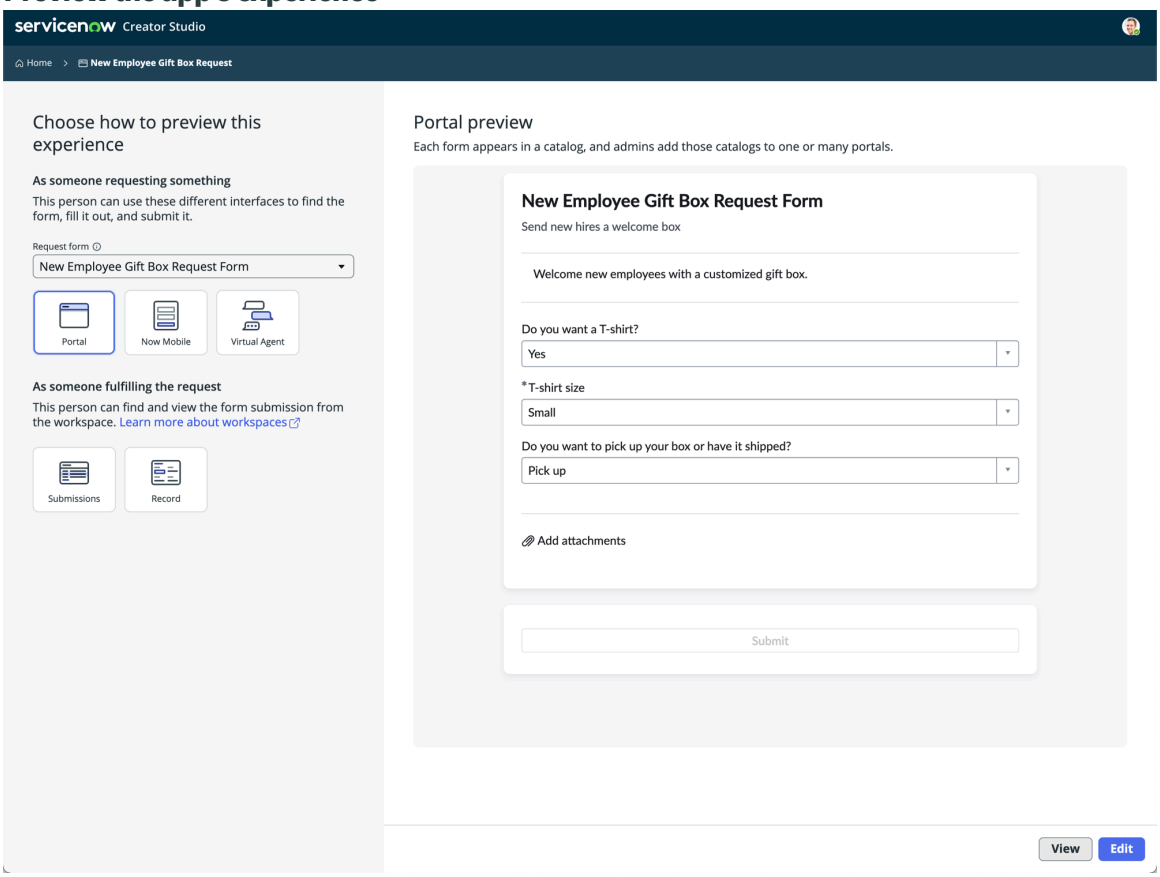
   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)

- **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

> **Note:**
> Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

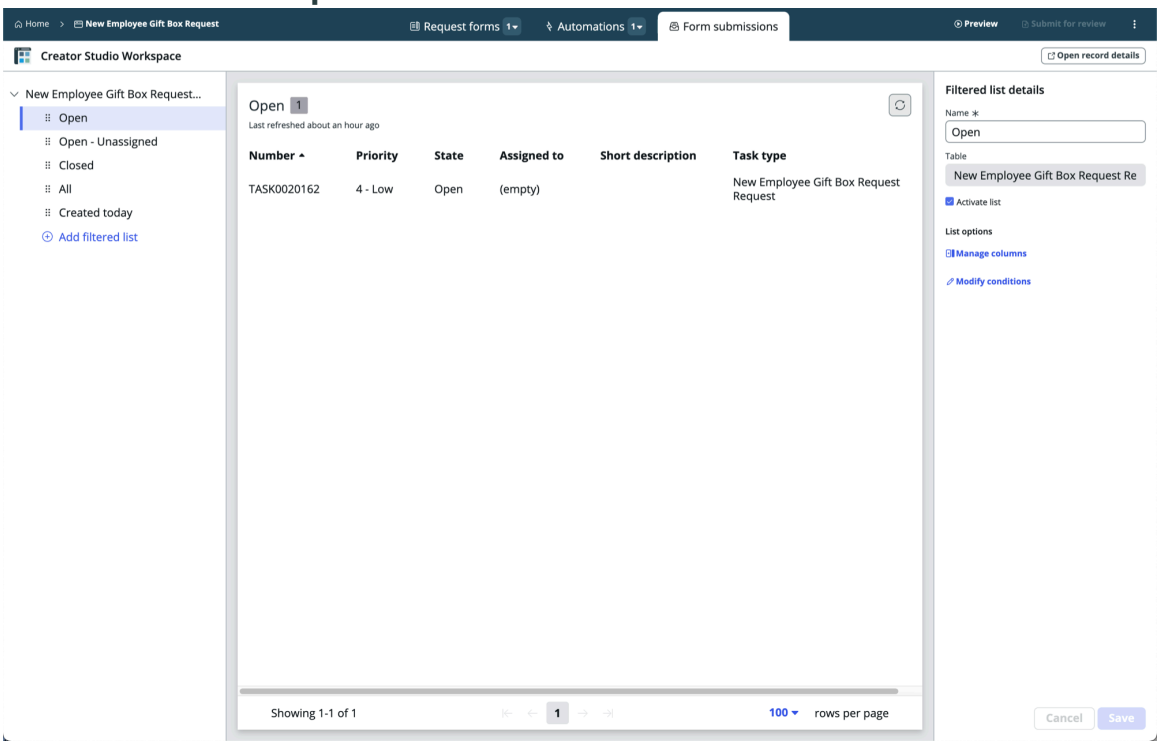
Preview the app's experience



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

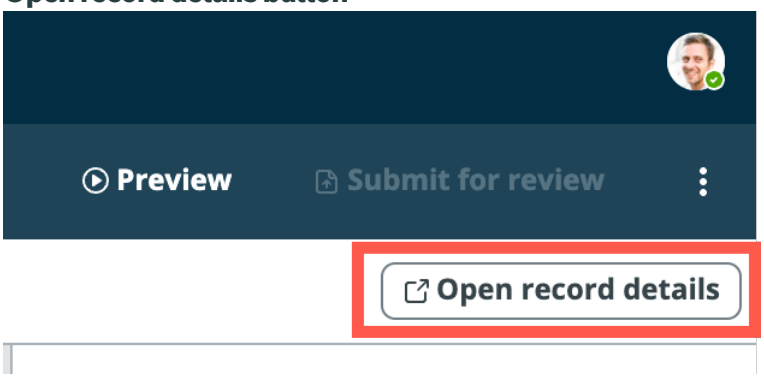
4. Select the **Form submissions** tab in the application header.

Form submission workspace in Creator Studio



5. Select the **Open record details** button.

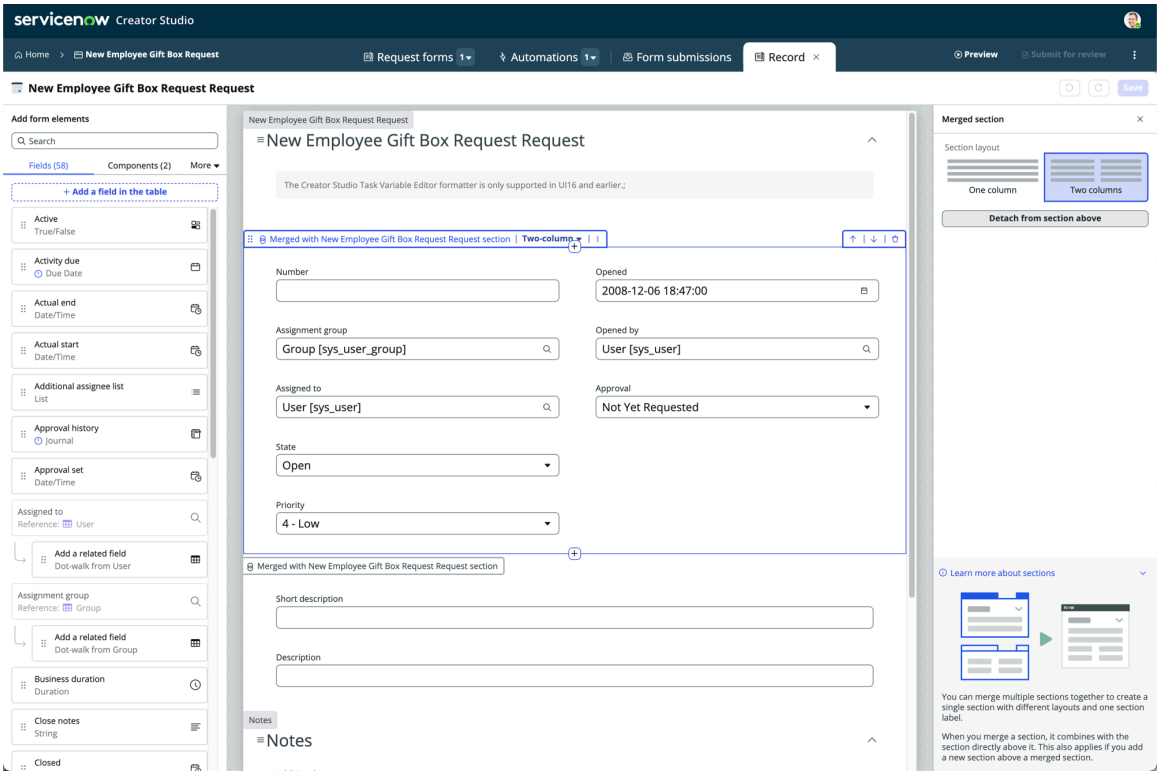
Open record details button



The Record details appear in a new **Record** application tab.

6. Make any additional changes to how the record appears, such as moving the fields around. Check out Important Task table fields for a list of the most commonly used fields that you can work with and their definitions.

Record details



7. Select **Save**.

What to do next

After you're done customizing how the app's records will appear, you can check out how they'll look in the Request App Workspace while remaining in Creator Studio. For more information, see Preview how an app's records appear.

Preview how an app's records appear

You can preview and interact with records generated by your app directly in Creator Studio, which mimics what the fullfiller sees in the Request App Workspace.

Before you begin

You can customize how the record will appear before testing it in a preview. For more information, see Change the layout of an app's record in Creator Studio.

To preview how an app's record will appear in the form submission workspace, you must be given permission to work on the app.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that contains the form submissions workspace you want to edit.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

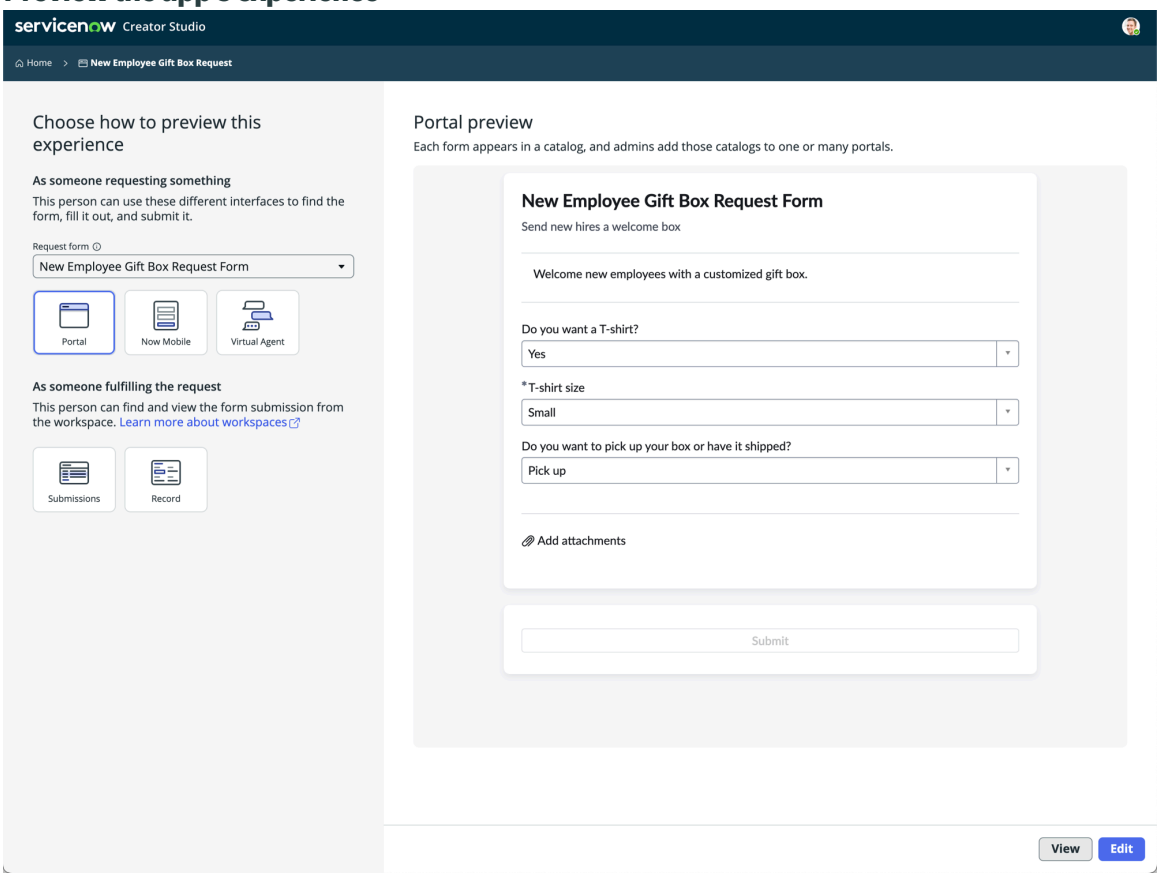
   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)

- **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

> **Note:**
> Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

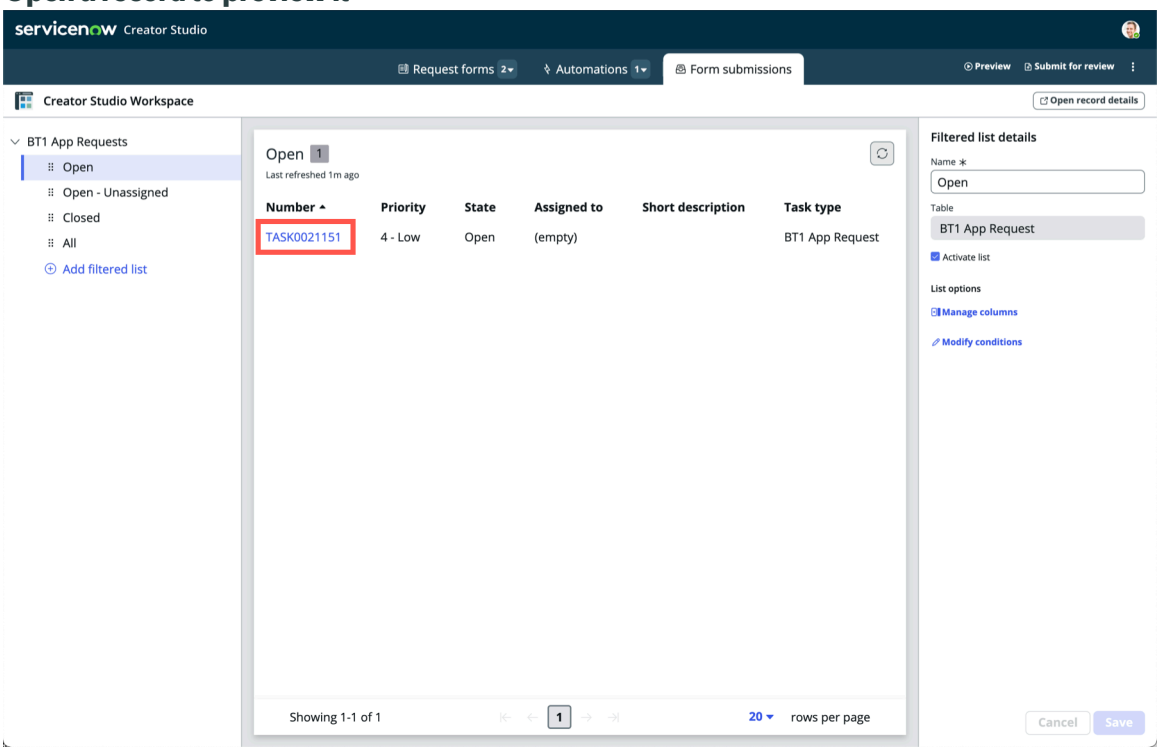
Preview the app's experience



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

4. Select the **Form submissions** tab in the application header.
5. Select a filtered list that contains a record, such as the *Open* records list.
6. Preview a record by selecting it in the canvas.

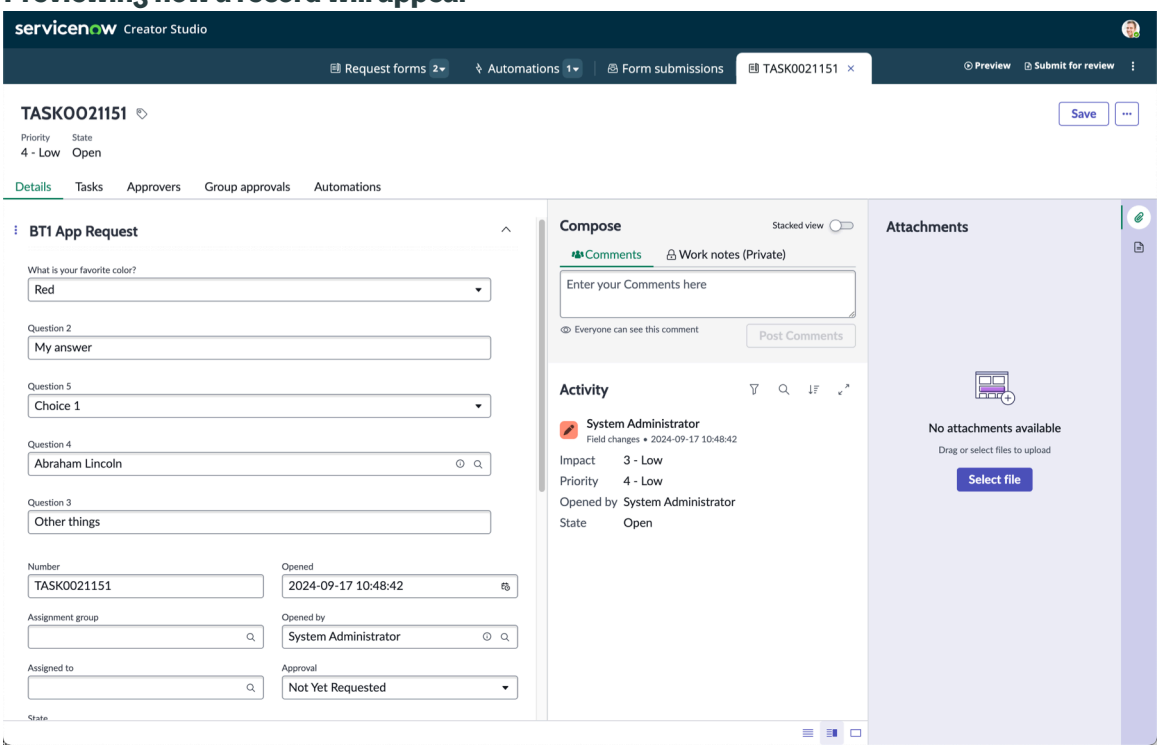
Open a record to preview it



The record opens in a new tab within Creator Studio.

7. Play with the record to see how its parts will appear.
   For example, you can select the record's **Approvers** tab to view how its approvers list will appear.

Previewing how a record will appear



Result
If you don't like what you see, you can tweak the record's appearance. Find out how in Change the layout of an app's record in Creator Studio.

Deploying your Creator Studio app

Congrats, you've built your app and are ready to share the first version of it with the world. That's great! Deployment is the next step in the process.

Deployment is a term that describes the process of an app moving from a non-production instance like development or QA through to production. Production is the instance that your customers see, or your "live" instance. To kick this process off, you should build your app in a non-production instance, and request that an admin deploy it to a production instance when you're ready.

Deploying an app automatically publishes the app, as well as all of the published forms and activated playbooks that you include, to a production instance.

Deployment uses pipelines configured in Pipelines and Deployments. If you don't have a pipeline configured, the Creator Studio home page alerts you and suggests you ask an admin to set it up for you.

The ServiceNow AI Platform uses the Application Repository to move apps between instances, so release notes and a version number are required. Find out more in App versioning and release notes for Creator Studio apps.

Requesting app deployment to production

You should test your app and all of its published forms and activated playbooks on a non-production instance. Once it's ready, you can submit the app for deployment to production. For more information, see Request deployment for your app from Creator Studio to production.

Apps aren't deployed directly from Creator Studio. Instead, your admin uses Pipelines and Deployments. Admins should check out Managing deployments using pipelines in AEMC.

Deploying forms and automation with the app

You may feel some hesitation at this point, perhaps wondering if your app is actually ready. Well, you can take small steps and include only the forms and activated playbooks that you want in your deployment request.

When you submit an app for deployment, all of the app's published forms and activated playbooks are available to be deployed. When you request deployment though, you can specify:

- Published forms that you don't want to appear in the catalog after the app is deployed to production.
- Activated playbooks that you don't want to appear when users view records generated by the app after it's deployed to production.

If you don't activate a playbook and its app is deployed to production, the automation won't run on the applicable records. However, your App Engine admin can activate the deployed playbook for you.

Assigning users to a deployed app

Your app is officially published and ready to be used, congratulations! But, who gets to use it, and do they have access? Your admin gets to take charge here.

After your admin deploys the app to production, the admin must assign users and groups that can access the app. That is, you can't assign user access in the deployment/publishing process. See I've built my app in Creator Studio, now what? to find out more about where to access published parts of your app.

App versioning and release notes for Creator Studio apps

Each deployed version of an app must have a version number and release notes, which helps administrators track app usage and changes.

App version number

The version number enables admins to track which version of the app is deployed to each instance. Follow your organization's versioning guidelines, or use the x.y.z format, where x = major update, y = minor update, and z = patch.

Creator Studio automatically updates the next version when you update a deployed app, for example from 1.0.0 to 1.1.0 and then to 1.2.0.

App release notes

The release notes help admins track what the changes are between app versions. When submitting your app for deployment, specify information that helps people understand differences between app versions.

Request deployment for your app from Creator Studio to production

After you've tested your app's forms, playbooks, and workspace category on a non-production instance, for example a development instance, deploy it to a production instance so users can access it.

Before you begin

Your admin must set up a deployment pipeline before you can submit your app for review. Contact your admin if you need a pipeline configured.

To request that your app is deployed to production, you must be given permission to work on the app.

About this task

You can't cancel the review cycle after you submit an app for deployment review, so make sure that your app is ready to go.

Procedure

1. Go to **All** > **App Engine** > **Creator Studio** to see all the apps on the Creator Studio home page.
2. Open the app that you want to deploy to production.
3. Select to **Preview** how different forms appear in various experiences, or select the **View** or **Edit** button to view or edit the application experience.

   Select different previewing options if you want to make sure that you've selected the correct application and see how it will appear in the following formats:

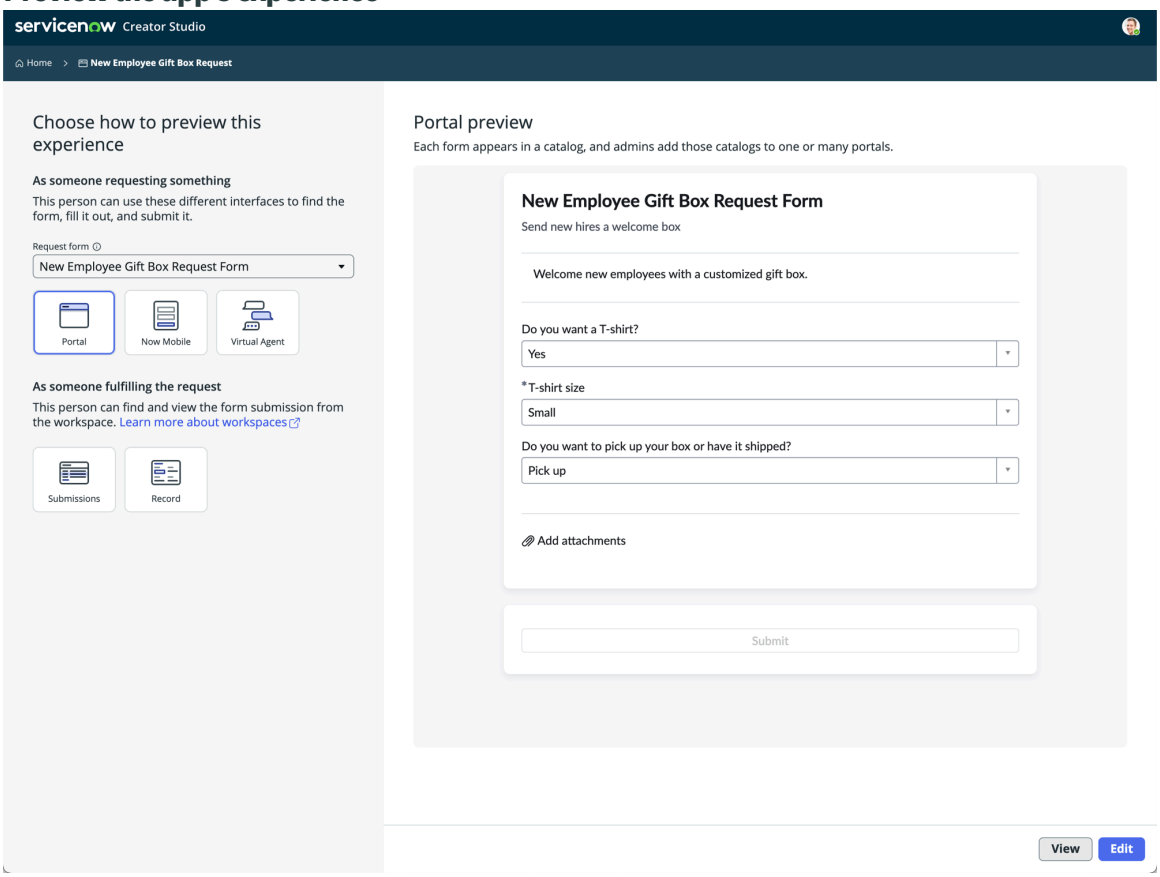
   - **Portal** (see a preview of how it'll appear on a desktop website)
   - **Now Mobile** (see a preview of how it'll appear on a mobile phone or device)

- **Virtual agent** (see a representation of how it'll appear on a chatbot interface)

> **Note:**
> Your organization should have the correct plugins installed to see how the form will appear in Virtual Agent. If you're interested, ask your admin and see Catalog builder preview topic conversation for more information on previewing forms and their catalog items in Virtual Agent.

The **View** button displays forms that have been published and doesn't explicitly create a new draft form for development. The **Edit** button takes you to a development form, for example, a new draft version of a form that's already been published.

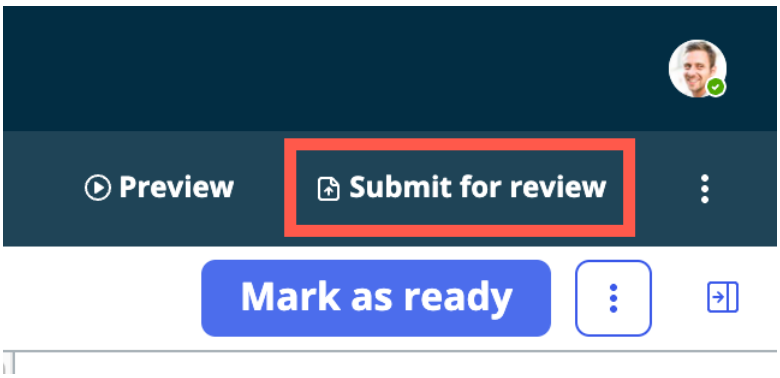
Preview the app's experience



You can also check out a representation of how the form submissions workspace will appear by selecting the **Submissions** preview, as well as the records your app generates for it (by selecting the **Record** preview).

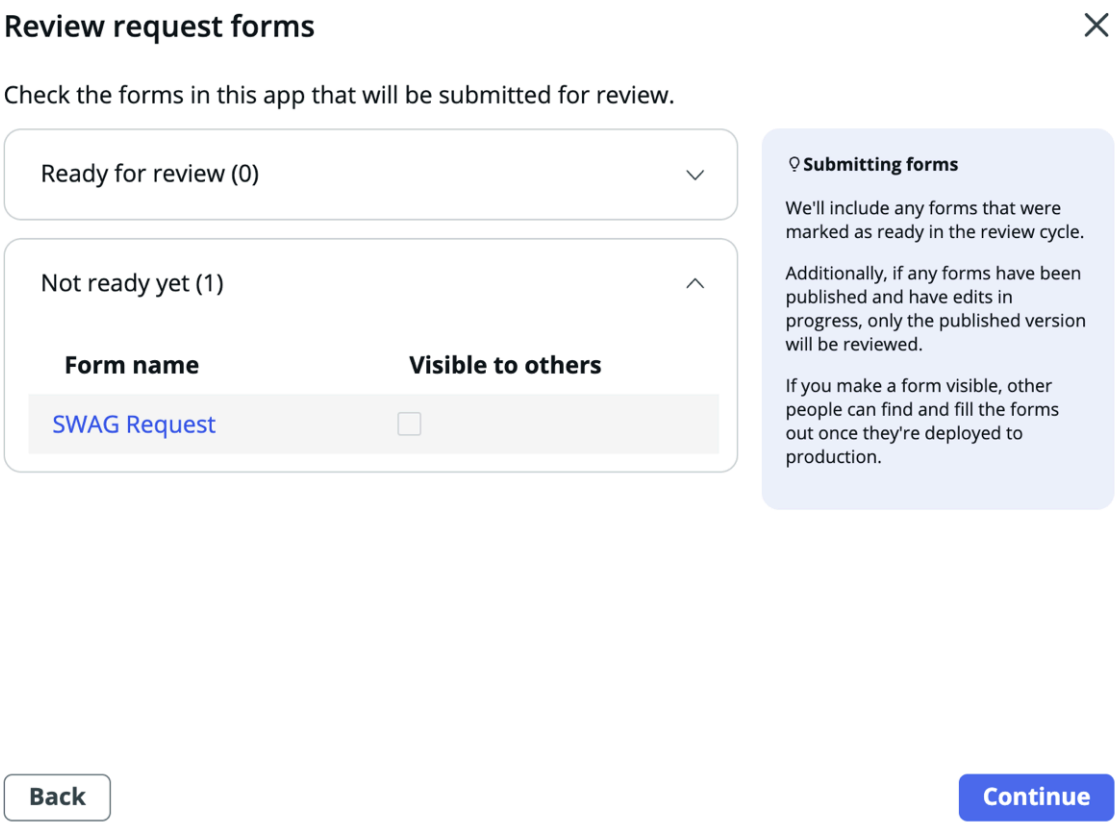
4. In the application header, select **Submit for review**.

Submit for review



5. Select **Continue** on the Submit app for review modal.
6. Now you need to choose which published forms are visible to users in the catalog. In the Ready for review section of the Review request forms modal, select which of the app's published forms that you want to be available after the app is deployed selecting the **Visible to others** option.

Review request forms for deployment



7. Select **Continue** when you're happy with the forms being deployed to production.
8. Next, you must decide which of the app's activated playbooks will run on production after the app is deployed. In the Review playbooks modal, select the **Run on production** option for each playbook that you want to run on records that the app generates.

Note:
If you can't select a playbook, you need to go back to the **Automations** tab of Creator Studio and activate it. If you need a refresher on that, check out Activate a playbook in Creator Studio.

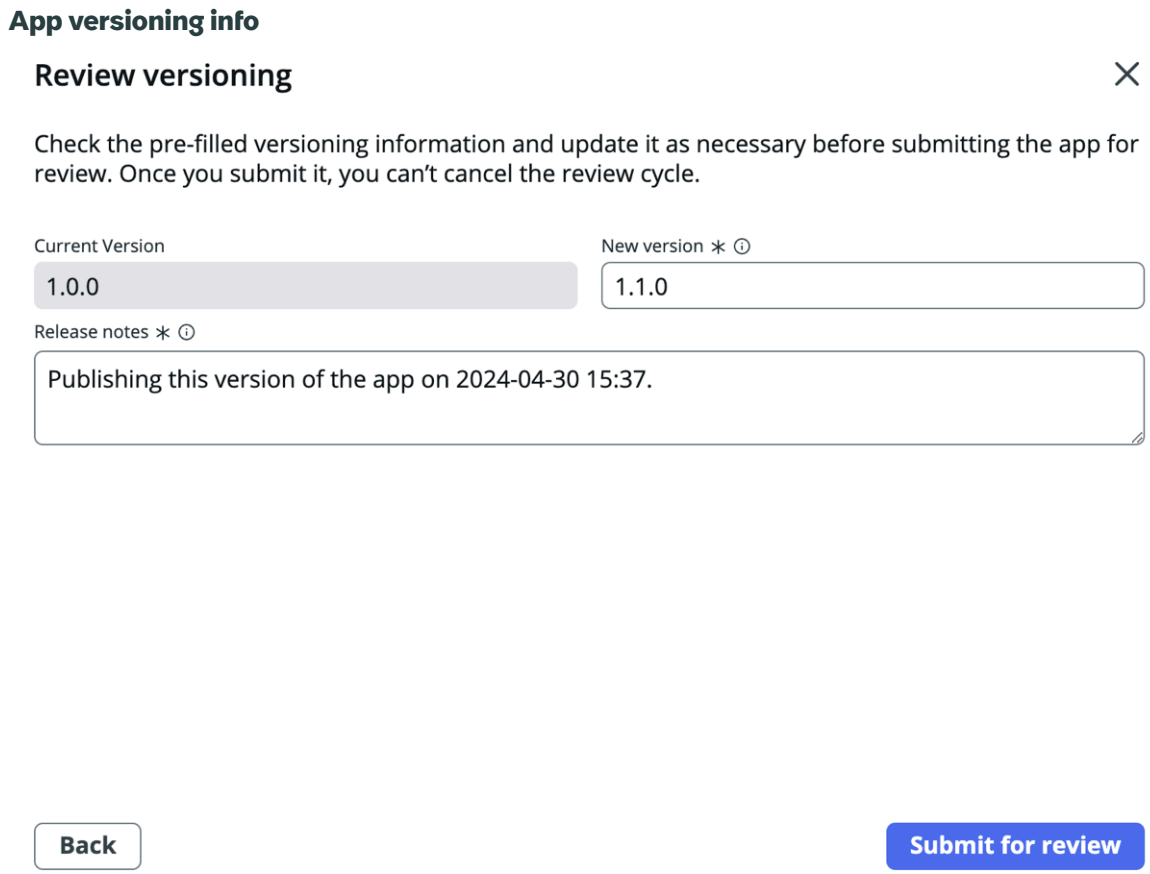
Unactivated playbooks still get deployed to production with the app, they just won't run on the records unless your admin activates them on production or redeploys the app with the playbook activated.

9. Select **Continue** when you're happy with the playbooks being deployed to run on production.
10. Finally, make sure that all the release details for the published app are correct.

Versioning options for deployment

Field	Description
New version	Version number of the app you're requesting for deployment. Creator Studio automatically generates an updated version number, but you can change it. Follow your organization's versioning guidelines, or use the x.y.z format, where x = major update, y = minor update, and z = patch.
Release notes	Details on what's changed in this new version of the app, or a general description of what the app does if this is its first version.

Read more about this step of requesting deployment in App versioning and release notes for Creator Studio apps.



11. Select **Submit for review** when everything is correct and ready for your admin to review and deploy.

Result
Woohoo! Your app is ready to be reviewed for deployment.

What to do next
Your admin uses Pipelines and Deployments to deploy the app to production. You can still make changes to the app after it's been deployed, you'll just need to request re-deployment when it's ready.

If you're an admin, check out Managing deployments using pipelines in AEMC for the scoop on deploying apps.

I've built my app in Creator Studio, now what?

You've added forms and automation to your app, and customized the form submission workspace. Now what?

What happens after I submit my app for review?

When you finish creating your app and select to **Submit for review**, your admin gets a deployment request. They then review your app deployment request and move the app to production.

After the app is deployed to production, the forms are available on the ServiceNow AI Platform.

Where do I find my app after it's deployed?

After deployment, your app lives as forms in the Service Catalog and categories you specified when creating the forms.

Users can access those forms directly in Service Catalog, as well as Service Portal and Employee Center.

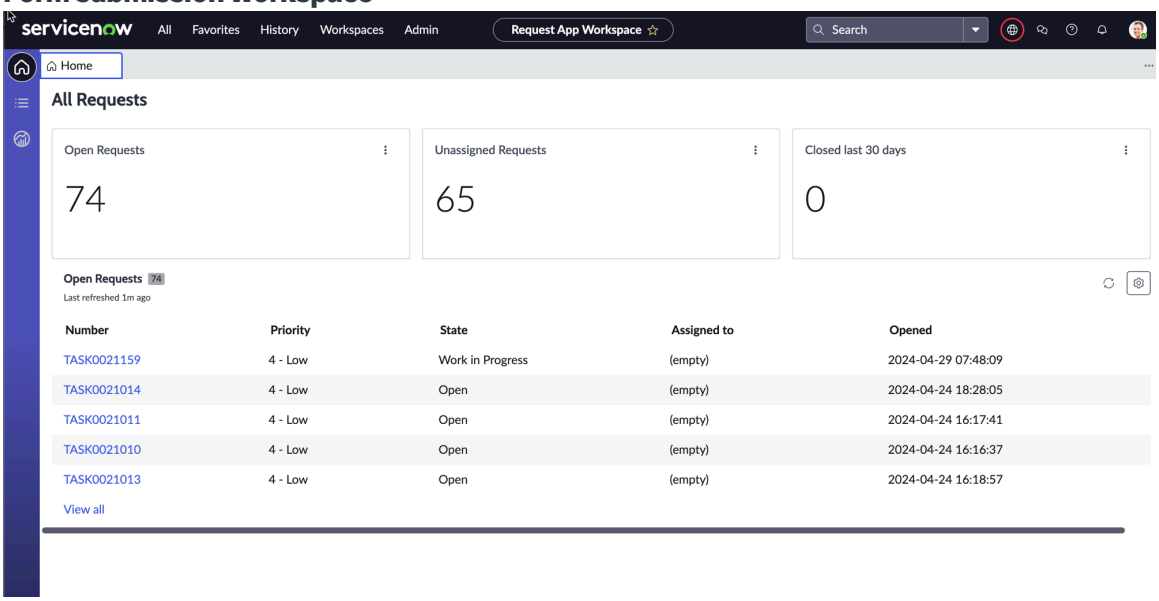
If you associate the app's form with one or more topics, the form will appear in the relevant, dynamically created topic pages in Employee Center. Find out more about topics in Associate a catalog item with a taxonomy topic in Employee Center, and more about taxonomy, which is a categorization method, in Unified Taxonomy for Employee Center.

Where do fulfillers work on submitted forms?

Fulfillers access the default workspace on the ServiceNow AI Platform by going to **All** > **App Engine** > **Request App Workspace**. The Request App Workspace displays a category for each app with filtered lists of records, both default and any extra that you added. Fulfillers then select a record to view the details of the request and make updates as needed.

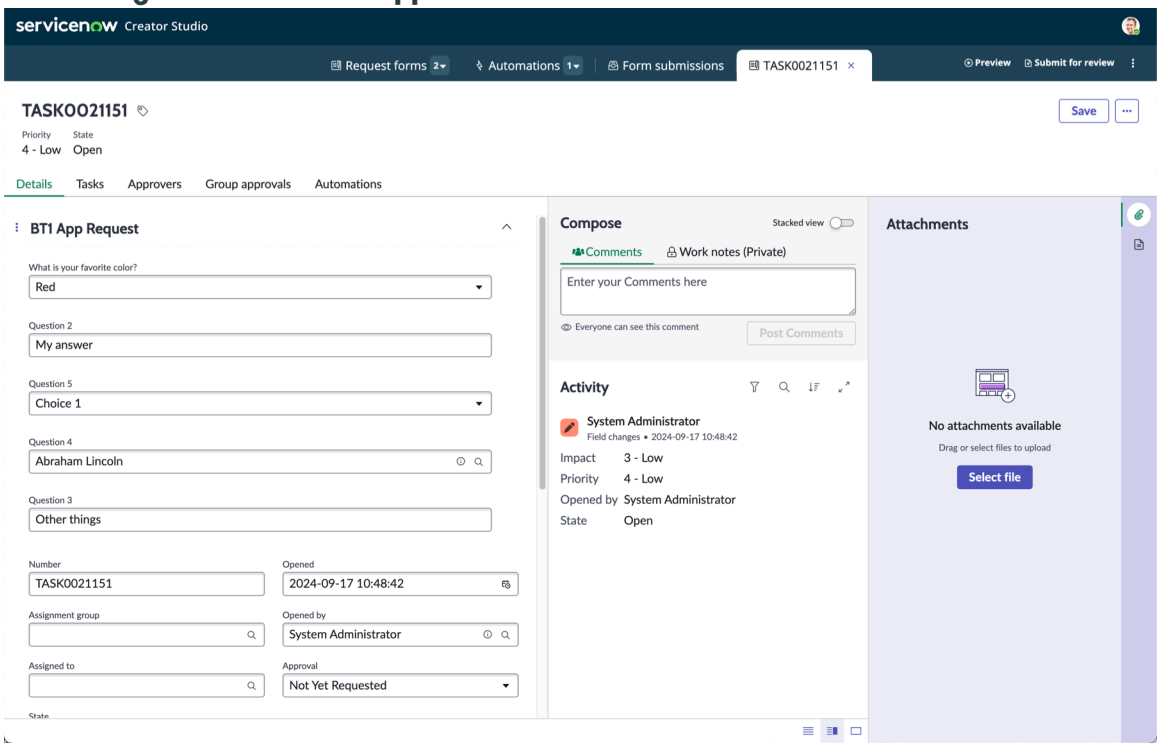
Note:
You can't access the Request App Workspace as the owner of an app, though you can access it if you also have an admin or the applicable agent role. The app's agent role is <app scope>.agent, for example, x_snc_app_name.agent.

Form submission workspace



You can also preview and interact with how records will appear in the Request App Workspace directly within Creator Studio.

Previewing how a record will appear



Can I continue developing my app in other ServiceNow AI Platform builders?

You can open Creator Studio apps in other builders. For example, open it in App Engine Studio to add more experiences or complex automations.

Get help and check out the FAQs

Got more questions? Take a look at the Creator Studio FAQs on the Community site !

You can also watch a quick video on how to contact your admin and check out some helpful resources.

Closing requests and app notifications in Creator Studio

The records that your app creates when a form is completed are not automatically closed, and some notifications are not automatic and must be configured.

Closing requests generated by Creator Studio apps

After an app's automation is done and the request is fulfilled, users must manually close the request record in the Request App Workspace. When fulfillers manually close the request, the ServiceNow AI Platform emails the person who made the request to notify them that the request is closed.

Additionally, you can create a playbook in the app to automatically email the requester when their request is closed. For more information, see Add activities to an app's playbook in Creator Studio.

Default notifications for apps built with Creator Studio

Apps created in Creator Studio send the following notifications by default.

Default app notifications

Event	Recipient	Trigger
Request opened	User who opened the request from the app	Record in the app's Task table is created
Request assigned to user	User the request is assigned to	Record in the app's Task table is created or updated
Request assigned to group	All users in the group that the request is assigned to	Record in the app's Task table is created or updated
Request closed*	User who opened the request	Record is updated any closed stated, such as Complete or Cancelled

*Remember, request records must be manually closed.

For more information on notifications from the ServiceNow AI Platform, see Receive notifications.

Notifications Creator Studio sends about building apps

Default app notifications

Event	Recipient
A Creator Studio Restricted User submits a new app request	App Engine Admin group
The requested app is approved and created by the admin	Creator Studio Restricted User who requested app creation
The app creation request has been fulfilled	App Engine Admin group

Creator Studio reference

Find details on the parts of building apps in Creator Studio, such as descriptions of all possible types of questions or layout options for forms.

Creator Studio form settings

You can edit the settings for an app's form in Creator Studio at any time. For example, to change who can access it.

General form settings

The **General** tab defines the form's basic information and how things appear on it.

General tab for app settings

Field	Description
Form name	Name of the form. Apps can have multiple forms. If you want to edit the settings for a different form, select that form in the **Request forms** tab.

General tab for app settings (continued)

Field	Description
Short description	Brief description of what the form is used for. For example, "Laptop request form."
Image	Image to appear on the form. Select the logo and navigate to a JPG, PNG, or SVG file to upload.
Description	Informative reason why people should use the form. You can include content like videos, images, and links to other sources of information, such as knowledge base articles.
Make form visible to others	Option to have the form available outside of the app, for example, in the specified catalogs. You can edit this option only for forms that have been published. **Note:** You can also specify whether the form is visible after deployment when you're ready to deploy your app. Find out how in Request deployment for your app from Creator Studio to production.
Hide "Add to wish list" option	Option to prevent users from favoriting the form on their wish list. Favoriting a form makes it easier to find.
Hide attachment option	Option to prevent users from adding attachments to the form.
Mark attachment as mandatory	Option to require attachments on the form. If you select the **Hide attachment option** setting, you can't select this setting.

Uneditable form settings

After you create the app, its **Template** can't be edited. If you want to change it, create a new form using a different catalog template. The **Record submission table** is automatically generated, and can't be changed.

Catalog location form settings

Location settings define where the form appears in a catalog, how it's categorized, and what topics it appears for. You must select the edit icon (Edit) to make changes to the catalog and topic settings.

Select the catalog that represents the business area the app will use. For example, you could select a service catalog that contains software and laptop cables for an IT fulfillment app. Expand the carat for each catalog to see its sub-catalogs. Then select as many items in the catalogs as you need.

If you associate the app's form with one or more topics, the form will appear in the relevant, dynamically created topic pages in Employee Center. Find out more about topics in Associate a

catalog item with a taxonomy topic in Employee Center, and more about taxonomy, which is a categorization method, in Unified Taxonomy for Employee Center.

User access form settings

Select the edit icon (Edit) to define access for roles and groups for the following:

- **Available for**: Search for and select roles and groups that can access the form.
- **Not available for**: Search for and select roles and groups that can't access the form.

Available question types in Creator Studio

Build apps using different types of questions on a form in Creator Studio.

You get to pick the questions to ask requesters. Creator Studio provides two types of questions you can ask:

- **General questions** are where you give requesters an empty box to write a short answer (think name or email) or a list of choices to pick from (like a dropdown menu or those little circles you fill in).
- **Reference-based questions** are where you provide a list of choices that you define in a table.

The type of question you select from the **Content type** field in the Question details panel changes the question's settings that appear.

Types of general questions

General question type form elements

Content type	Description
Short answer (Single-line text)	Single-line question.
Multi-line text	Longer option that provides space for a question multiple lines in length.
Dropdown	Question with a list of pre-defined answers that users select from a dropdown menu. Users can select only one answer.
Radio button	Question with list of pre-defined answers that users select from radio buttons. Users can select only one answer.
Check box	True or false question that users answer by selecting the relevant check box. **Note:** If you put two checkbox questions side-by-side on a form, they make a section. You can't add other types of questions to that section.
Yes/No	Question where users select Yes or No as their answer.
Date	Question where the answer is a date that users must select from a calendar.

General question type form elements (continued)

Content type	Description
Date and time	Question where the answer is a date and time that users must specify.
Question set	Preconfigured questions that you can't change. Admins create question sets in Service Catalog. For more information, see Service catalog variable sets. **Note:** • You can't change an existing question into a question set. To include a question set on a form, you must newly add it to the form. • If you put two checkbox questions side-by-side on a form, they make a section. You can't add other types of questions to that section.

Types of reference-based question types

Reference-based questions pull their possible values from the table that you must designate as the **Source table**.

Reference-based question form elements

Content type	Description
Record choices	Question where the answers are retrieved from an additional specified table when the filter conditions, which you must also define, are met. Use this question on a form that references data from another form, such as a form that uses the Incident table. "Choose the related incident" is an example Record choices question, which could use a filtered list of incidents as possible answer selections.
Field choices	Question where the answers are retrieved from a field on the specified table when the filter conditions, which you must also define, are met. For example, you could create a Field choices question for a table with a "Country" field, which the user could then use to select their country.

Reference-based question form elements (continued)

Content type	Description
Multi-select	Question where users can select multiple answers that are retrieved from the specified table when the filter conditions, which you must also define, are met. Use this question on a form that references data from multiple other forms, such as a form that pulls answers from multiple separate Incident tables.

Question details

Depending on the type of question you add to a form, you must specify some of the following details.

Question type details

Field	Description	Available for
Question label	Question or statement that says what the question is asking.	All question types
Help text	Additional information that appears as a tooltip to guide the user when they hover over the information icon.	All question types
Show questions on form	Hide the question from the form by leaving this option unselected.	All question types
Mark as required	Option to make the question required before the user can submit the form.	All question types
Mark as read-only	Option to make the question read-only. You can use this option to associate data with the form behind the scenes by adding it to the app's Task table. For example, if you don't automatically pull the user's role into the form's Task table, you could create a single-line text question where the default value is their role (such as Sales) and then hide it. You could even hide the question from the form to make the data exchange invisible to the user.	All question types

Question type details (continued)

Field	Description	Available for
Default value	Value that's auto-populated in the question's answer when the form is opened. For reference-based questions where you designate a source table, you must use an asterisk (`*`) as a wildcard when searching for values in the **Default value** field. For example, if the question references the User table and you're looking for Abel Tuter, entering `Tuter` won't find that user but entering `*Tuter` will.	All question types
Choices	Customizable answers to the question, which you can drag to rearrange in a new order.	• Dropdown • Radio button
Include "None" as a choice	Option to include **None** as a possible answer to the question.	• Dropdown • Radio button • Field choices
Default to the first choice	Option to display the first choice as the possible answer when the user sees the question.	Radio button
Source table	Table location that provides data for answers to the question.	• Record choices • Field choices • Multi-select
Field	Field in the specified source table that possible answers are retrieved from.	Field choices
Filter conditions	Conditions that must be met to view certain choices. Select **Add condition** and use the condition builder to create the filter. For more information, see Condition builder.	• Record choices • Field choices • Multi-select

Layout options for forms in Creator Studio

Adjust how your form, which creates a catalog item, appears in Creator Studio using layout options.

Layout options enable you to design a form by dragging them from the Form elements panel to the form canvas. You can also drag layout options to rearrange them.

Note:
To edit or delete a section, you must hover over the section name and then select **Section** to see the section details in the properties panel, as well as the delete icon.

Selecting a section

Layout options

Element	Description	Details
One-column section	Section with optional header (label) where all questions in that section of the form appear in a single column.	• Specify a **Section label** to give the section a header. • Select **Show section label on form** to display the section header.
Two-column section	Section with optional header (label) where all questions in that section of the form appear in two columns.	• Specify a **Section label** to give the section a header. • Select **Show section label on form** to display the section header.
Divider line	Displays a line between questions or sections.	N/A
Heading	Displays header text, such as a question or a statement based on what type of information you're asking the user to provide on the form.	Specify header text in the **Name** field.
Rich text	Displays longer text that you can enhance with options like links, different fonts, and bullets and tables.	Use the rich text editor to enhance any supplemental text with the following typographic formatting: • Bold/Italics/Underlining • Font face and size

Layout options (continued)

Element	Description	Details
		• Font and background color • Tables • Links • Images • Source code to include scripting • Alignment • Bulleted list • Numbered list

Creator Studio glossary

Learn about the terms and concepts used in Creator Studio.

activity

Activities in a playbook complete automated tasks, such as obtaining approvals, assigning records, and sending email.

automation

Tasks or requests moving through a workflow without manual intervention.

board view

A traditional layout of viewing a playbook, which shows each activity as a card in the navigation panel. When you select an activity, for example a task, its editable details appear in the canvas for you to work with them.

catalog

A collection of items that represent all the items a requester can ask for.

collaboration descriptor

Level of collaborative access assigned to a developer. The collaboration descriptor that someone is assigned determines if they can assign, manage, and monitor delegated development permissions. For example, people who are Owners of an app can do more than people with the Editor collaboration type.

delegated development

A collaborative process that enables developers to invite other people into apps so they can co-create and develop the app together.

deployment

The process of an app moving from a non-production instance like development or QA through to production. Production is the instance that your customers see, or your “live” instance.

diagram view

A visually intuitive diagram that shows you the order that activities occur in the playbook. When you select an activity, such as the trigger or a decision, a modal pops up where you can edit its details.

dynamic behavior

A form for a catalog item can use dynamic behavior to automatically update based on how users answer a question. For example, if a user says they don't want lunch for an event they're attending, you can make a **Meal preference** field disappear.

> **Note:**
> Dynamic behaviors are also called Catalog UI Policies.

extend a table

When you create a table based off an existing table. Extending a table means the new table inherits the parent (extended) table's columns, as well as its business logic. Apps built in Creator Studio extend the Request Task table.

field

A field in Creator Studio is a column on the related Task table. Ii you're using condition builder, a field is either a column on the table or a question from the form.

form

A content page that displays fields and values for a single record from a database table.

form template

A pre-defined set of questions arranged on a form provided by Creator Studio.

fulfiller

Someone who works on requests. Fulfillers may also approve or deny requests, depending on any approval automation for the app.

item

Something a requester can ask for, such as new computer equipment or time off work. Each item requires its own form in Creator Studio.

playbook

A set of automated activities that occur based on a trigger, for example, sending an email when a request is approved. Think of playbooks as pre-programmed workflows that kick in after a form is submitted. They are like if/then statements: If this happens, then that will happen.

record

A task or request that needs to be acted on. All rows in your app's tables are individual records. Records have different states depending on what actions have been taken.

reference-based question

Question on a form that retrieves its possible answers from the table you define.

release notes

Information about what's included in each new version of the deployed app, which helps admins track what the changes are between app versions.

requester

Someone requesting something, like a piece of equipment or permission to do something, such as take time off work.

target table

The table where apps created in Creator Studio store their requests.

trigger

The event that makes an automated playbook begin running. For example, submitting a request automatically assigns it to a manager.

version number

A number assigned to the app that increases with each subsequent time it's deployed.

Workflow

A sequence of activities to automate processes in applications.

Building low-code applications

Create and manage custom applications in a simple, low-code environment.

ServiceNow Studio

Use ServiceNow Studio to build apps and app files with integrated tools, access and edit metadata in scoped and global apps, and package app changes for deployment, all in one powerful development tool.

App Engine Studio

Use App Engine Studio for guided, low-code development to build a custom app from scratch or with a template.

Creator Studio

Use Creator Studio to quickly create a request/fulfill app based on Service Catalog using helpful templates.

ERP Data Hub

Use ERP Data Hub to connect to an ERP (Enterprise Resource Planning) system, create an ERP model, and extract data from the ERP system, such as sales orders, data records, and purchase documents. You can then use the ERP data when building low-code apps on the ServiceNow AI Platform.

ERP Customization Mining

Use ERP Customization Mining to find the custom applications in your ERP system for platform migration, such as from SAP to the ServiceNow AI Platform.

ServiceNow CLI

ServiceNow CLI enables you to create custom commands to manage applications from your local system's command line.

Which builder should I use to create an app?

Types of builders

Want to build an app easily, without code?

Creator Studio specializes in helping you craft request-fulfillment applications without writing code. For example, an application to request office supplies by filling out a form, and someone approves or denies your request. For more information, see Exploring Creator Studio.

Need a more general app but still want low-code options?

App Engine Studio lets you build a broader range of apps than Creator Studio without being a programming pro. For more information, see Exploring App Engine Studio.

Are you a developer who wants more control in a centralized user interface?

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity. For more information, see Exploring ServiceNow Studio.

Are you a developer who wants to use industry-standard development tools and processes?

The ServiceNow IDE and ServiceNow SDK support developing applications in source code with ServiceNow Fluent, creating JavaScript modules, and using third-party libraries. ServiceNow Fluent is a domain-specific programming language for creating application metadata in code.

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. The ServiceNow SDK uses Visual Studio Code Desktop locally. For more information, see Building applications in source code.

Related applications and features

Exploring decision tables

Use Decision Builder to create and manage decision tables. Embed business logic into a series of if-then decision rules.

ServiceNow Studio

ServiceNow Studio gives developers access to app development builders and tools, all in one place.

Flows in Workflow Studio

Workflow Studio automates processes and repetitive work to improve efficiency and experience.

Table Builder

Use Table Builder to design tables, forms, and flows visually using a single user interface.

UI Builder

Use UI Builder to build web user interfaces for CSM Configurable Workspace, App Engine Studio generated workspaces and portals, or custom web experiences using Next Experience Components and custom web components.

Workflow Studio

Integrate workflow authoring, configuring, and monitoring into a single page experience.

Workspace Builder

Workspace Builder for App Engine enables you to create a custom workspace from App Engine Studio (AES) quickly and efficiently.

Build apps using App Engine Studio

ServiceNow® App Engine Studio (AES) is a guided, low-code tool for developing rich web applications to store information, automate business processes, and solve business problems. Work that was once assigned to administrators can now be delegated to employees with little to no training using AES.

This video shows a visual overview of the app creation process using App Engine Studio.

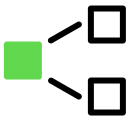
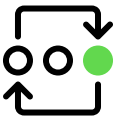
App Engine Studio overview

Get started

By delegating development to citizen developers in your organization, administrators are freed up to address more strategic, system-wide priorities. AES can help you do the following:

- Build powerful apps quickly to fit any business need by using a suite of development tools, including guided workflows and templates.
- Create experiences users love with Catalog Builder, Workflow Studio, Workspace Builder, Mobile App Builder, and other integrated tools.
- Scale apps without sprawl with straightforward governance and tools that support both citizen and pro developers, such as App Engine Management Center (AEMC).

Explore	Install	Configure
		
Learn about App Engine Studio concepts and features.	Install App Engine Studio and its associated apps.	Configure environments, tools, and user access.

Build	Deploy	Reference
		
Build applications using App Engine Studio.	Deploy applications to the instances in your pipeline, from development to production.	Get details about App Engine Studio components such as fields, tables, and properties.

Additional resources for App Engine Studio

Learn more about App Engine Studio	ServiceNow resources
App Engine Studio is a guided, low-code tool for developing rich web applications to store information, automate business processes, and solve business problems.	 Create Apps Fast with App Engine Studio  Build Apps Fast with App Engine  Getting Started on App Engine Studio

Request AES on the store

Visit the ServiceNow Store website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Before you can use AES, you must first download it from the ServiceNow Store. When you have completed the download, you may need to clear your local cache before it appears on your instance.

Get help with App Engine Studio

- Contact your company's Customer Admin to unlock or add user accounts, perform restores or zBoots, and more.

  Contact your company's Customer Admin

- Ask or answer questions in the App Engine Studio community forum
- Search the Known Error Portal for known error articles
- Learn more about how to create your own apps on the developer site.
- Contact Customer Service and Support

Exploring App Engine Studio

Learn how you can use the low-code app development tool, App Engine Studio (AES), to build powerful applications using guided setup, predefined templates, and workflows.

This video shows a visual overview of the app creation process using App Engine Studio.

App Engine Studio overview

App Engine Studio users

Users

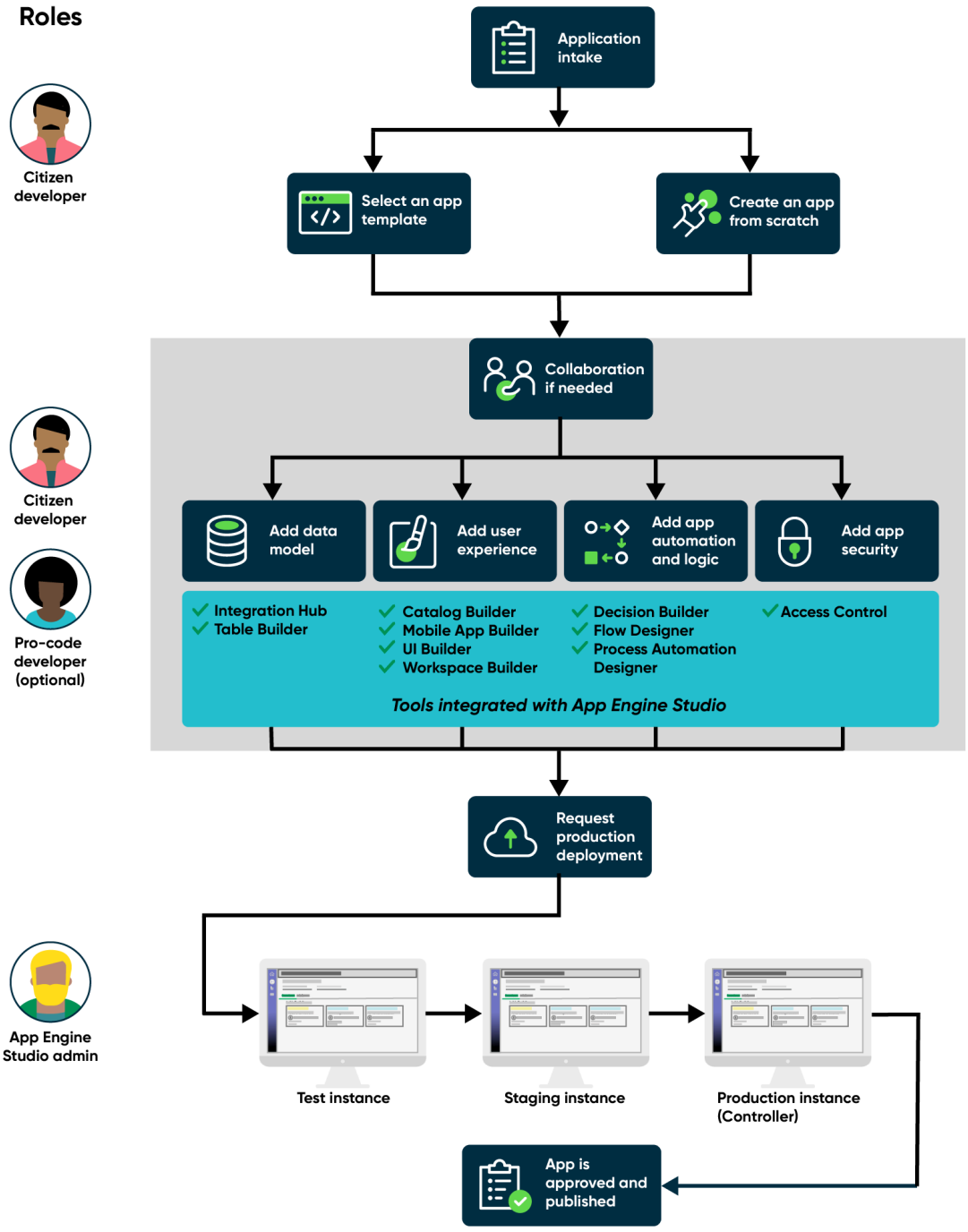
User	Description
Citizen developer	Citizen developers without programming experience can efficiently create applications on a simplified low-code, no-code platform that uses graphical interfaces and templates. The lower barrier to entry means developers of all skill levels can build applications.
Pro-code developer	Pro-code developers are familiar with using programming languages to build applications.
App Engine Studio admin	App Engine Studio admins manage the processes related to application development in AES. They review new application ideas, manage application development and deployment, and manage collaborator access, usually in the App Engine Management Center.

Workflow to create apps using App Engine Studio

The following illustration describes the basic tasks involved in creating an app using App Engine Studio. For detailed instructions, see Building apps in App Engine Studio.

Note:
Select the image to enlarge it.

Application creation process using AES



The workflow for creating apps in App Engine Studio is as follows:

1. As a citizen developer, you submit your plan for an app through Application Intake for approval and development in AES.
2. The AES admin reviews the intake request. If it's rejected, that closes out the request. But you can always update your idea and submit a new request.
3. When the request is approved, you must decide whether to create your app using a predefined template, or to create it from scratch. You should explore the available templates. They can save you lots of time.
4. Then you must determine whether you have the resources to build the app on your own. If you need help, you can request collaboration, or the AES admin can assign a collaborating developer.
5. If you use a template, the base system app is created automatically, with a basic data model, user experiences, automated workflows, and security roles. For more

Additional resources for App Engine Studio

Learn more about App Engine Studio	ServiceNow resources
App Engine Studio is a guided, low-code tool for developing rich web applications to store information, automate business processes, and solve business problems. **Note:** AES requires an App Engine subscription or product packaged with one. For more information, refer to Installing App Engine Studio. Contact your account representative for details.	 App Engine Studio Playlist  App Engine Studio vs. Studio  ServiceNow Community site  Build My First App Engine Studio Application - ServiceNow Developers site  App Engine Migration Assessment

Get to know AES

AES is designed to address enterprise low-code development needs by putting the power of the ServiceNow AI Platform in the hands of low-code developers. Guidance-driven flows and easy-to-customize templates help citizen developers to come up to speed quickly.

Regardless of their skill level, citizen developers become more productive using AES. Whether you're building a custom app from scratch or using a template, the simplified low-code experience in AES speeds up development on the ServiceNow AI Platform.

Key features of the AES include:

- **Guided experience for first-time users:** AES features a guided experience that orients developers, making it easy to navigate the environment. The first time you use the guidance, you're stepped linearly through the entire process of creating an application. After that, the guidance displays assistance related to what you're working on. You can hide the guidance by selecting **Don't show me this again** and exiting the pop-up.
- **App templates:** App templates simplify the app development process. Using templates based on proven use cases gives developers the ability to create powerful apps without the learning curve.

- **Guardrails:** Citizen developers receive tooltips and recommendations along the way, keeping them on track. Submitting apps for approvals also ensures quality.
- **Collaboration:** Developers of all skill levels can seamlessly collaborate on app development in AES's low-code environment.
- **Integrated tools:** Why work with multiple low-code tools when AES integrates with several? From designing tables to building catalogs and creating custom flows, AES has it all.

App Engine Studio benefits

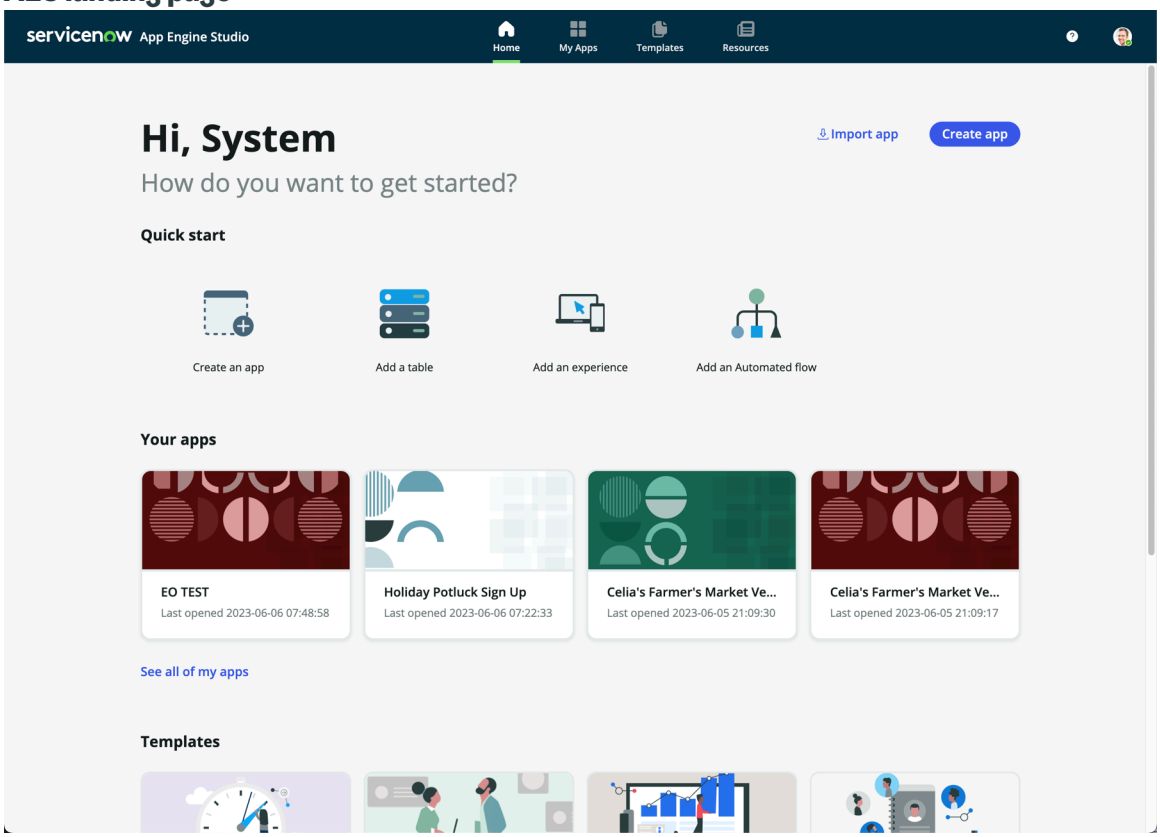
App Engine Studio provides the following benefits.

Benefit	Feature	Role
Submit ideas for an app, which admins then approve.	Submit your idea for app development	Citizen developer
Use templates for standard application types, such as procurement portal or time off request, as a starting point to create and customize new apps quickly.	Create your app using an application template	Citizen developer
Access multiple development tools in one interface, such as Table Builder and Workspace Builder.	Integrated development tools for AES	Citizen developer
Scale development while maintaining process to review and test applications before deployment.	Managing app development using the App Engine Management Center	Administrator

Accelerate your low-code development with AES

Using App Engine Studio's low-code development environment, citizen developers can quickly build high-quality apps. Citizen developers are individuals in your organization who aren’t experienced programmers, but who are subject matter experts with ideas for creating apps. With an approved idea in hand, a citizen developer chooses an app template, fills in a few parameters, and AES builds the app. The resulting application includes data, experiences, logic and automation, and security, all of which can be customized to your organization's needs.

AES landing page



Apps created from a template can be easily modified to meet the requirements defined by the citizen developer, or new apps can be created from scratch.

You can perform the following tasks to customize apps created using AES.

Task	Description
Create your data foundation easily	App Engine Studio templates contain default data tables that include everything you must deliver base system app functionality. Using the Table Builder app included with AES, you can edit or add new tables, add and delete columns, change table types, and more. You can also import existing data from spreadsheets, making it easy to automate existing manual spreadsheet-driven workflows.
Deliver great experiences	Experiences are role-based user interfaces that end users access to interact with your app. AES includes several distinct experience types, including configurable workspaces, portals, catalog items, and mobile experiences. You can add an experience by following a few simple steps. For example, you can quickly create a workspace by adding a name and description, a URL, and the roles that can access the workspace. Then you can tailor the workspace experience to the needs of your organization using the Workspace Builder tool within AES. More advanced configurations can be edited in UI Builder.
Automate your apps with workflow templates	AES templates include a set of powerful digital workflows that you can use to simplify the user experience, increase productivity, and strengthen your business processes.

Task	Description
	Simply choose from a list of common automation activities, and let the wizard quickly guide you through configuring the workflow. To adapt template-generated workflows to your needs, launch the Workflow Studio app to edit workflows.
Keep your apps secure	You can maintain control over who uses your apps by adding roles in AES. The app templates include predefined roles, giving you a head-start on security. You can add or delete existing roles, or create roles to meet your security needs.

Scale development while maintaining oversight

App Engine Studio lets you scale development across your organization while establishing and maintaining guardrails. You can control access to AES. You can also review apps created by your citizen developers before they're moved through your pipeline and published to your production environment. This review process gives you confidence to ensure the quality of the apps you create, release, and deploy.

What to explore next

To learn more about configuring and using App Engine Studio, see:

- Configuring App Engine Studio and related apps
- Building apps in App Engine Studio
- App Engine Studio reference

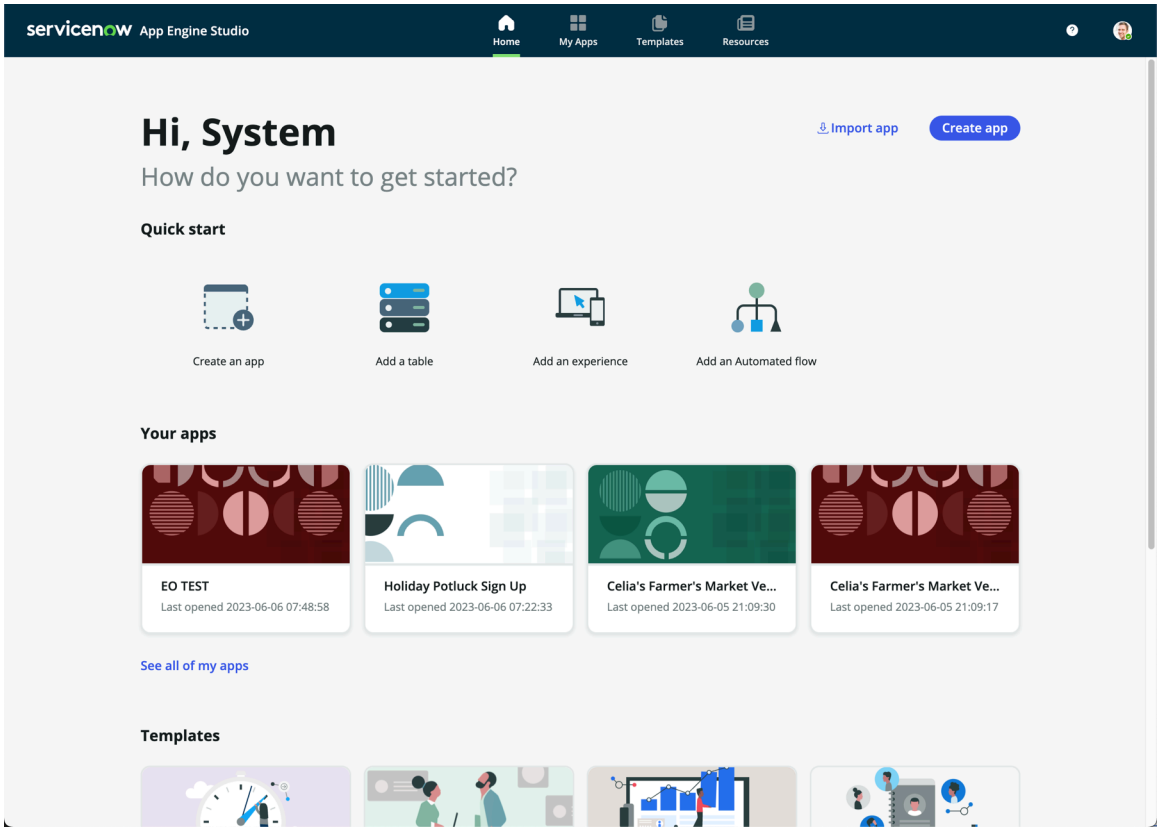
AES user interface

Learn about the App Engine Studio (AES) user interface.

AES is a web user interface that lets you build custom applications for your organization. You can build applications using templates which can be customized to fit your organizations needs.

App Engine Studio home

The App Engine Studio home page provides easy access to creating apps, viewing your app, quick start actions, and templates.



Use the top navigation bar to view your apps, access app templates, and find help resources. You can also access the help center by selecting the in the top right.

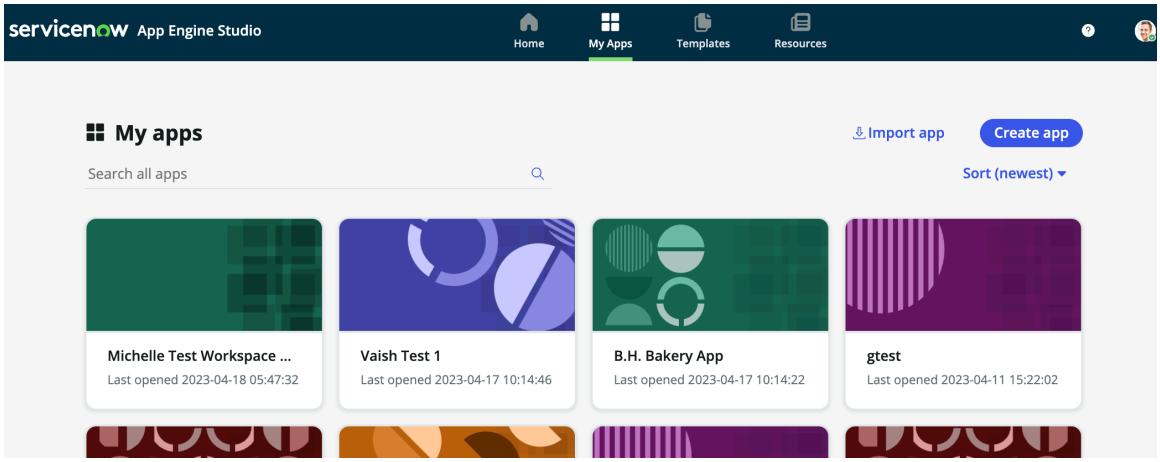
Users can access the following in AES:

- Create app — Create an application from scratch.
- Quick start — Links to add objects to apps, browse app templates, and learn more about the App Engine Studio tools.
- My recent apps — A list of applications you have accessed recently using App Engine Studio.
- Templates — A list of templates you can use to create an application with preconfigured data, experience, logic and automation, and security.

> **Note:**
> If you're in the App Engine Studio User Limited group, you won't see any templates on this tab.

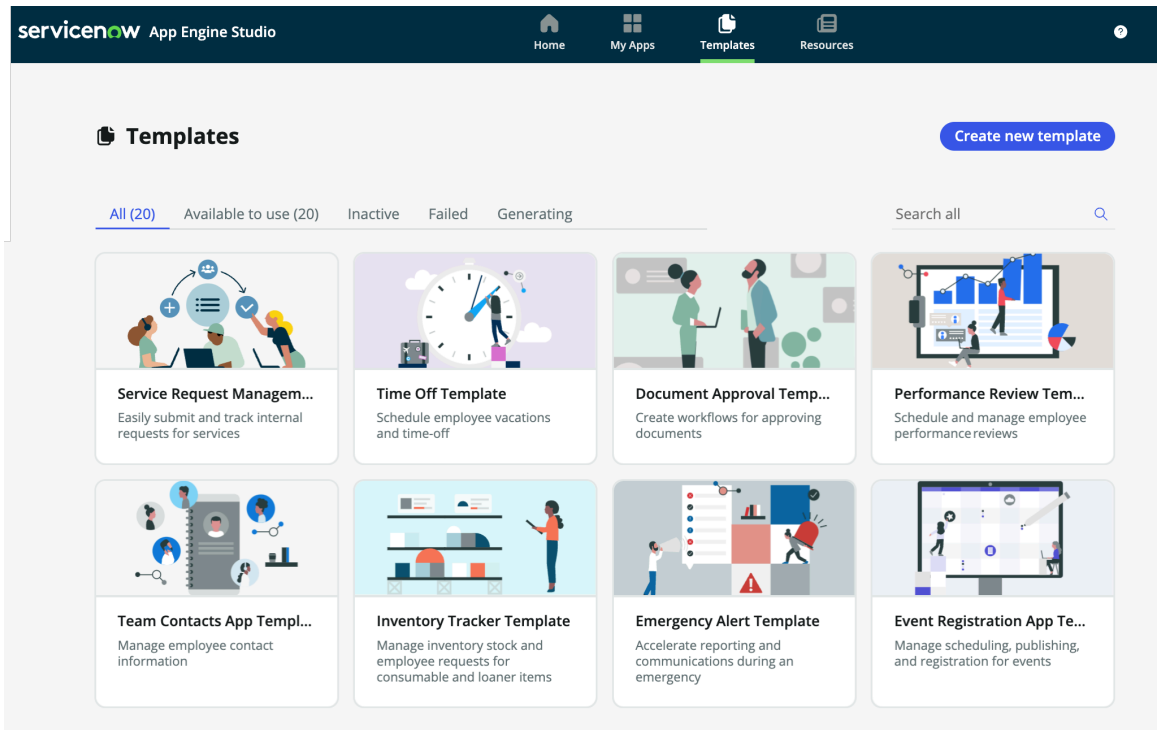
My Apps tab

View and search the applications you created.



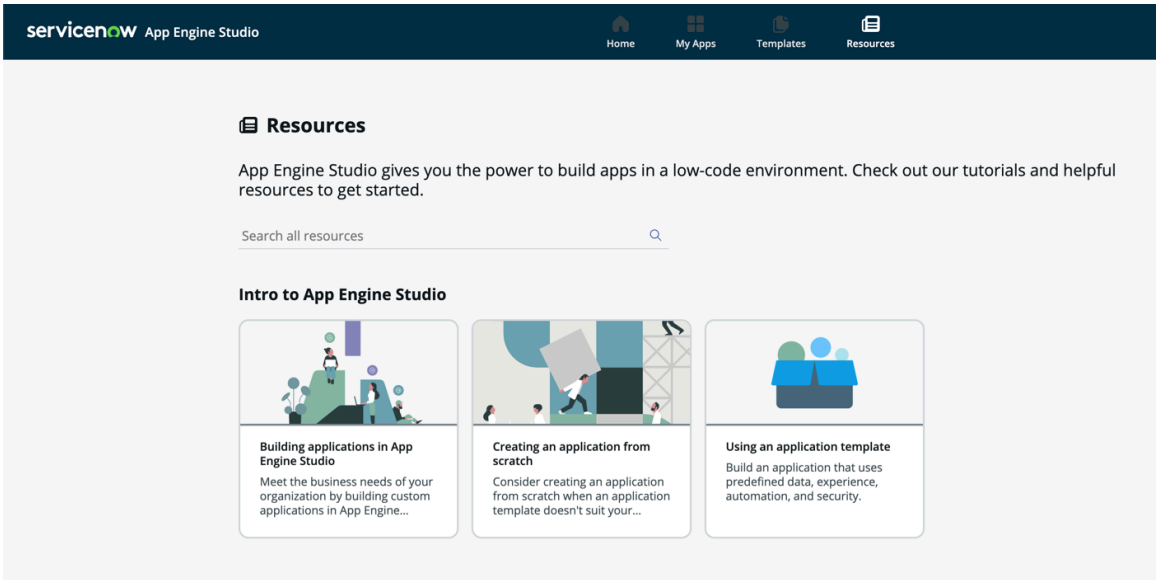
Templates tab

Provides a list of available app templates.



Resources tab

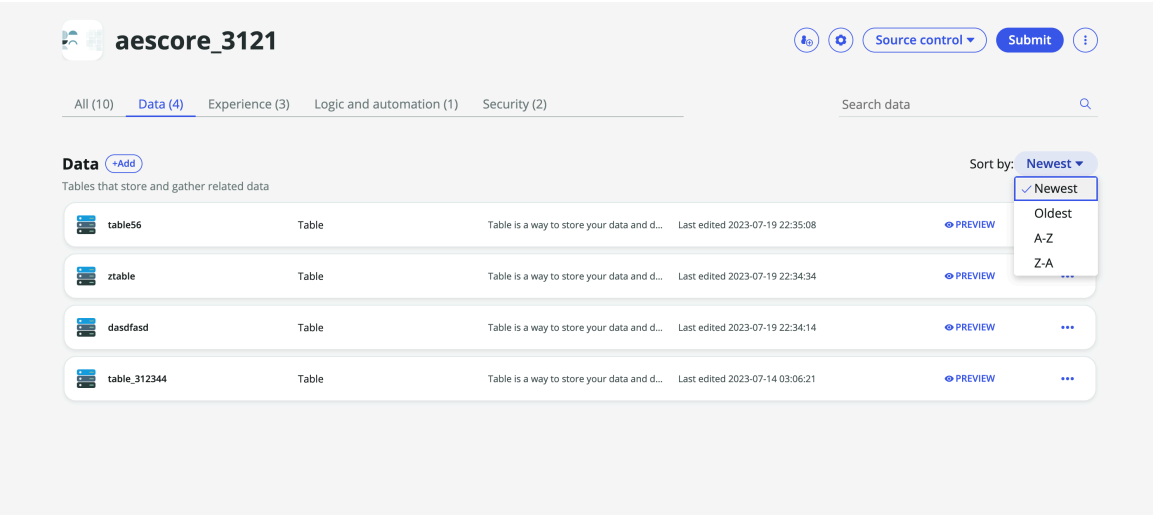
View tutorials and helpful resources to get started in AES.



App Home

Add data, experience, logic and automation, and security to your app from the app home.

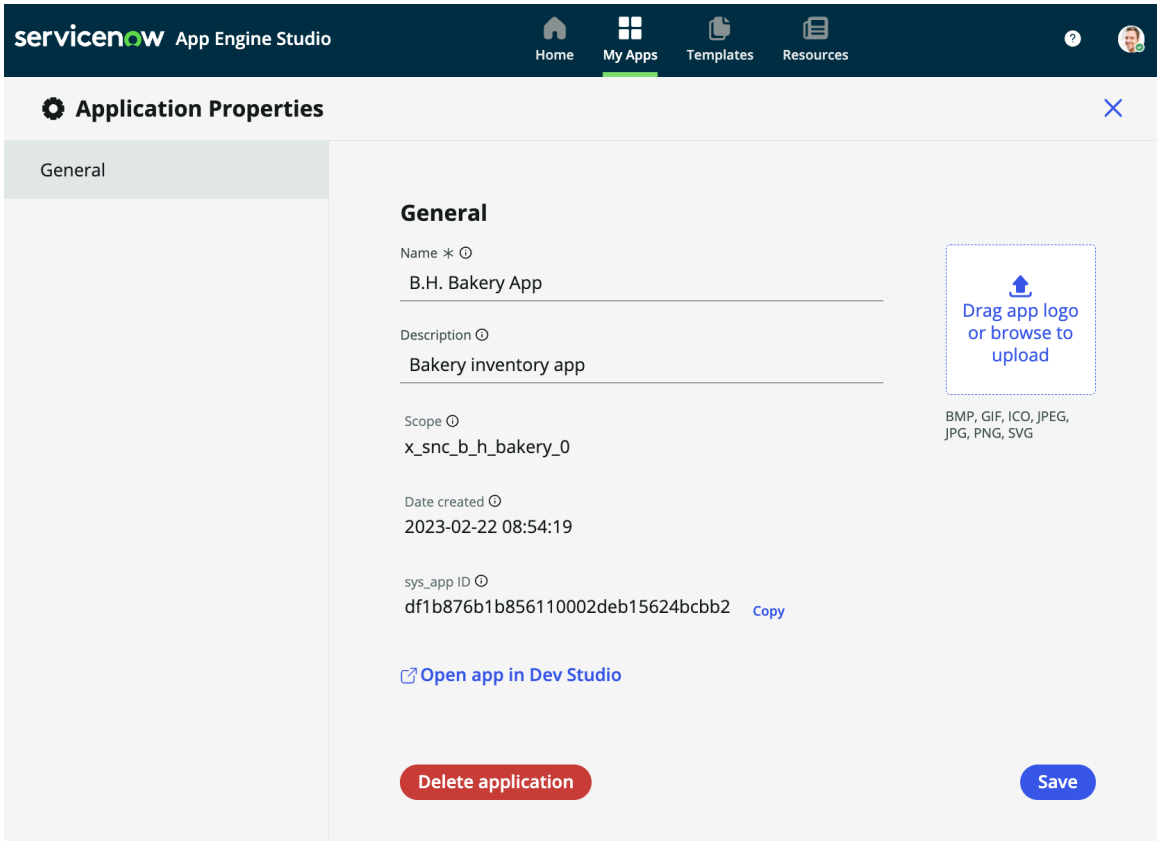
Note:
When viewing an app home page in AES, if you tab to a section (such as tables or experiences) with more than three items, the **Sort by** option enables you to display them alphabetically or temporally, by **Newest** or **Oldest**.



Some objects require access depending on your role. You can request access by selecting **Contact your system administrator** to become a delegated developer. For more information on the delegated developer role, see Delegate developers using AES.

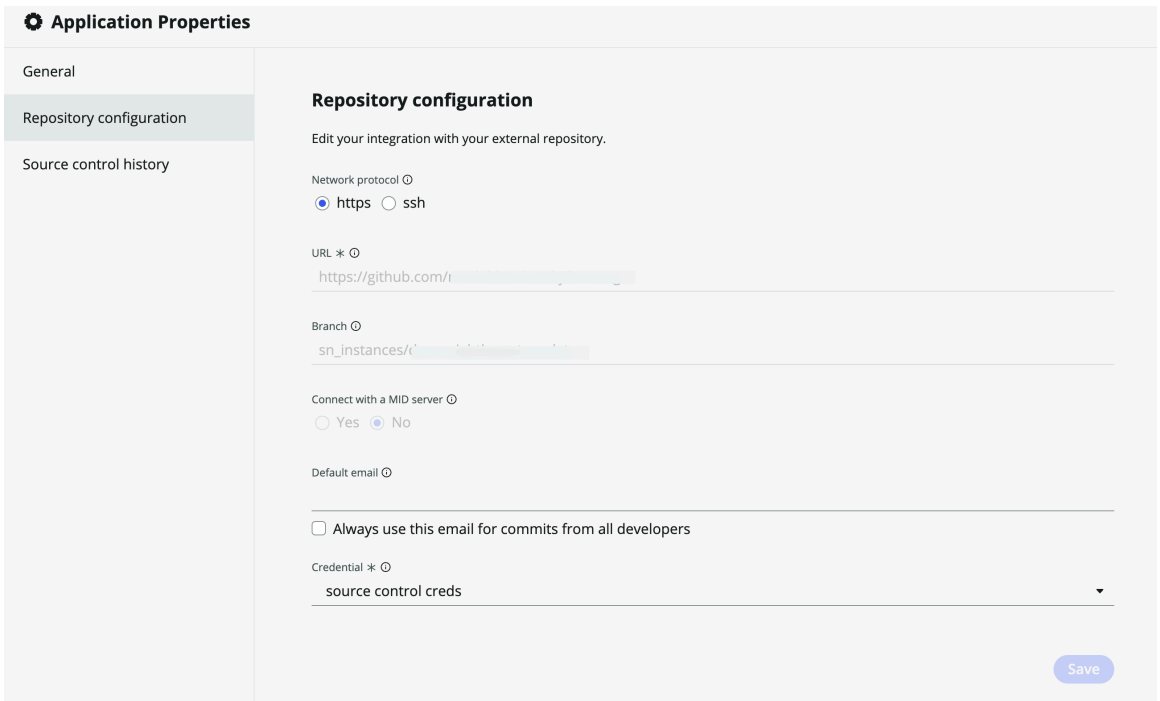
Application properties

View general settings of your application such as the name, description, and logo. You can also open your application in ServiceNow Studio or delete your application.



Repository configuration

View and edit the source control repository settings for your application.



Use App Engine instead of customization

App Engine development tools, such as App Engine Studio, offer an excellent alternative to customizing existing applications on the ServiceNow AI Platform.

When your company needs to add new functionality to the ServiceNow AI Platform, you can customize existing applications, such as IT Service Management (ITSM), or create a new application using App Engine developer products, such as App Engine Studio, Creator Studio, or ServiceNow Studio. A simple guideline for which path to choose is:

- If the customization extends the intended purpose of the application, it works better to customize. For example, you can add IT functionality to ITSM.
- If the customization doesn't extend the intended purpose of the application, it works better to create a new application using App Engine developer products. For example, do not repurpose the ITSM workflow to add a travel request workflow.

Examples of when to use App Engine

ServiceNow products work best when they're used as they were intended. If you find yourself heavily customizing an application to repurpose it, a better plan is to create a new application using App Engine developer products.

The following scenarios demonstrate where creating a new application works better than heavily customizing an existing ServiceNow application:

- Your company has a business process that augments existing product functionality but doesn't follow the same workflow exactly.
- You have a novel use case for an app that doesn't align with any product workflow.
- You have a use case that could be built by heavily customizing an out-of-the-box application, but it doesn't align with what the existing application was intended to do.

Let's dive deeper into the last use case.

Issues with repurposing existing products

ServiceNow applications come with roles, processes, and flows that are specially tailored to their use case. For example, ITSM apps help with IT users, IT issues, IT reports, and IT cases.

You might have an idea for an app that's similar to but doesn't exactly align with ITSM. Because ITSM gives you a starting point, you might be tempted to customize ITSM to add the new functionality. For example, ITSM tracks IT issues, and a travel app you want to create might track travel requests. While the workflows sound similar, in fact, they use very different data, different user interfaces, and the details of each workflow vary greatly. Rather than heavily customize ITSM to repurpose it, a better plan is to use App Engine developer products for the following reasons:

- Combining two workflows creates conflicts.
- Customizing applications has implications.

Combining two workflows creates conflicts

In the ITSM example, the repurposing of ITSM to include a travel workflow uses different data, different tables, different roles, and different workflows than ITSM. As ITSM, ITSM customizations, and the travel workflow grow over time:

- Their features will continue to diverge.
- Adding new functionality or fixing problems in one workflow might adversely impact the other.
- The performance of ITSM may suffer.
- The code base will grow and the two purposes of ITSM will make troubleshooting more difficult.
- Quality engineers will require two different testing frameworks.

All these issues may cause unnecessary complications, poorer performance, upgrade delays, and software problems.

Customizing applications has implications

The ServiceNow AI Platform is built to embrace customization and configuration. The ServiceNow AI Platform is flexible enough to fit your company's business needs. How you customize ServiceNow applications, however, can have significant impacts on ServiceNow support, upgrading to future ServiceNow AI Platform versions, and the functionality of the platform.

Let's start by differentiating customization and configuration:

- Customization is any change made to the code that is part of the baseline installation of a ServiceNow instance. You use code to customize applications.
- Configuration is any change you make to the behavior of a product that does not touch the code in the baseline installation of a ServiceNow instance. You can use system properties, ServiceNow products, or code to configure an application.

The following are some of the implications that result from customizing applications:

- If you add code to an application, you own it whether or not it modifies the code in the baseline installation on a ServiceNow instance.
- The platform marks all customizations and skips them when you update to a new version of the platform. That means you are responsible for manually updating the customizations. This can have a significant impact on the time and resources required to update to new platform versions.
- The ServiceNow AI Platform uses a framework that supports applications in how they process tasks, how forms are rendered in multiple browsers, and the overall user experience. Introducing customizations can have unintended consequences on this framework.
- You own the burden of testing custom code and determining if it impacts platform functionality.
- ServiceNow Customer Support cannot troubleshoot custom code or issues caused by custom code.

Customization is one of the key features of the ServiceNow AI Platform. However, over-customizing an application to repurpose it is likely to generate technical debt, lengthen your upgrade cycle, and complicate future platform upgrades because the custom code may not easily migrate to new platform versions.

Conclusion

Customization and configuration are hallmarks of the ServiceNow AI Platform that enable your company to customize workflows to fit its specific needs. Proceed with these tasks in the following order:

1. Configure ServiceNow applications as much as you can before customizing them.
2. Customize an application only when it extends the intent of the application.
3. Use App Engine developer products, such as App Engine Studio, Creator Studio, and ServiceNow Studio, to create new applications rather than customizing an application to create functionality that doesn't align with its original purpose.

Customization vs. configuration with App Engine Studio

There are important differences between customizing and configuring ServiceNow applications. The ServiceNow platform is built to embrace customization and configuration, but how you do so can have significant impacts on ServiceNow support, upgrading to future ServiceNow platform versions, and the functionality of the ServiceNow platform.

The general rules around customization are:

- Customize an application only if it extends the original intent of the application. For example, add IT functionality to ITSM but don't add a travel workflow. Instead of over-customizing an application, create a new application using App Engine products, such as App Engine Studio, Creator Studio, or ServiceNow Studio.
- Configure as much as you can before customizing an application.
- If you add code or make other modifications to out-of-the-box functionality, you own them.

What is configuration

Configuration is the process of using ServiceNow built-in tools and features to modify an application's behavior without making changes to flows or the code that is part of the baseline installation on a ServiceNow instance.

Configuration can take the form of using ServiceNow built-in tools to add tables and more, setting instance-wide parameters, as well as using code to extend an application's functionality to meet business needs as long as the code does not modify the baseline code installation. The entire platform is designed for you to add configuration code.

If you add code, such as workflow scripts, you own it, even if it doesn't alter the baseline code installation. That includes owning the impact it has on the entire ServiceNow platform. Issues that arise from added code are beyond the scope of ServiceNow support to debug.

Reverting a configuration should not require any change to the baseline code.

Configuration examples include:

- Forms: Configure tables, fields, data types, default values, and field dependencies to configure the data you capture and display.
- UI elements: Modify layouts, add related lists, add buttons, and change field names.
- Service Catalog: Configure portals where your customers can request catalog items such as service and product offerings.
- ACLs: Restrict unauthorized users from accessing forms and data.
- System property values: Modify the application's experience for all users.

What is customization

Customization is any change made to the flows or the code that is part of the baseline installation on a ServiceNow instance. You use ServiceNow products or code to customize applications.

If you add code, you own it, even if it doesn't alter the baseline installation. That includes owning the impact it has on the entire ServiceNow platform.

Customization examples include:

- Scripting: Customize ServiceNow through scripting using JavaScript. This includes creating client scripts, server-side scripts, and business rules with intricate logic that modify the baseline code.
- Custom tables: Develop custom tables to accommodate specialized data that doesn't fit within standard tables.
- Integration: Customize integration with external systems, such as APIs and web services, for seamless data exchange.

- Widgets and portals: Create custom widgets and portals to provide unique features and user experiences.
- Workflows: Create and modify workflows using Workflow Studio. Create and manage playbooks, flows, actions, decision tables, and integrations from one design environment to automate tasks. Upgrading to a new version of a flow requires reapplying your customizations.

Tools for customization and configuration

ServiceNow offers many tools and features, such as business rules, to modify the out-of-the-box behavior of ServiceNow applications. Whether they customize or configure an application depends on how they're used. Using these tools to modify the installed code base constitutes customization. Using these tools to add code that does not modify the flows, or the installed code base constitutes configuration. In both cases, you own the code you add as well as the impact it has on the ServiceNow platform.

ServiceNow tools include:

- UI Policies: Dynamically modify the visibility of fields and attributes on a form according to user inputs.
- Business rules: Automatically trigger actions based on specified conditions.
- UI Actions: Extend and customize forms and lists by adding buttons, context menu items, or other UI elements that perform specific actions when clicked.
- Client-side scripts: Scripts that execute within the user's browser when certain actions occur on a form or a UI page.
- Server-side scripts: Scripts that execute on the ServiceNow server or database, for example, to update record fields when a database query runs.

What is personalization

Personalization is when users use out-of-the-box application tools to modify an application's look and feel only for themselves. Admins can change the look and feel for all users and that is considered configuration. Personalization examples include a user choosing to use dark mode or choosing which table columns to display.

Personalization does not change the baseline code installation on a ServiceNow instance. So, personalization does not impact customer support or interfere with upgrades to new ServiceNow versions.

Ramifications of customizing ServiceNow products

The ServiceNow platform is extremely flexible and built to embrace customization and configuration to fulfill a wide range of business requirements. How you customize ServiceNow applications, however, can have significant impacts on ServiceNow support, upgrading to future ServiceNow platform versions, and the functionality of the platform. Instead of customizing ServiceNow applications, consider using App Engine development products, such as Creator Studio and ServiceNow Studio to create new applications.

The ServiceNow platform uses a framework that supports applications in how they process tasks, how forms are rendered in multiple browsers, and the overall user experience. ServiceNow relies on the framework's integrity to develop and provide support in a consistent manner. Customizations can impair this framework, change platform functionality, and impair workflows and upgradeability.

Customizations trigger the platform to create sys_update_xml records, which are stored in the Customer Update table. The platform marks all customizations and skips the customized records

when you update to a new version of the ServiceNow platform. That means you are responsible for manually updating the customizations. This can have a significant impact on the time and resources required to update to new platform versions.

Note:
The Customer Update table also contains modifications or additions to configuration metadata, for example, creating a new catalog item or a new workflow.

For more information, see the Customer Updates table. Note that the complexity of maintaining customizations dramatically increases as the number of customizations increases.

Customizing the installed code base can be costly, generate technical debt, lengthen your upgrade cycle, and complicate future platform upgrades because the custom code may not easily migrate to new platform versions. Custom code can change the standard functionality of the ServiceNow platform in unintended ways. Evaluate demands for customization carefully and only resort to customization where there is clear business value and no alternative. Wherever possible, avoid customization by using configuration instead.

If you customize a product:

- You are responsible for maintaining the customization going forward.
- Customer Service and Support will not support problems caused by custom code. If it is the cause of the problems, the support team will likely advise you to revert to the out-of-the-box code.

What is the Customer Service and Support stance on supporting customization

The ServiceNow Customer Service and Support stance on customization is if you add code, you own it and its consequences. Why? Customer Support is not privy to your custom business logic, doesn't know what the expected behavior should be, is unable to reproduce the issue on an out-of-the-box instance, and customer support engineers are not certified implementation specialists, so they are not certified to review customized code logic.

Alternatives to customization

If you have requirements and ideas for enhancements, instead of customizing the installed code base, you can:

- Use configuration instead of customization.
- Submit an Enhancement Request to the ServiceNow development team. Each request is evaluated and, if approved, will be incorporated into a future release.
- Create an app using App Engine developer products to handle desired functionality.

When to use App Engine developer products instead of customizing

When your company needs to add new functionality to the ServiceNow platform, you can customize existing applications, such as ITSM, or create a new application using App Engine developer products, such as Creator Studio or ServiceNow Studio. A simple guideline for which path to choose is:

- If the customization extends the intended purpose of the application, it works better to customize. For example, you can add IT functionality to ITSM.
- If the customization doesn't extend the intended purpose of the application, it works better to create a new application using App Engine developer products. For example, do not repurpose the ITSM workflow to create a travel request workflow.

For example, ITSM is designed to handle IT issues. To customize it to handle travel requests goes beyond the original intention of ITSM. Because IT and travel requests have different workflows, it's better to create a travel request app using App Engine developer tools, such as Creator Studio and ServiceNow Studio, instead of customizing ITSM.

For more information, see Build apps using App Engine Studio.

Examples of when to use App Engine developer products

ServiceNow products work best when they're used as they were intended. If you find yourself heavily customizing an application to repurpose it, a better plan is to create a new application using App Engine developer products.

The following scenarios demonstrate where creating a new application works better than heavily customizing an existing ServiceNow application:

- You have a novel use case for an app that doesn't align with any product workflow.
- You have a use case that could be built by heavily customizing an out-of-the-box application, but it doesn't align with what the existing application was intended to do.
- Your company has a user group or business process that should be separate from the OOTB product workflow.

Guidelines for customizing ServiceNow products

If you must make a customization, consider the following suggestions:

- Maximize configuration options first.
- Avoid copying objects. Instead, update objects in place wherever possible, except for Service Portal widgets and other items designated to be reused.
- Default to "add before edit." This means that you should, for example, add fields to forms rather than change the type of an existing field. Avoid using the same names as out-of-the-box objects, methods, or classes when adding them.
- Minimize the number of fields you add to a form. The more fields you have on a form, the longer it may take to load.
- Export original records as backups before customizing them. Keep track of their out-of-the-box sys_id's in case you must restore them in the future.
- Use scoped applications as your default for any new custom development.
- Document all customizations. Add comments explaining why you customized (including business justification). Review all comments before upgrading to determine if you can revert to out-of-the-box code.
- Create tests for all customizations. Write Automated Test Framework (ATF) tests for all customizations where possible.
- Use HealthScan regularly to identify unnecessary customizations.
- Customizations should be made to baseline objects where necessary so that conflict resolution and decision-making can be appropriately recorded in the updates. Hidden customizations may cause administrators to overlook updates in future assessments in case reverts or merges are necessary.
- Test your customization for all use cases. Include performance testing and the introduction of unintended consequences.
- Administrators are responsible for verifying that their customizations work after a ServiceNow platform upgrade, and for keeping track of what customizations are made.

Handling customizations when you upgrade

Customizations trigger the platform to create sys_update_xml records, which are stored in the Customer Updates table. These records are not updated during platform version upgrades. ServiceNow marks them as skipped records in the ServiceNow Upgrade Monitor. To make sure they're successfully ported to the upgraded instance, you must manually process the skipped changes. For more information, see the Customer Updates table.

Assuming you've documented all your customizations—including the business justification—take your documented inventory and compare it with the skipped records identified in the Upgrade Monitor. After filtering out low-risk changes that have resulted in skipped records (for example, field labels or form layouts), you'll need to decide whether to:

- Retain each customization
- Revert to out-of-the-box
- Merge your customization with the base system to resolve conflicts

Integrated development tools for AES

App Engine Studio (AES) provides instant access to several tools available in the ServiceNow AI Platform development suite.

As you create apps using App Engine Studio, you can extend their capabilities with these powerful development tools.

App Engine Studio development tools and features

Tool	Description	Reference
Access Control	Work with access control and roles for the application.	See Add application security.
Catalog Builder	Add record producer catalog items in a guided tool for building a Service Catalog.	See Add a standard catalog item.
Decision tables in Workflow Studio	Add decision rules to decouple decision logic from code using decision tables.	See Add a decision.
Email notifications	Create and edit email notifications that are automatically sent when a record is created or updated. Email notifications are also used as a reference in flows or can be triggered by an event.	See Add an email notification.
ERP Data Hub	Build an ERP (Enterprise Resource Planning) data model and extract data from your ERP system.	See ERP Data Hub.

App Engine Studio development tools and features (continued)

Tool	Description	Reference
ERP Customization Mining (ERP-CM)	Identify all of the customizations in your ERP system.	See ERP Customization Mining (ERP-CM).
Flows in Workflow Studio	Add flows to automate application processes.	See Add a flow from scratch.
Integration Hub - Import	Import and map data into existing tables.	See Create a data integration.
Mobile App Builder	Add mobile experiences.	See Add a mobile experience.
Process Automation Designer	Edit cross-functional processes and consolidate them into automated task-oriented views.	See Edit a process.
Table Builder	Work with data models in a tabular format. Table Builder is exclusive to App Engine Studio and necessary for creating apps.	See Create a blank table.
UI Builder	Edit workspace and portal experiences using a highly configurable UI development tool.	See Add a workspace and Add a portal.
Workspace Builder	Quickly build custom workspaces to give users a service desk-like experience to manage tickets.	See the following topics: • Building workspaces in AES • Configure a workspace in Workspace Builder • Configure workspace settings in Workspace Builder • Configure a workspace home page in Workspace Builder • Configure a record page for a workspace in Workspace Builder

App Engine Studio development tools and features (continued)

Tool	Description	Reference
		• Configure lists for a workspace in Workspace Builder • Configure analytics for a workspace in Workspace Builder

Get help with App Engine Studio

To get help with App Engine Studio, your ServiceNow instance, plugins, permissions, and more, watch a short video to contact the ServiceNow admin who works in your company.

Get help now from your Now Support administrator

Watch the following video to find out how to get help quickly.

https://player.vimeo.com/video/1056112088?h=1a5cd3b245&badge=0&autopause=0&player_id=0&app_id=58479

Helpful resources

Some ServiceNow resources that can provide you with helpful information are:

Video

Watch Build apps fast with App Engine Studio.

ServiceNow Community

App Engine Studio Community

App Engine and App Engine Studio

ServiceNow® App Engine Studio (AES) is a development tool for creators of varying skill levels to build applications that meet the immediate needs of your organization. The App Engine Enterprise version includes App Engine Studio (AES) for building low-code apps.

Streamline app development with App Engine

App Engine Starter and Enterprise versions

App Engine Starter is included in all products (that is, it's free). To access additional features outside of App Engine Starter, you need a paid App Engine Enterprise license.

The App Engine Enterprise version enables users to act as a Fulfiller for unlimited custom apps and unlimited custom tables. Customer entitlements may vary depending on the type of App Engine subscription purchased. Contact your account team or your company's ServiceNow admin for entitlement confirmation.

For a full list of App Engine products, see the ServiceNow product site.

Additional information on App Engine

Learn more about App Engine	Additional ServiceNow resources
ServiceNow provides several additional resources on App Engine.	 App Engine Enterprise purchase overview and version comparison
	 App Engine Enterprise data sheet (PDF)
	 Entitlements for subscription products
	 Custom table guide (PDF)

Learning plan for getting started with App Engine Studio

ServiceNow provides citizen developers with several quick training modules for getting started building low-code apps in App Engine Studio (AES).

The ServiceNow Developers site provides numerous resources for developing apps in AES.

Use the following learning modules to quickly explore ways to build apps.

ServiceNow Developers site learning modules

Course	Description
Build My First App Engine Studio Application	Create an application to track a collection using AES. The application will be secure, automated, and provide an engaging user experience.
Create an App and Data Model	Create an application from scratch with AES, including the data model used by the application. The data model is a foundation for learning how to add user experiences, logic and automation, and security to your application in other learning modules.

ServiceNow Developers site learning modules (continued)

Course	Description
Create Record Producer User Experiences	Create record producer catalog items in AES. Users can submit a request or create a task using Service Catalog items in Service Portal.
Create Workspace User Experiences	Create a workspace to report on and manage data in an application created with AES.
Create Mobile User Experiences	Create and test a mobile experience in AES that enable users to access applications from anywhere.
Automate Processes with Flows	Create flows in AES to automate business processes. Flows reduce human error, improve efficiency, and make apps work better.
Secure Apps and Data	Use AES to secure an application by creating roles, assigning the roles to users, and testing user access to application data.

Additional learning resources

Learn more about application development
You can learn about How citizen developers cook up innovation on the ServiceNow blog.
For more information about defining cross-scope access to an application resource and approving or denying requests, see Define cross-scope access to an application resource.

Configuring App Engine Studio and related apps

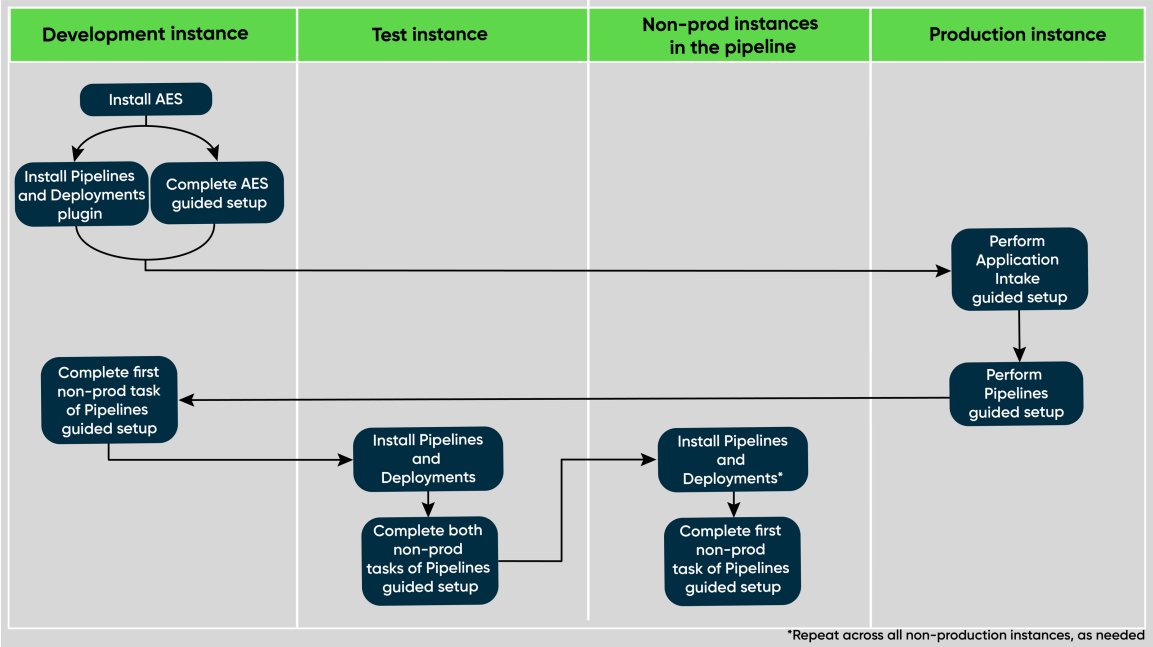
App Engine Studio (AES) guided setup walks you through the implementation process, one step at a time. As you finish each setup procedure, you're prompted to begin the next one.

When you have finished configuring AES, you have the option of configuring other products related to AES:

- App Engine Management Center (AEMC)
- Application Intake
- Pipelines and Deployments

Each of these products has its own guided setup. If you have already configured one or more of the products and you are prompted to configure it again, you can simply ignore that instruction. The following diagram illustrates the preferred method for implementing AES and associated applications:

AES implementation



AES implementation usually follows these steps.

1. On a development instance, install AES and the Pipelines and Deployments plugin. For more information, see Installing App Engine Studio.
2. Complete the AES guided setup. For more information, see Configure App Engine Studio.
3. On your production instance:
   a. Complete the Application Intake guided setup. For more information, see Configure Application Intake.
   b. Complete the Pipelines and Deployments guided setup. For more information, see Configure Pipelines and Deployments.
4. On a development instance, complete the first non-production task of the Pipelines and Deployments guided setup.
5. On your test instance, install Pipelines and Deployments.
6. Complete both non-production tasks of the Pipelines and Deployments guided setup on the test instance.
7. In all the other non-production instances in your pipeline:
   a. Install Pipelines and Deployments.
   b. Complete the first non-prod tasks of the Pipelines and Deployments guided setup.
8. Repeat step 7 across all non-production instances, as needed.
9. Install the AEMC plugin on each instance in your pipeline. For more information, see Configure the App Engine Management Center.

   AEMC allows admins to manage app development from intake through production on your production instance.

Note:
If you plan on cloning your production instance to one or more non-production instances, you should also install the AES product on your production instance prior to cloning. For more information, see System clone and Cloning instances with AES.

AES and domain separation

Domain separation is unsupported for App Engine Studio (AES). Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Support level: No support

- The domain field may exist on data tables but there is no business logic to manage the data.
- This level is not considered domain-separated.

For more information on support levels, see Application support for domain separation.

Related topics

Domain separation for service providers

Installing App Engine Studio

You can install the App Engine Studio (AES) application (com.snc.app-engine-studio) if you have the admin role. The application installs related ServiceNow® Store applications and plugins if they are not already installed.

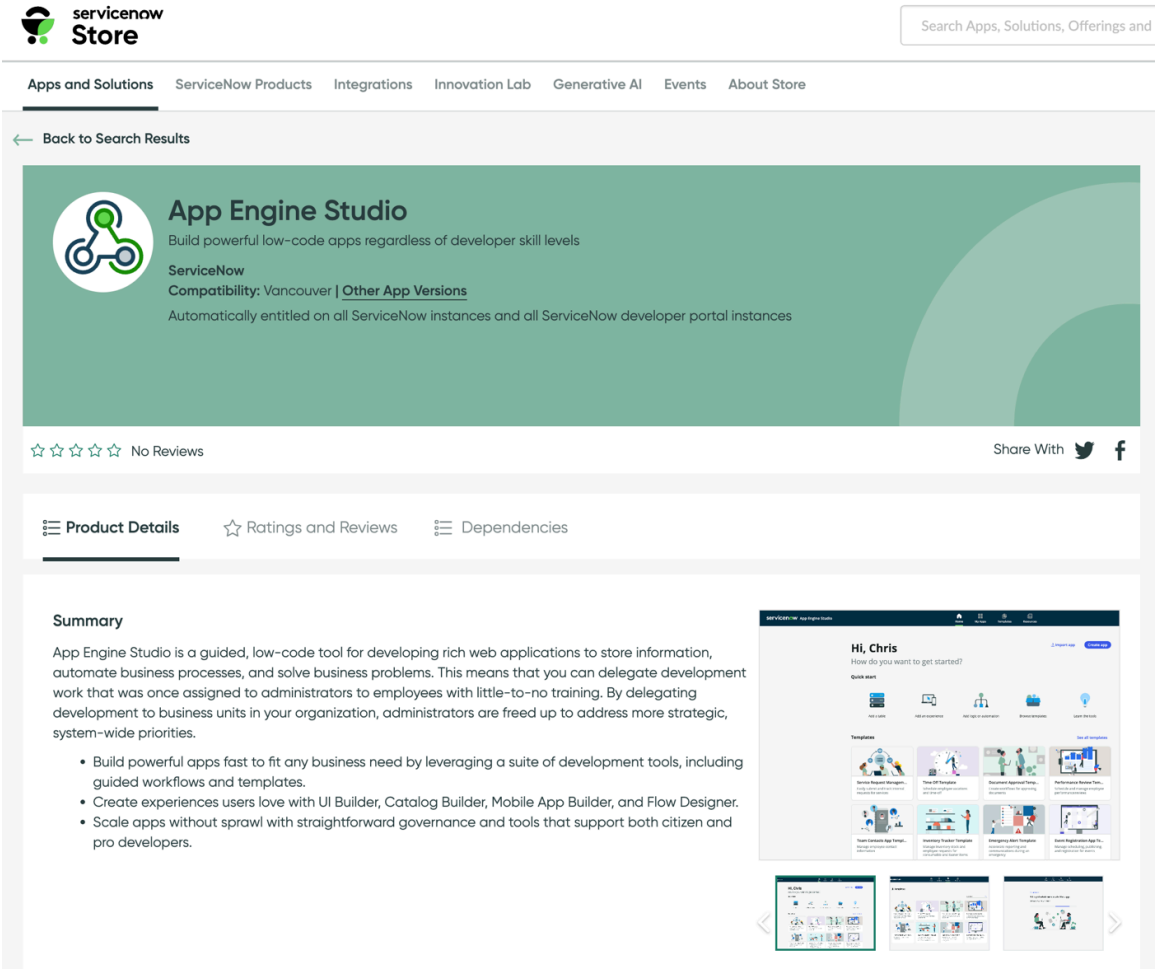
Trial version of AES

To try AES before purchasing a license, you can install it on a personal developer instance (PDI). For more information about PDIs, see Personal developer instance guide. To obtain a PDI, see the ServiceNow Developer Site.

Note:
An App Engine Studio subscription is required to install on a production or non-production instance. For more information, contact your ServiceNow representative.

Using the ServiceNow Store to install AES

The ServiceNow Store enables you to download core products and applications. A product contains one or more applications that are licensed as a group. For example, App Engine Studio is a product that contains numerous tools and applications within it, such as Table Builder and Mobile App Builder. If you install the basic AES application and not the full AES product, you won't get its full suite of features and tools.



Check that the application and all of its associated ServiceNow Store applications have valid ServiceNow entitlements. For more information, see Get entitlement for AES apps.

When you install AES, you also install the following related items:

- Roles and groups
- Tables
- Plugins

For more information, see Components installed with AES.

Installation workflow for AES

The following is an overview of the tasks that you must perform to install AES for the first time:

1. Develop your instance strategy for AES.
2. Get entitlement for AES to ensure you have a license.
3. Install AES (the full product, not the application) from the ServiceNow Store. App Engine Studio is a product that contains numerous tools and applications within it, such as Table Builder and Mobile App Builder. If you install the basic AES application and not the full AES product, you won't get its full suite of features and tools.
4. Install the AES integrations and plugins.

Alternatively, you can update existing AES installations by doing the following:

- Update a previously installed AES version
- Upgrade your AES instance to the next family release

Learn more about AES installation	Additional ServiceNow resources
ServiceNow provides several additional resources on installing and administering App Engine Studio.	App Engine Studio release notes ServiceNow Community site ServiceNow University Introduction to App Engine Studio for Citizen Developers course

AES instance strategy

You should install App Engine Studio (AES) on all ServiceNow instances where users will develop applications.

You will need to establish your company's instance strategy for AES. Some of the decisions you will need to make include:

- Whether you want to allow anyone within your company to have access to AES to start building apps.
- Whether you want to grant access to a select group of people.
- Whether you want to grant access on a case-by-case basis by setting up a form where individuals can complete information about the app that they are looking to build. IT then decides whether to give those individuals access to build that app.

A non-production instance that is similarly configured to your production instance may be the best candidate for your test environment. You can then more accurately find issues that may arise if the application is deployed to production.

Note:
If you plan on cloning your production instance to one or more non-production instances, you should also install the AES product on your production instance prior to cloning. For more information, see System clone and Cloning instances with AES.

After you have established your instance strategy, you must also establish and automate your approval or review process. AES is a product that runs on your non-production environment. For an organization with multiple non-production environments, you will need to decide which non-production environment AES will run on. You must also determine which pipeline to use for promoting apps from a particular non-production instance to your test instance, and then

finally to production where the app will be running live. For more information, see Pipelines and Deployments.

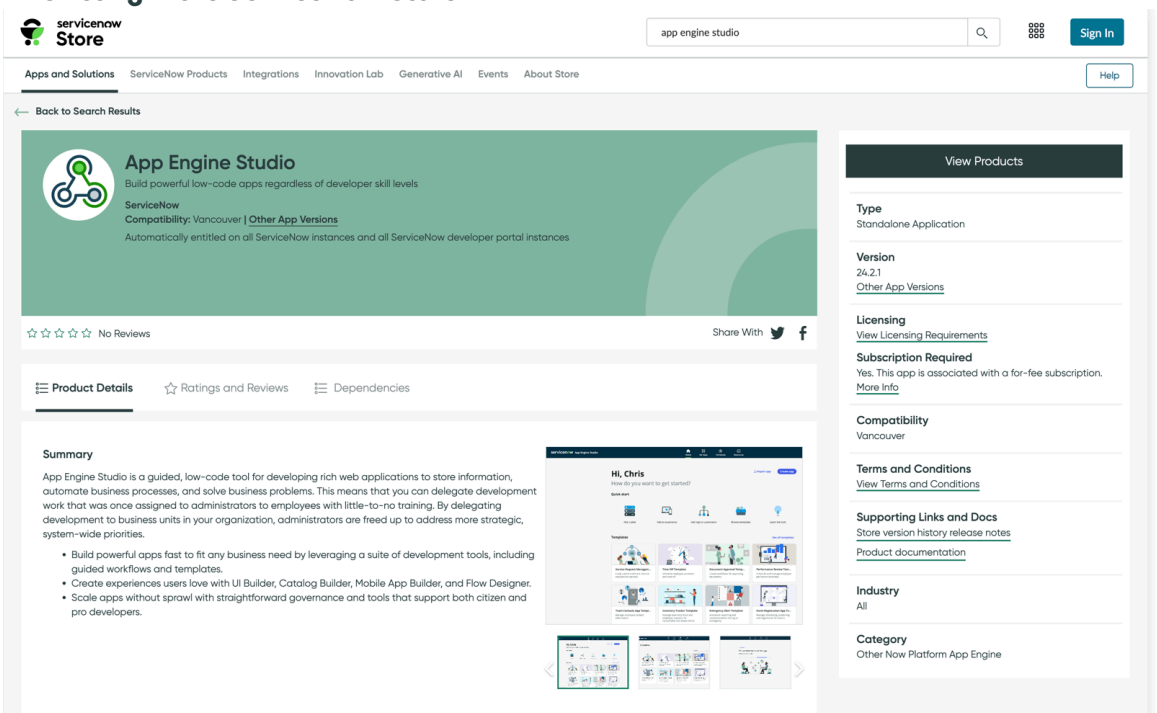
AES and the ServiceNow Store

App Engine Studio (AES) enables you to obtain new and updated features more rapidly. Before you can use AES, you must verify that you have entitlement to it, meaning that you have valid licenses to use it. Then, you can install AES and its plugins from the ServiceNow Store.

Request apps on the Store

Visit the ServiceNow Store website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

AES listing in the ServiceNow Store



Methods for installing AES

The process you use to obtain AES depends on whether you are installing AES for the first time, updating a version of AES that you previously installed from the ServiceNow Store, or upgrading from one family release to the next one.

Get entitlement for AES apps

Install App Engine Studio (AES) by verifying that the application or the product and its associated applications have valid ServiceNow entitlements. An entitlement means you have a license to install the product.

This video shows you how to perform the following procedure.

Get entitlement for AES apps

Before you begin

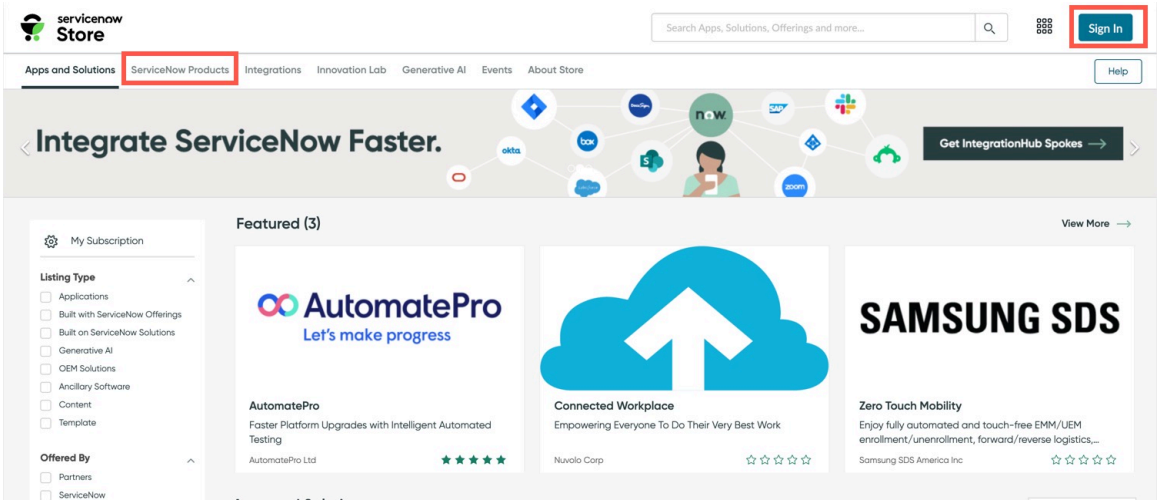
Note:
An App Engine Studio subscription is required to install on a production or non-production instance. For more information, contact your ServiceNow representative.

Individuals performing entitlement must have a Now Support account and have permission to request applications for the instances under consideration.

Role required: admin

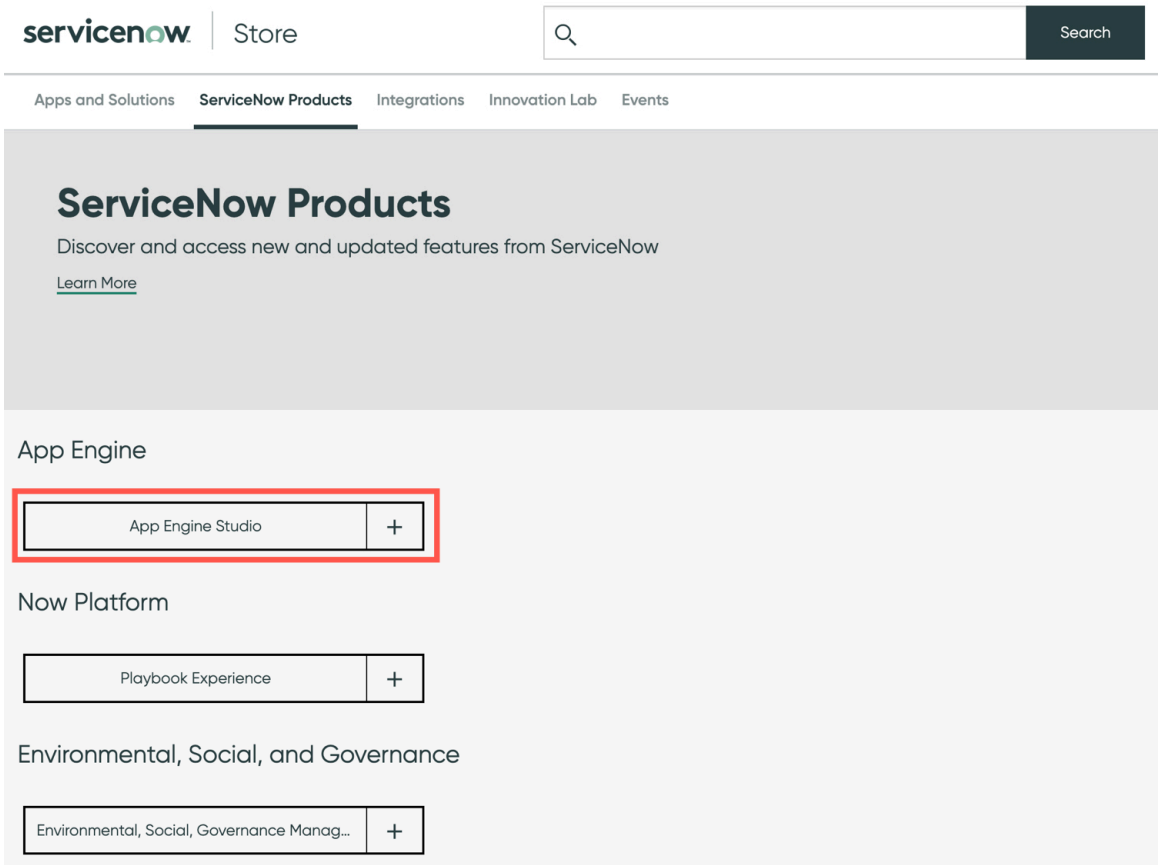
Procedure

1. Navigate to the ServiceNow Store.
2. Select **Sign In** and log in using your Now Support credentials.



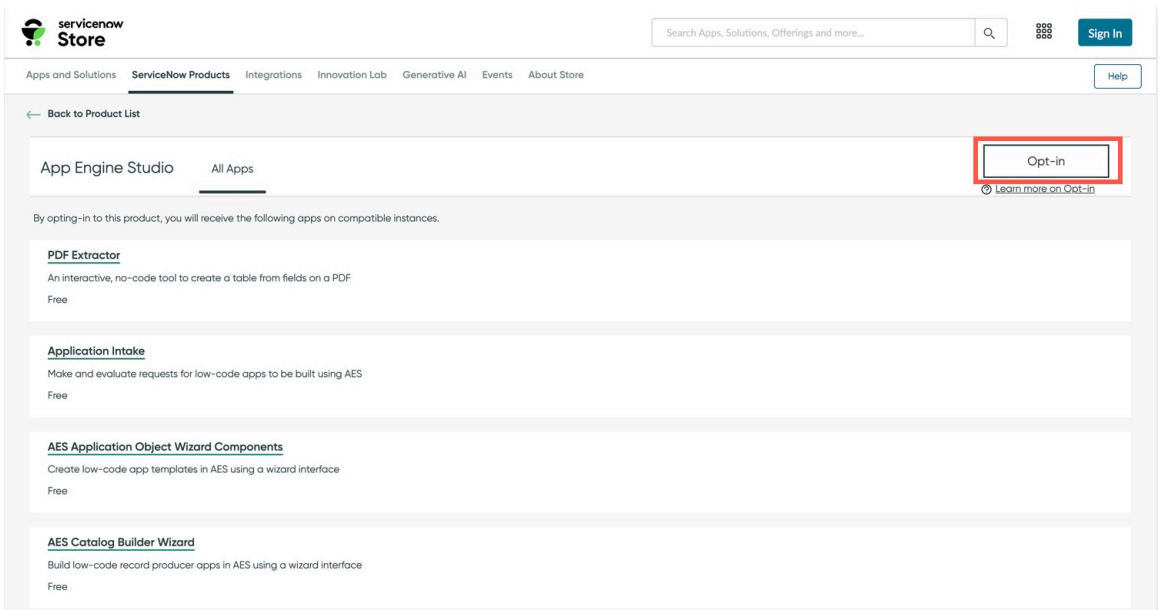
Integrations and other types of content appear on the **Certified Apps** tab.

3. Select the **ServiceNow Products** tab to view all available ServiceNow products.



4. Select **App Engine Studio**.

> **Note:**
> When you have obtained entitlement to the core product, the applications listed under it are also entitled.



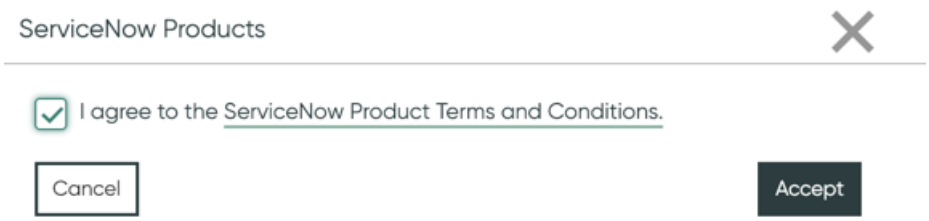
The page lists all the applications that are associated with the AES product you're entitling.

5. Verify that you have entitlement to the product and applications by selecting **Opt-in**.

Note:
The **Opt-in** button is not available unless you purchased an AES subscription.

6. When the ServiceNow Store prompts you to read and accept the ServiceNow terms and conditions, select the check box and select **Accept**.

   Opting in to entitlement is a one-time process for each product.



   A check mark appears next to the product name, and the **Manage Entitlement** button appears. The button indicates that you are subscribed to the product and its associated applications.

7. Change the instances affected by the entitled applications by selecting the **Manage Entitlement** button and then specifying one of the following options:
   - **Remove all existing entitlements**
   - **Entitle all instances**
   - **Entitle selected instances**

Result
Install and activate AES. Navigate to **All** > **My Company Applications** and select App Engine Studio.

Install AES from the ServiceNow Store for the first time

Install App Engine Studio (AES) from the ServiceNow Store for the first time. The installation involves several easy steps. Some steps are performed on the ServiceNow Store and some in your ServiceNow AI Platform instance.

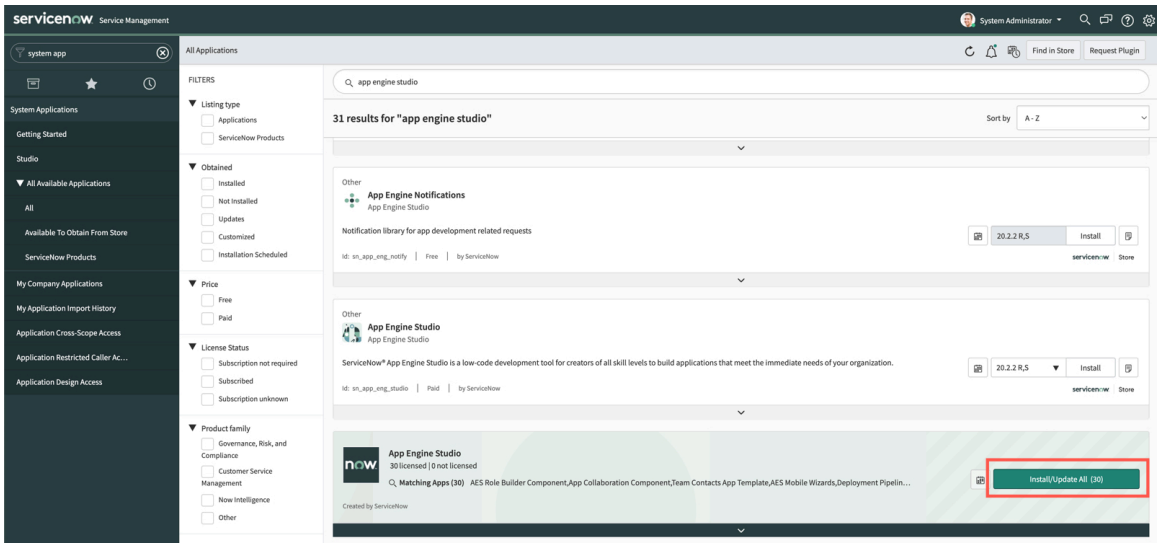
Before you begin
Role required: admin

About this task
The ServiceNow Store enables you to download core products and applications. A product contains one or more applications that are licensed as a group. For example, App Engine Studio is a product that contains numerous tools and applications within it, such as Table Builder and Mobile App Builder. If you install the basic AES application and not the full AES product, you won't get its full suite of features and tools.

For a list of all dependencies and tools installed with the AES product, see the listing in the ServiceNow Store.

Procedure
1. In the ServiceNow Store, ensure that you have entitlements (or licenses) to the product and its dependent applications.
2. Install AES from the ServiceNow Store.
3. Run the installation on your instance, and accept all the plugins.

Result



Install the AES product and integrations

Install the App Engine Studio (AES) product to ensure you get all the dependencies, integrations, and tools, such as templates and builders.

Before you begin

Store installation requires a Now Support account. You must also have permission to request applications for the instances you're supporting.

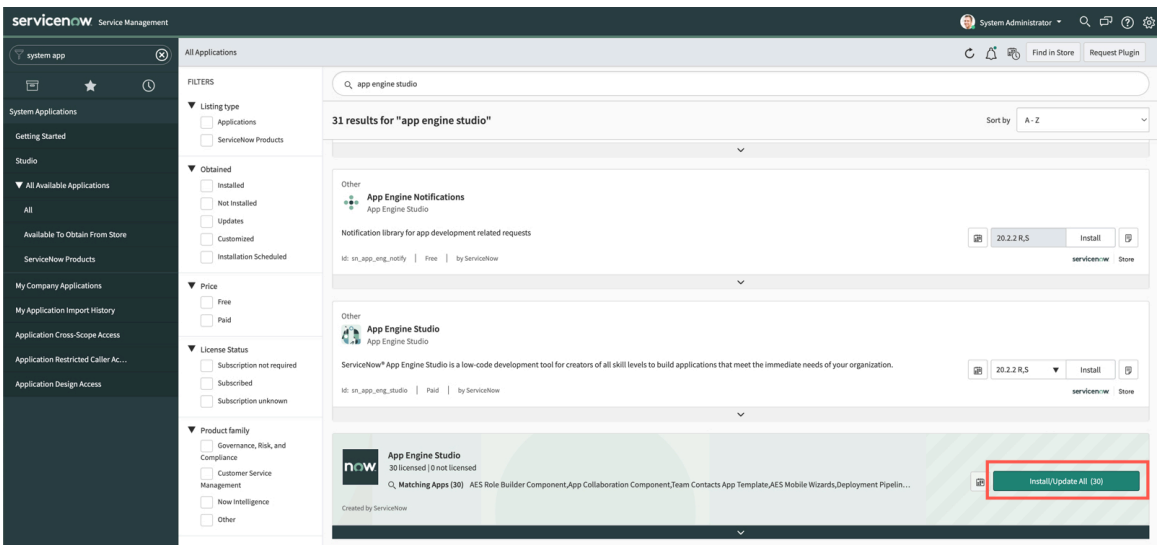
Role required: admin

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. Locate the App Engine Studio product (not the application), and select **Install**.

   Visit the ServiceNow Store website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

   Note:
   You must select the AES product, not the application.



3. On the Install dialog box, review the list of applications and dependencies to install.
   Dependent plugins and applications are listed if they will be installed, are currently installed, or need to be installed. If any plugins or applications must be installed, you must install them before you can install App Engine Studio.
4. **Optional:** If demo data is available and you want to install it, select the **Load demo data** check box.
   Demo data comprises the sample records that describe application features for the common use cases. Load the demo data when you first install the application on a development or test instance.
5. Select **Install**.

Result
App Engine Studio is installed on your instance.

Update a previously installed AES version

If you previously installed App Engine Studio (AES) from the ServiceNow Store and if a new version is available, then update the version in your instance.

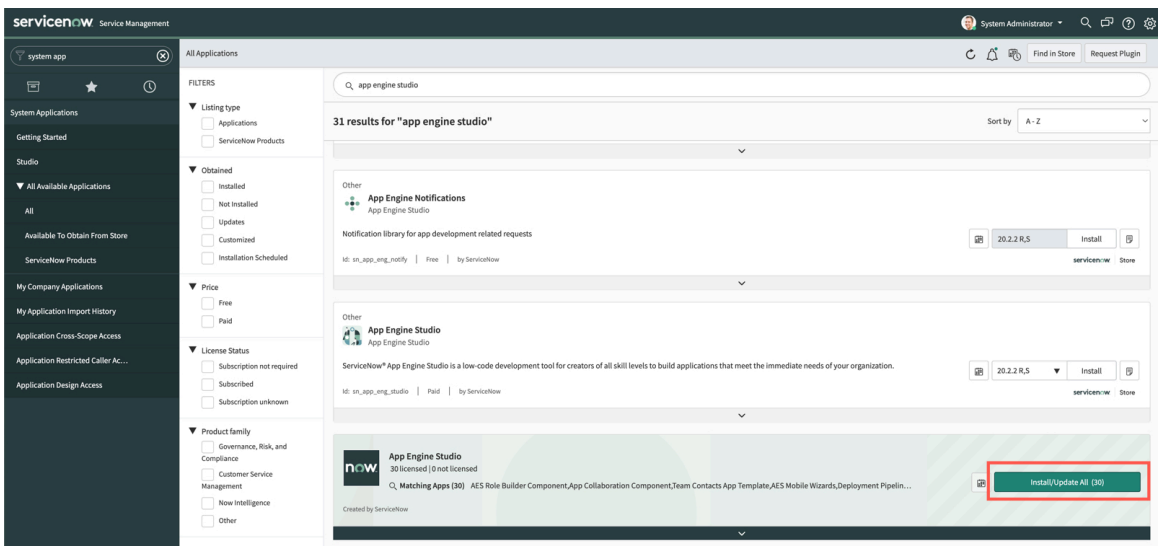
Before you begin

Note:
There should be no changes to the underlying applications or code bases when updating AES.

Role required: admin

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. Search for `App Engine Studio`.



3. Select the version that you want to update to, and select **Update** or **Install/Update All**.

Result
App Engine Studio automatically updates on the instance.

Upgrade your AES instance to the next family release

If you are currently running a family release (Rome, for example) and want to upgrade App Engine Studio (AES) to the next release, you don't need to acquire the applications from the ServiceNow Store.

The application is automatically updated when the platform is updated to the minimum required version.

Components installed with AES

Several types of components are installed with activation of the App Engine Studio (AES) plugin, including tables and user roles.

Note:
The Application Files table lists the components that are installed with this application. For instructions on how to access this table, see Find components installed with an application.

Roles installed

Role title [name]	Description	Contains roles
App Engine Studio admin [app_engine_admin]	Administers AES, including users, app configuration, and other management tasks. This role is assigned automatically to users in the App Engine Studio Admins group.	• sn_request_read • sn_request_write • sn_app_eng_studio.user
App Engine Studio app template admin [app_template_admin]	Administers the use, sharing, and activation or deactivation of templates. This role is assigned by an admin to individual users. For more information, see Manage template access.	• app-template-author • app-template-runner • flow-designer • flow-operator
App Engine Studio user [sn_app_eng_studio.user]	Builds applications in App Engine Studio. This role is assigned automatically to users in the App Engine Studio Users group. For more information, see Grant user access to AES.	• catalog_builder_editor • app_template_runner • sn_g_app_creator.app_creator

Tables installed

Table	Description
App Details [sn_app_eng_studio_app_details]	Details about the operations that a developer used to create an application in App Engine Studio. This table is updated automatically as developers build applications in App Engine Studio.
Applications in Projects [sn_app_eng_studio_project_application_m2m]	Applications that a developer creates in App Engine Studio. This table is updated automatically as developers build applications in App Engine Studio.
Environment [sn_pipeline_environment]	Instances that you use for the developing, testing, or launching an application. You update this table as you define environments for App Engine Studio.
Deployment Request [sn_deploy_pipeline_deployment_request]	Requests to review and publish an application that a developer created in App Engine Studio. From the deployment request form, a reviewer can deploy the application to different environments, accept or reject an application, and send feedback to a developer. For more information, see Managing app deployments using Pipelines and Deployments.
Pipeline [sn_pipeline_pipeline]	Configurations for deploying applications to different environments. There can be only one active pipeline at a time. You update this table as you create a pipeline for the deployment of applications from App Engine Studio. For more information, see Managing app deployments using Pipelines and Deployments.
Project [sn_app_eng_studio_project]	Details about App Engine Studio development sessions. A project is created automatically when a developer creates an application in App Engine Studio. This table is updated automatically as developers build applications in App Engine Studio.
Resources Content [sn_app_eng_studio_resources_content]	Help resources that a developer can access in App Engine Studio. This table includes configurations to support the default user experience for App Engine Studio.

Table	Description
Resources Content Topic [sn_app_eng_studio_resources_content_topic]	Categorizations of help resources in App Engine Studio. This table includes configurations to support the default user experience for App Engine Studio.
Taxonomy [sn_app_eng_studio_taxonomy]	Application files that a developer creates in App Engine Studio. This table is updated automatically as developers build applications in App Engine Studio.
Taxonomy Category [sn_app_eng_studio_taxonomy_category]	Categorizations for application files in App Engine Studio. By default, application files are categorized as data, experience, logic and automation, or security. This table includes configurations to support the default user experience for App Engine Studio.
Taxonomy Details [sn_app_eng_studio_taxonomy_details]	Details about application files that a developer creates in App Engine Studio. This table includes configurations to support the default user experience for App Engine Studio.

Note:
The following data preservers are added for tables related to pipelines:

- Environment
- Pipeline
- Pipeline Environment Order
- Pipeline Types

The data preservers prevent records in these tables from being overwritten during cloning on a non-production instance. For more information, see Data preservation on cloning target instances.

Configure App Engine Studio

App Engine Studio (AES) guided setup provides a sequence of tasks that help you configure AES on your ServiceNow instance.

App Engine Studio must be installed on the instances where you expect to be developing your applications. In earlier versions, you were required to install AES on all instances in your pipeline.

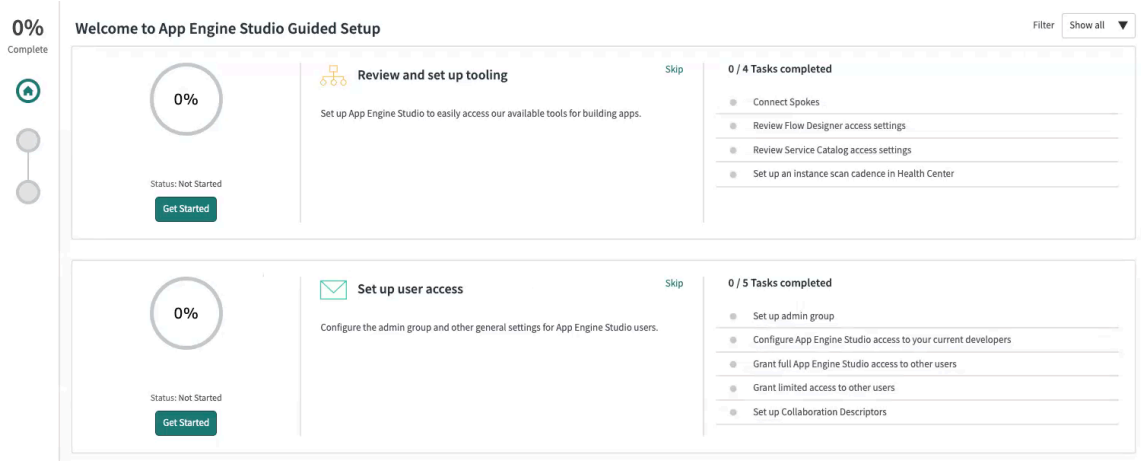
If you plan on cloning your production instance to one or more non-production instances, you should also install the AES product on your production instance prior to cloning. For more information, see System clone.

Learn more about AES configuration	Additional ServiceNow resources
ServiceNow provides several additional resources on configuring and administering App Engine Studio.	App Engine Studio release notes ServiceNow Community site App Engine Studio video on adding security

Guided setup to configure AES

To open App Engine Studio guided setup, navigate to **All** > **App Engine** > **App Engine Studio** > **Configuration** > **Guided Setup**.

The landing page provides information on the different tools and user access. Select the **Get Started** button in the top, right corner to start your configuration.



The App Engine Studio Guided Setup page provides a list of different categories. Select the **Get Started** button under each category to start configuring App Engine Studio.

Note:
If you have previously started any of the guided setup tasks, and then exited without completing them, the **Get Started** button is labeled **Continue**.

For detailed instructions on any of the tasks initiated from guided setup, see the Perform AES configuration tasks.

Configure AES personas and roles

The responsibilities of your staff are controlled by roles assigned to each member. Personas aren't explicitly part of App Engine Studio (AES) but administrators assign roles to give team members permission to configure or use AES.

Low-code/citizen developer

Low-code/citizen developers are tech savvy and interested in creating apps. Though they might not have formal coding or app development training, citizen developers can submit ideas for new apps and, if approved, build them using AES.

App Engine Studio admin

App Engine Studio admins manage all processes related to app development in AES. They review new app ideas, manage app development and deployment, and manage collaborators, usually in the App Engine Management Center.

App template admin

The app template admin makes sure that the right people have access to appropriate predefined and custom templates for app development in App Engine Studio.

App template author

The app template author creates and edits custom app templates in App Engine Studio and can share them with other users or groups.

Security admin

The security admin creates and modifies roles and access control lists for apps. This role is set on the platform level, and it is required for making updates to roles in App Engine Studio.

System administrator

The system administrator has access to all system features, functions, and data, regardless of security constraints. Grant this privilege carefully. If you have sensitive information, such as HR records, that you must protect, create a custom admin role for that area and train a person who is authorized to see those records to act as the administrator.

Professional ServiceNow developer

Professional ServiceNow developers can work in App Engine Studio, usually as collaborators with citizen developers. Because of their coding knowledge and development background, though, they're more likely to build and customize apps on the ServiceNow AI Platform, using more complex building tools.

Personas for AES

Persona	Roles required	Responsibilities
Low-code/citizen developer	App Engine Studio Users group (sn_app_eng_studio.user)	• Submit requests for new custom applications to build in App Engine Studio. • Understand ServiceNow and application development best practices. • Build and test applications in App Engine Studio.

Personas for AES (continued)

Persona	Roles required	Responsibilities
		• Submit App Engine Studio applications for IT review. • Maintain and change the application during its life cycle if determined during the intake process.
	App Engine Studio User Limited group (sn_app_eng_studio.user)	• Collaborate with other users on applications they've been given permission to see. • Submit App Engine Studio applications for IT review. • Understand ServiceNow and application development best practices. **Note:** Users in the App Engine Studio User Limited group can't create new apps.
App Engine Studio admin group	• atf_test_designer • scan_admin • app_engine_admin	• Manage App Engine Studio processes and properties. • Review and approve/reject submitted application requests based on intake guardrails defined by the system administrator. • Provision App Engine Studio user access. • Review submitted App Engine Studio applications. • Manage deployment requests. • Manage promotion of App Engine Studio applications across instances. • Execute ATF tests/suites in testing instances. • Ensure instance scans run with proper definitions.

Personas for AES (continued)

Persona	Roles required	Responsibilities
		• Collaborate with the system administrator to resolve application conflicts that arise on the platform.
App template admin	app_template_admin	• Control access to custom and predefined templates. • Activate and deactivate templates. • Share templates globally or with users and groups.
App template author	app_template_author	• Create and edit custom templates in App Engine Studio. • Share custom templates with users or groups.
Security admin	security_admin	• Create and modify roles and access control lists. • Required for making updates to roles in App Engine Studio.
System administrator	Admin	• Install and configure App Engine Studio and its dependent apps. • Define environments. • Configure pipelines. • Provision access to the App Engine Studio administrator group. • Upgrade the App Engine Studio application, when applicable. • Define guardrails for which AES administrators can approve or reject application intake requests. • Collaborate with professional ServiceNow developers to create

Personas for AES (continued)

Persona	Roles required	Responsibilities
		instance scan definitions for the platform. • Collaborate with AES administrators to resolve application conflicts that arise on the platform.
Professional ServiceNow developer	• atf_test_admin • delegated_developer • scan_admin • sn_app_eng_studio.user	• Build and test applications in ServiceNow Studio and App Engine Studio • Collaborate with citizen developers on an as-needed basis to deliver applications in App Engine Studio, including the following tasks: ○ Ensuring application development and organizational best practices are followed by citizen developers. ○ Helping with review and testing of submitted App Engine Studio applications. ○ Building complex configurations involving UI Builder, Mobile App Builder, Workflow Studio, or other builder tools. • Create ATF tests/suites for applications. • Collaborate with a system administrator to create instance scan definitions for the platform.

Perform AES configuration tasks

As you work through the App Engine Studio (AES) guided setup, you must perform different configuration tasks.

Setting up environments, tools, and user access

The tasks for configuring App Engine Studio are listed below, along with links to detailed instructions for completing them.

App Engine Studio configuration tasks

Task	Description
Install App Engine Studio in your development instances.	You can install the App Engine Studio application (com.snc.app-engine-studio) if you have the admin role. The application installs related ServiceNow® Store applications and plugins if they are not already installed. For detailed instructions, see Installing App Engine Studio.
Set up instance credentials.	For each instance that you're using, create a Connections and Credentials alias. Ensure that the alias type is **Credential**, not **Connection and Credential**. For detailed instructions, see Set up instance credentials.
Activate Playbooks.	You can edit processes in App Engine Studio, but you must activate Playbooks for App Engine Studio and enable the **Process Automation Designer for App Engine [com.glide.pad.license]** plugin to get started. For more information, see Activate playbooks.
Set up IntegrationHub spokes.	Activate spokes to enhance your Workflow Studio and App Engine Studio experience with integration-specific content. The following spokes are available for use with App Engine Studio. • Gmail spoke • Google Sheets spoke • Jira spoke • Microsoft 365 Excel spoke • Microsoft Teams spoke • Slack spoke • Twilio spoke • X Spoke (formerly Twitter Spoke) • Zoom spoke
Review access settings for Flow Designer in your development instance.	Enable your developers to use the editing capabilities that best suit them. For detailed instructions, see Review Flow Designer access settings.
Review access settings for Catalog Builder.	Enable your developers to add items to the appropriate catalogs and categories. For detailed instructions, see Review Catalog Builder access settings.

App Engine Studio configuration tasks (continued)

Task	Description
Set up an instance scan cadence in Health Center.	Schedule regular scans of your instance so that you can identify possible issues that arise from application development in App Engine Studio. For detailed instructions, see Set up an instance scan cadence in Health Center.
In your production instance, add users to the App Engine Studio Admin group.	Add members to the group so they can review app- and deployment-related requests. Also, define a contact email address for the group. For detailed instructions, see Add users to the App Engine Admin group.
In your development instance, add developers to the App Engine Studio Users group.	Enable developers in your organization to build applications in App Engine Studio. For detailed instructions, see Grant user access to AES.
In your development instance, add users to the App Engine Studio User Limited group.	Enable developers in your organization to collaborate on applications that someone else created in App Engine Studio. For detailed instructions, see Grant user access to AES.
Set up collaboration descriptors.	Set up custom collaboration descriptors in the global scope for use in App Engine Studio. For detailed instructions, see Set up custom collaboration descriptors.
In your non-production instance, specify an app template admin.	Grant the app_template_admin role to users who will manage template activation, deactivation, and sharing in App Engine Studio. For detailed instructions, see Manage template access.

Learn more about AES configuration	Additional ServiceNow resources
ServiceNow provides several additional resources on configuring and administering App Engine Studio.	 App Engine Studio release notes

Learn more about AES configuration	Additional ServiceNow resources
	 ServiceNow Community site
	 App Engine Studio video on adding security

Add users to the App Engine Admin group

Add users to the App Engine Admin group in your production instance to give them administrative rights to App Engine Studio (AES). You must also identify a contact email address for the group. After you configure the email address of the App Engine Admin group, members can receive notifications for app development-related requests, including app intake requests.

Before you begin
Role required: admin

About this task
If you are not using pipelines, you must also add users to the App Engine Admin group in your development instance.

Procedure
1. Navigate to **All** > **User Administration** > **Groups**.
2. Open the App Engine Admins group.
3. Select the **Group Members** related list and select **Edit**.
4. Select one or more names in the **Collection** list.
5. Enter a contact email address for the group.
6. Select **Add**.
7. Select **Save**.

Grant user access to AES

Control who has access to build applications in App Engine Studio (AES) by adding users to AES Users group or the AES User Limited group.

Before you begin
Role required: user_admin or admin

About this task
Grant access to build applications in AES by adding users to the App Engine Studio Users group in your organization's development instance. Each group member is automatically assigned the sn_app_eng_studio.user role.

To restrict access so that users can work in App Engine Studio but not create apps or see templates, add users to the App Engine Studio User Limited group.

For more information about the differences between the App Engine Studio Users group and the App Engine Studio User Limited group, see Configure AES personas and roles.

Procedure

1. Navigate to **All** > **User Administration** > **Groups**.
2. Add users by selecting either the **App Engine Studio Users** group or the **App Engine Studio User Limited** group.
   a. In the Group Members related list, select **Edit**.
   b. On the Edit Members page, move each developer from **Collection** to **Group Members List**.
   c. Select **Save**.
   d. On the group record, select **Update**.

Related topics

Components installed with AES

Manage template access

Control who has access to templates in App Engine Studio (AES) using the app_template_admin role. App template admins activate and deactivate templates and grant other users access to share their templates.

Before you begin
Role required: admin

About this task
An administrator grants the app_template_admin role to users who will manage template activation, deactivation, and sharing. This role should be granted on non-production instances where App Engine Studio users author and use templates.

To create apps and work with templates in AES, users must be an admin or in the App Engine Studio Users group. If they are in the AES User Limited group, they can only edit existing apps, not create new ones.

Procedure

1. Navigate to **All** > **User Administration** > **Users**.
2. Select the user you want to give the app_template_admin role to.
3. On the Roles related list, select **Edit**.
4. Move **app_template_admin** from the **Collection** list to the **Roles list**.
5. Select **Save**.

Set up custom collaboration descriptors

Customize the set of development collaborators you list in App Engine Studio (AES) using Guided Setup.

Before you begin
Role required: admin

About this task
Owner and Editor collaboration descriptors are provided out of the box in AES and have special capabilities. Development permissions included in each descriptor can be configured to your

platform needs. You can create custom descriptors as well, in the global scope, so they can be used in AES.

Note:
You should create collaboration descriptors in addition to Owner and Editor in the global scope. If you want collaboration descriptors to appear and be used in AES, you should also set them to `standard = TRUE`. AES doesn't support collaboration descriptors that are created in custom scopes, and non-standard collaboration descriptors don't render in AES.

Admins or delegated developers who invite collaborators can use these descriptors to delegate different types of work.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio** > **Guided Setup**.
2. Select **Get Started**.

   Note:
   If you have previously started any of the guided setup tasks, and then exited without completing them, the **Get Started** button is labeled **Continue**.

3. In the Set up user access section, select **Set up Collaboration Descriptors**.
4. In the Set up Collaboration Descriptors section, select **Configure**
5. Select **New**.
6. Give the new app collaboration descriptor a name and description.
7. Select **Submit**.

What to do next

Further customize collaborator permissions in App Engine Studio. For more information, see Change collaborator permissions.

Delegate developers using AES

Delegated development enables designated users without a system admin role to develop or deploy applications on the ServiceNow AI Platform. This enables administrators and delegated developers to work together to deliver custom applications through App Engine Studio (AES).

After an administrator reviews a request, they delegate the development work by providing a link to the developer that opens App Engine Studio in your organization's development environment. The administrator can:

- Send the link directly to a developer via email
- Post the link on a portal page
- Create a catalog item for delegated developers to request App Engine Studio access

Delegated developer role in AES

The delegated developer role enables an admin to elevate user permissions to a dedicated role with specific access for App Engine Studio.

A delegated developer has more permissions than a user, but less than an admin. The delegated developers each have their own permissions.

An admin may have several delegated developers, each with their own set of permissions. For example, you may want a designated developer for developing a time-off request, and another developer with separate permissions for creating support tickets.

Some AES features may be inaccessible to the delegated developer, depending on permissions. Your role enables you to access, edit, and add objects. If a feature of App Engine Studio is unavailable to you, contact your system administrator.

For more information about delegated development and permissions, see Delegated development and deployment.

Restricted ability to create apps

In addition to delegated development, admins can define whether AES users can create apps or only work on existing apps.

- Grant access to build applications in AES by adding users to the App Engine Studio Users group in your organization's development instance. Each group member is automatically assigned the sn_app_eng_studio.user role.
- To restrict access so that users can work in App Engine Studio but not create apps or see templates, add users to the App Engine Studio User Limited group.

AES integration with a Git source control repository

Enable application developers to integrate App Engine Studio (AES) with a Git source control repository to save and manage multiple versions of an application from a non-production instance.

Linking an application to source control enables all application developers on a non-production instance to:

- Import applications from a Git repository.
- Pull and apply remote changes from a Git repository.
- Commit all local changes on the instance to a Git repository.
- Create tags to permanently link to a given version of an application.
- Create branches to maintain multiple versions of an application simultaneously.

Integration requirements

To link an application to source control:

- The user must have the admin role.
- The non-production instance must have network access to the Git repository.
- Each application must be within its own Git repository.
- The repository user credentials must grant read and write access.

Note:
All application developers on the instance share a single set of credentials per repository.

Options available from App Engine Studio

After linking an application to source control, application developers can use App Engine Studio to manage the repository. From App Engine Studio, developers can:

- Edit the application repository credentials.
- Commit all local changes on the instance.
- Apply remote changes from the repository.
- Create a branch.
- Switch branches.
- Import an application from a remote repository.

Source control integration does not support managing applications on a production instance. Instead, you can manage applications on a production instance using the application repository, an update set, or App Engine Studio. For more information about managing applications on a production instance, see Application sharing.

Options available from a Git repository

The ServiceNow platform offers limited support for modifying linked application files outside of an instance. From Git, developers can:

- Move application files to a different Git directory structure.
- Edit application files outside of App Engine Studio.

The system generates a properties text file called `sn_source_control.properties` at the root level of the repository. To move application files to a different Git directory structure, application developers can set the *`path`* parameter to specify the subfolder path containing their application files. For example, if you moved your application to the `src/app` subfolder, set the *`path`* to `path=src/app`.

The system generates a `checksum.txt` file in the Git repository to determine if any application files have been changed outside of App Engine Studio. When the checksum value from the file matches the current checksum value, the integration skips the validation and sanitization process. When the checksum values do not match, the integration validates and sanitizes the application files as part of the source control operation. The sanitization process:

- Creates upgrade log entries for each sanitization action taken.
- Removes unsupported folders and files from the repository.
- Aborts all source control operations when a system application file fails XML schema validation. For example, if a database dictionary record fails XML schema validation, the system aborts all operations.
- Skips the current source control operation when a non-system application file fails XML schema validation.

The Git integration sanitizes only content within the application path listed in the `sn_source_control.properties` file. Repository content outside the application path is ignored.

MID Server support

Use an existing MID Server to connect to a source control repository. Connecting an application through a MID Server enables access to repositories behind a firewall.

Source control role permissions

For more information on roles and collaborators, see Application collaboration.

Link an application or application-customization to source control

Linking an application or application-customization to source control allows application developers to manage changes in App Engine Studio (AES) from a Git repository.

Before you begin

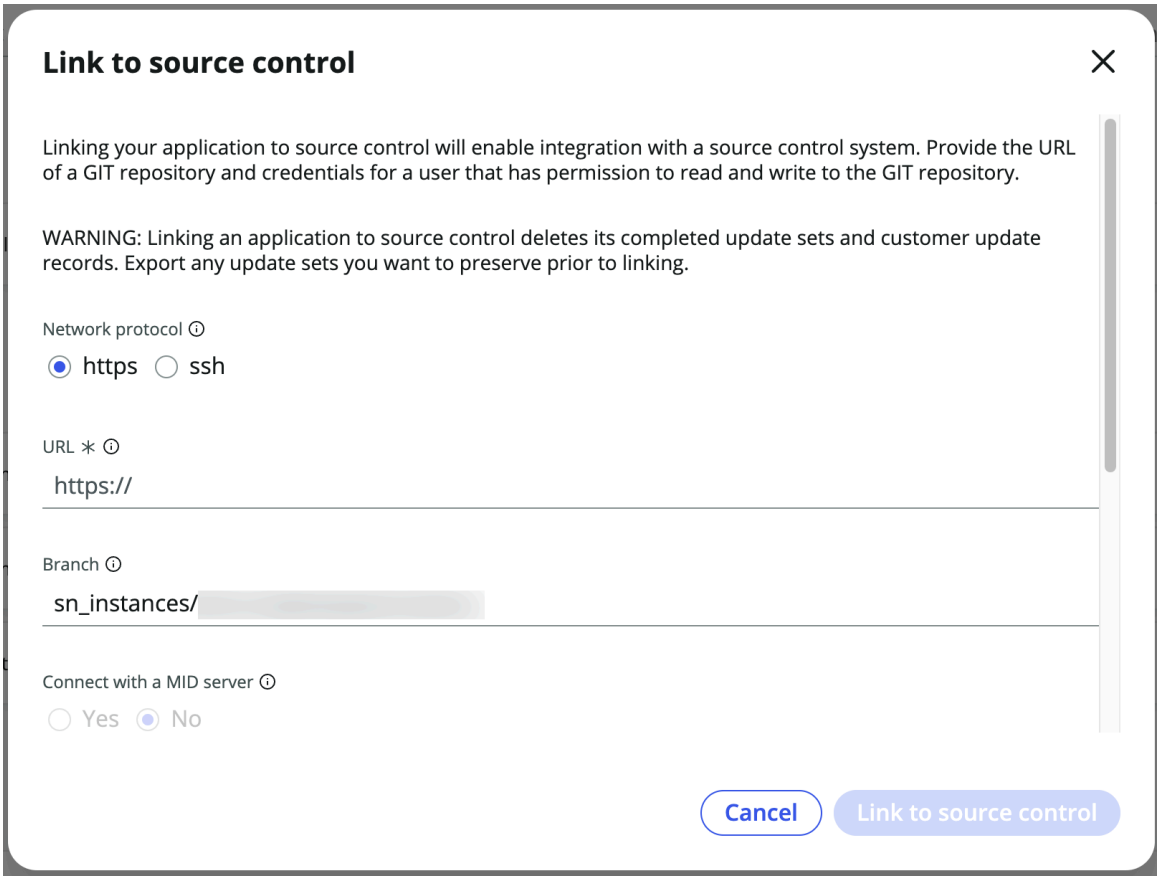
- Learn more about Manage customizations to applications.
- Create a dedicated Git repository for the application. For increased security, enable multi-factor authentication for the Git repository.
- Generate an access token that the source control integration can use instead of a password and multi-factor authentication passkey while creating a Credential record. Search for personal access token on GitHub or GitLab.
- Restrict permissions on the access token to allow read and write access to the Git repository.
- Verify that the non-production instance has network access to the Git repository.
- Ensure that users add the email address to their respective Users Table (ServiceNow sys_user) records that they use in their commits to the Git repository.
- Role required: admin

About this task

The source control integration does not support linking to an application or customization on a production instance. Instead, install applications on a production instance from the application repository, an update set, or the App Engine Studio.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select **Source control** > **Link to source control**.
   App Engine Studio displays the Link to source control dialog box.



4. Enter the connection details for the Git repository.

Source control connection details

Field	Description
Network protocol	Https or SSH credential type that enables secure channel data exchange.
URL	The URL to the Git repository where you want to save application files. For SSH protocol, use command to generate private key `ssh-keygen -t rsa -m PEM -b 4096 -C "email@address"`. **Note:** If the Git repo URL for SSH provided by your Git server does not work, check with your Git server owner or provider for the correct URL. There may be additional specifications such as scheme protocol prefixes, port numbers, and so on, required for your Git repo URL to function.
Branch	The repository branch to work on within the application. **Note:** This branch is used for commits. The default branch is set to "main" if it is not already set in the remote repository. If there is no default branch on the remote git repository, the instance creates a new default branch with the name "main". This is configured using the `glide.source_control.git_default_branch` system property.

Field	Description
MID Server Name	The name of the existing MID Server to link through. **Note:** Use a separate MID Server to prevent conflicts with Discovery activities. Be sure that the MID server user can create files to the sys_attachment table and that the table can accept files of the “bundle” type. Linking or an application through a MID Server enables access to repositories behind a firewall. See MID Server for more information.
Default email	The committer email address is defined by the sys_user record if available. But if a committer's sys_user record email field is empty, the system generates an alternate email (username@instancename.service-now.com). You can also enter a default email address and change it later. To use that default email address in all cases, select the check box.
Credential	The credential to be used with the selected protocol. See Getting started with credentials to learn more about creating credentials. **Note:** If you select the SSH network protocol, enter a valid credential of the SSH private key type. If you select the https protocol, enter a valid credential of the Basic Auth credentials type.
Commit Comment	An optional description of the repository or application.

Note:
All application developers on the instance share a single set of repository credentials.

5. Select **Link to source control**.
   The system validates the connection and user credentials and displays a success message.

   All application developers on the instance can use the linked Git repository to manage changes.

Related topics

MID Server

Getting started with credentials

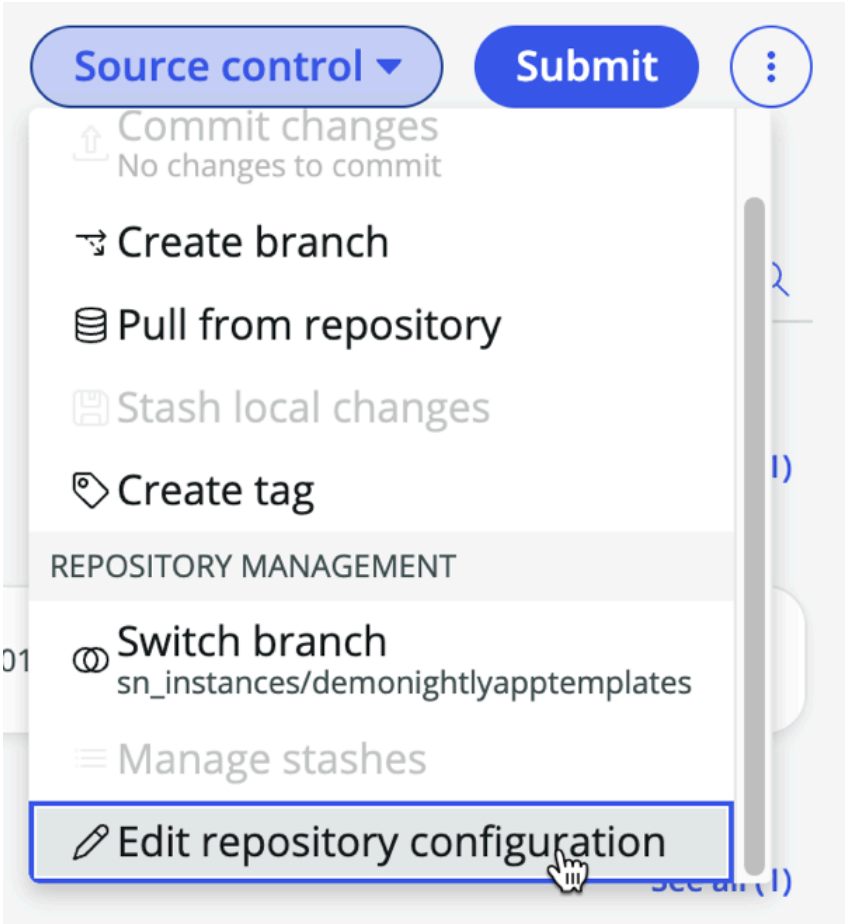
Edit a Git repository configuration

You can edit a Git repository's integration with App Engine Studio (AES) to change the network protocol selection, credentials or other field entries.

Before you begin
Role required: admin

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select **Source control** > **Edit repository configuration**.

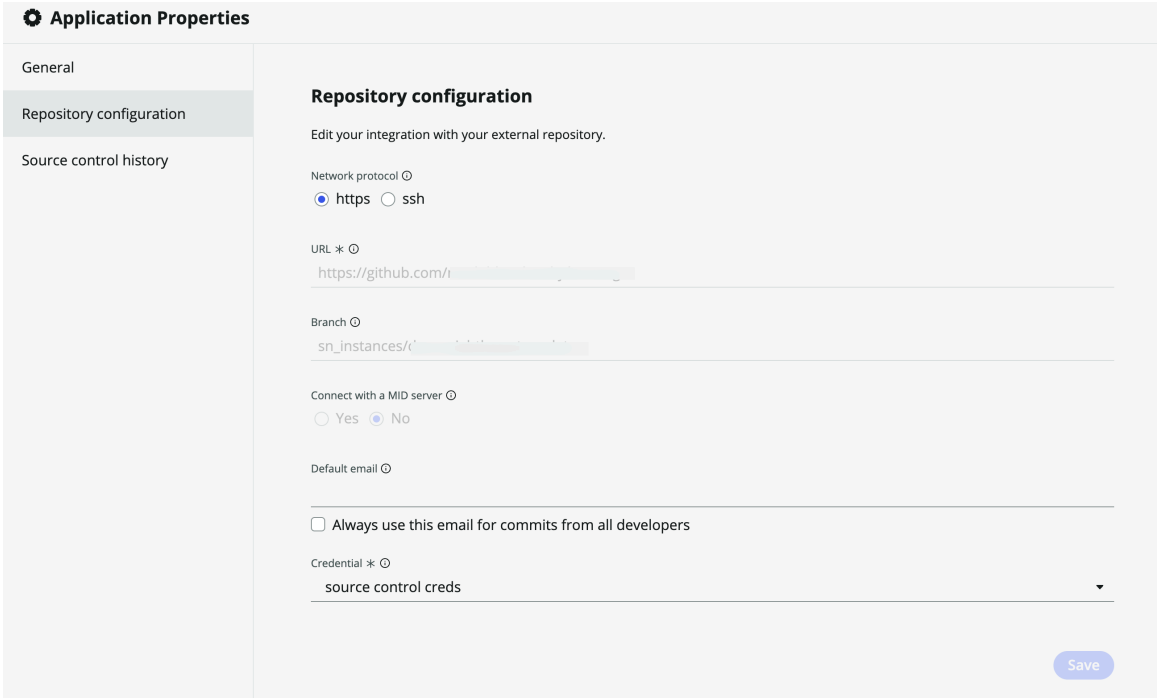


4. On the form, fill in the fields.

Repository configuration fields

Field	Definition
Network protocol	Protocol for source control, either **https** or **ssh**.
URL	URL address of your repository.
Branch	Current branch for the repository.
Connect with a MID server	Whether or not to use a MID server connection, which enables access to repositories behind a firewall. **Note:** If you have no MID server name, you can select a new one from the drop-down list. If you choose a new MID server, in the **Source control** menu before making any further source control operations to avoid errors.
Default email	Email address for the committer, which is defined by the sys_user record, if available.

Field	Definition
	If a committer's sys_user record email field is empty, the system generates an alternate email (username@instancename.service-now.com). You can also enter a default email address and change it later by selecting the **Always use this email for commits from all developers** check box.
Credential	Saved credentials to use for the source control connection. All application developers on the instance share a single set of credentials per repository. For information on working with credentials, see Getting started with credentials.



5. Select **Save**.

Configure the App Engine Management Center

Use the App Engine Management Center (AEMC) guided setup to step through the initial configuration of the Application Intake and Pipelines and Deployments applications. The Application Intake guided setup is optional, but if you want to use AEMC, the Pipelines and Deployments guided setup is required.

Before you begin

Ensure that AEMC is installed on your production instance and all of your non-production instances (development, test, and the like).

Role required: admin

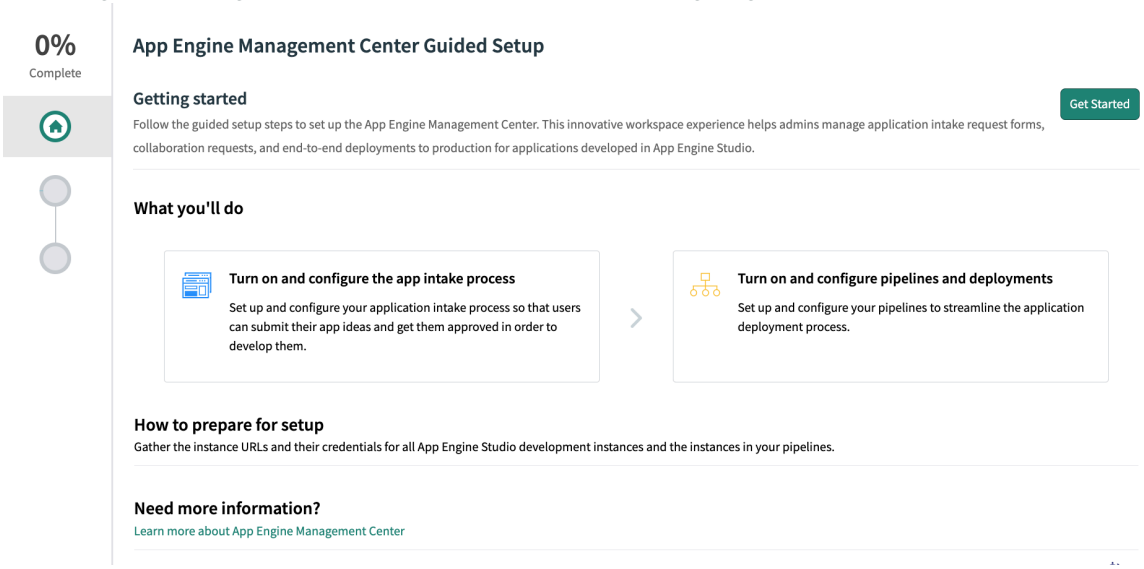
About this task

Essentially, the AEMC guided setup contains the guided setup steps for both the Application Intake and Pipelines and Deployments applications. If you have already configured one or both of these applications, you can skip those steps of the guided setup.

Procedure

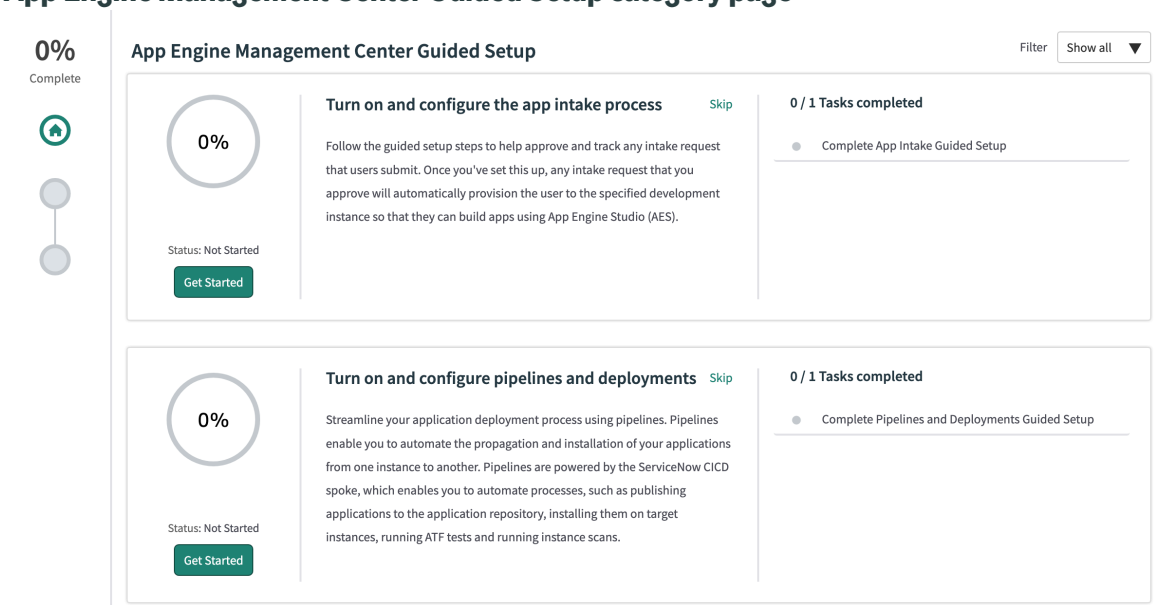
1. In your production instance, navigate to **All** > **App Engine** > **Administration** > **Guided Setup**.
   The landing page provides information on the two applications that you must configure, Application Intake and Pipelines and Deployments.

App Engine Management Center Guided Setup landing page



2. Initiate guided setup by selecting **Get Started**.
   The App Engine Management Center guided setup category page provides a list of different categories of tasks that you must complete before you can use AEMC.

App Engine Management Center Guided Setup category page



Note:
If you have previously started any of the guided setup tasks, and then exited without completing them, the **Get Started** button is labeled **Continue**.

3. Select the first **Get Started** button to initiate the Application Intake Guided Setup.
   There are several Application Intake configuration tasks you must complete.
   a. Activate the Apply for Citizen Development catalog item.
   b. Add users to the App Engine Admin group.
   c. Create development environment records.
   When you've completed all of the tasks in this category, the Category screen reopens.
4. Select the next **Get Started** button to initiate the Pipelines and Deployments Guided Setup.
   There are several Pipelines and Deployments configuration tasks you must complete.

   On your production instance:

   a. Configure environment credentials.
   b. Configure your pipeline environments.
   c. Configure your pipeline.
   d. Add users to the App Engine Admin group.
   e. Add ATF and instance scan suites for testing.

   > **Note:**
   > You can add ATF and instance scan suites for testing only if you've already created them and set up your testing instance. If you have not done this yet, you must come back to this step.

   f. Enable Change Management integration.
   g. Configure properties to integrate Change Management.

   On your testing instance:

   a. Configure environment credentials.
   b. Configure your controller instance.
   c. Enable Automated Test Framework (ATF) properties.
   d. Configure Automated Test Framework (ATF) suite.
   e. Configure Instance Scan suite.

   On your other, non-production instances:

   a. Configure environment credentials.
   b. Configure your controller instance.
   When you have completed the tasks in the second category, the Category page reappears.
   You have completed guided setup for App Engine Management Center.
5. **Optional:** Configure additional properties used to control system behavior in AEMC.

Test App Engine Management Center functionality on a non-production instance

Test App Engine Management Center (AEMC) on a non-production instance to confirm that everything is working as expected before moving to production.

Before you begin
Role required: admin

About this task
To test AEMC and pipelines before you proceed to a production environment, pick two non-production instances for testing purposes. Follow the Pipelines and Deployments guided setup

as if one of those non-production instances is a production instance. Because the system doesn't verify that the instance type on the environment record is actually a production instance, you can set the instance type to **Production** for the second non-production instance.

Procedure

1. Begin the Pipelines and Deployments Guided setup tasks as detailed in Pipelines and Deployments configuration tasks.
2. When you begin the tasks to configure your pipeline environments, select one of your non-production instances and designate it as a production instance in the pipeline environment record.
   For more information on the usual process, see Configure your pipeline environments.
3. Test AEMC and your pipeline functionality.
4. When you're done testing, change the **Instance type** of the non-production pipeline environment record to the correct type of instance.

Configure Application Intake

Use the App Engine Studio (AES) Application Intake guided setup to step through the initial configuration of the Application Intake application. Detailed instructions for each step are provided in subsequent sections of the product documentation.

Before you begin

Before you can use Application Intake to submit application ideas, you must ensure that the App Engine Studio application is installed.

Role required: admin

About this task

Application Intake guided setup provides a sequence of tasks that help you configure the Application Intake app on the ServiceNow AI Platform. For more information on each task, see Application Intake configuration tasks.

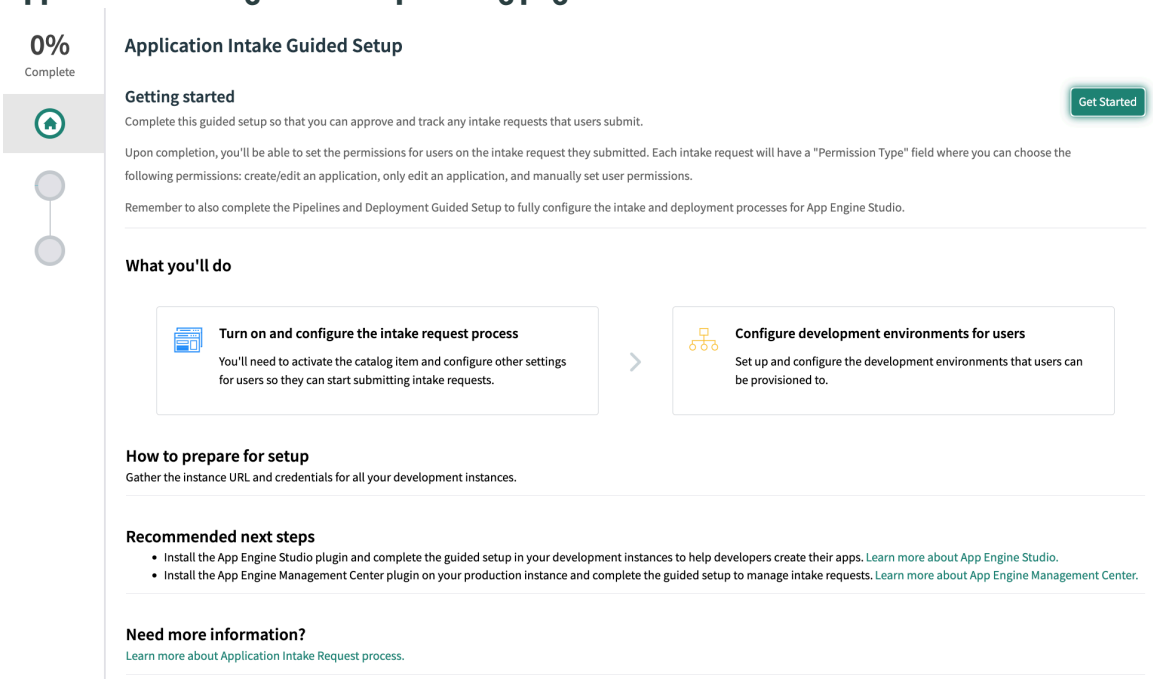
For general information about guided setup, see Guided setup .

Procedure

1. Navigate to **All** > **App Engine** > **Application Intake** > **Guided Setup**.

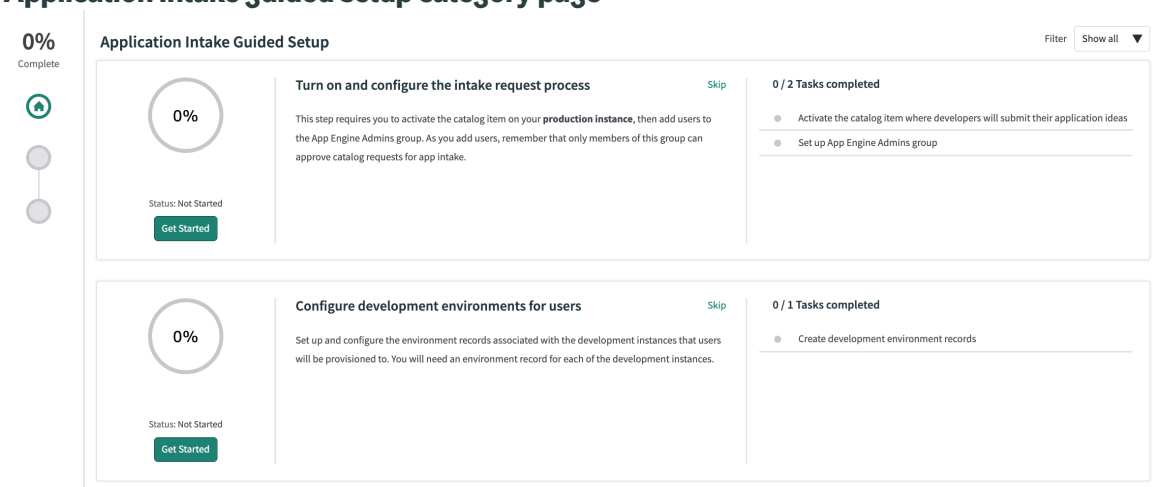
   The landing page provides information on the different categories, tools, and user access.

Application Intake guided setup landing page



2. To initiate guided setup, select **Get Started**.
   The Application Intake Guided Setup page provides a list of different tasks you must complete before you can use Application Intake to submit requests.

Application Intake guided setup category page



Note:
If you have previously started any of the guided setup tasks, and then exited without completed them, the **Get Started** button is labeled **Continue**.

3. Select the first **Get Started** button to initiate the Application Intake Guided Setup.
   There are several Application Intake configuration tasks you must complete.
   a. Activate the Apply for Citizen Development catalog item.
   b. Add users to the App Engine Admin group.
   When you've completed all of the tasks in this category, the Category screen reopens.
4. Select the next **Get Started** button to begin performing tasks for configuring development environments for your users.

On your production instance, Create development environment records for each development instance that you want to provision users to. This process allows your production instance to connect to your development instances.

Note:
If these records have already been set up in the Pipelines and Deployments Guided Setup, you can skip this step. When you have completed the tasks in the second category, the Category screen reappears.

Congratulations! You have completed guided setup for Application Intake.

Application Intake configuration tasks

As you work through the App Engine Studio (AES) Application Intake guided setup, you must perform different configuration tasks.

Activating and configuring the intake request process, and setting up development environments

The tasks for configuring Application Intake are listed below, along with links to detailed instructions for completing them.

Application Intake configuration tasks

Task	Description
Activate the catalog item where developers will submit their application ideas	Activate the Apply for Citizen Development catalog item so that citizen developers can submit their ideas for applications. For detailed instructions, see Activate the Apply for Citizen Development catalog item.
Set up the Apply for Citizen Development catalog item	For quick setup and use, use the predefined Apply for Citizen Development catalog item in Catalog Builder. To customize your app intake form, either update the Apply for Citizen Development catalog item, or use the AES App Intake template as a starting point and add questions and fields as needed. For detailed instructions, see Customize the App Intake form in Catalog Builder.
Add users to the App Engine Admins group	Add users to the App Engine Admin group. After the email address of the App Engine Admin group is configured, members can receive notifications for app development-related requests, including application intake requests. **Note:** If you already added users on the production instance using AES guided setup or Configure Pipelines and Deployments guided setup, you can skip this step. For detailed instructions, see Add users to the App Engine Admin group.
Create development environment records	On your production instance, create an environment record of type "Development" for each development instance that you want

Application Intake configuration tasks (continued)

Task	Description
	to provision users to. This will allow your production instance to connect to your development instances. **Note:** If these records have already been set up in the Pipelines and Deployments Guided Setup, you can skip this step. For detailed instructions, see Configure your pipeline environments.

When you have completed Application Intake guided setup, proceed to Pipelines and Deployments guided setup to fully configure App Engine Studio.

Activate the Apply for Citizen Development catalog item

Enable citizen developers to submit their ideas for applications by activating the Apply for Citizen Development catalog item. This is the first step in configuring the App Engine Studio (AES) Application Intake app.

Before you begin
Role required: admin

Procedure

1. Navigate to **All** > **App Engine** > **Application Intake** > **Guided Setup**.
   For more information, see Configure App Engine Studio.
2. Either start the process or continue with a process in progress.
   - To start the process, select **Get Started**.
   - If you previously completed one or more of the tasks, select **Continue**.
3. In the **Turn on and configure the intake request process** section, select **Get Started**.
4. In the **Activate the catalog item where developers will submit their application ideas** section, select **Configure**.
5. In the Apply for Citizen Development catalog item record, select the **Active** check box.
6. Select **Update**.
7. Close the browser window and return to the Application Intake guided setup.

Customize the App Intake form in Catalog Builder

Create a custom app intake experience for your organization by editing the fields and questions on the App Engine Studio (AES) App Intake form in Catalog Builder.

Before you begin
Role required: admin

About this task
The Out of the Box variables associated with this catalog item are set to read-only. To edit the App Intake form, you must create a copy of the existing **Apply for Citizen Development** catalog item. Then, disable the original catalog item, and continue editing the **Copy of Apply for Citizen Development** catalog item. For more information, see Copy a catalog item.

Procedure

1. Navigate to **All** > **Service Catalog** > **Catalog Builder**.
2. On the **Catalog Items** tab, select the copy of the **Apply for Citizen Development** item you created.
3. Select **Edit**.
4. Define the questions on the App Intake form on the **Questions** tab.
5. Edit any other settings on the form that you want to change.
6. On the **Review and submit** tab, review the form preview and make any needed corrections by selecting **Preview**.
7. Save your changes.
   - To save your changes without submitting the form, select **Save**.
   - To submit the form, select **Submit**.

Change the order of the questions in the App Intake form

Customize the App Intake form by changing the order in which questions appear.

Before you begin
Role required: admin

Procedure

1. Navigate to **All** > **Service Catalog** > **Catalog Builder**.
2. On the **Catalog Items** tab, select **Apply for Citizen Development** from the list of catalog items.
3. Select **Edit**.
4. On the **Questions** tab, point to the question you want to move.
5. On the left side of the question, select the Row drag and drop gripper icon (⠿) and drag the question to a new position.

Define dynamic behavior for an existing App Intake form question

Define dynamic behavior for a question on the App Intake form based on answers to other questions.

Before you begin
Role required: admin

About this task
Sometimes, if a user gives a specific answer to one question on the App Intake form, you might need a follow-up question to gather more information. For example, if you ask if they have taken any ServiceNow training, and the answer is "no", the user goes to the next visible question. If the answer is "yes", another question appears asking which training courses they have completed. Use the following steps to define this type of dynamic behavior.

Procedure

1. Navigate to **All** > **Service Catalog** > **Catalog Builder**.
2. On the **Catalog Items** tab, select **Apply for Citizen Development** from the list of catalog items.
3. Select **Edit**.

4. On the **Questions** tab, point to the question you want to update.
5. Select the Add dynamic behavior icon ( ) next to the question that you want to update.
6. Select **Define new behavior**.
7. On the **Actions** tab, specify the behavior of the question when certain conditions are met.

Actions tab

Field	Description
Specify the behavior of this question when the trigger conditions are met	
Make the question mandatory	Specifies how the dynamic behavior setting affects the mandatory state of the field, with the following options: ○ No action ○ Yes ○ No
Make the question visible	Specifies how the dynamic behavior setting affects the visible state of the field, with the following options: ○ No action ○ Yes ○ No
Make the question read-only	Specifies how the dynamic behavior setting affects the read-only state of the field, with the following options: ○ No action ○ Yes ○ No
Clear value	Option to clear the question value.

8. On the **Conditions** tab, specify the conditions that trigger the action on the question. For more information about building condition statements, see Condition builder.
9. On the **Settings** tab, specify the scenarios where the behavior setting applies.

Settings tab

Field	Description
Applicability	
Scenarios where this dynamic behavior applies	
Applies when the item is being requested	Option to specify if the dynamic behavior applies when the item is being requested.
Applies while viewing the catalog tasks after the request is submitted	Option to specify if the dynamic behavior applies while viewing the catalog tasks after the request is submitted.

Field	Description
Applies while viewing the requested item record after the request is submitted	Option to specify if the dynamic behavior applies while viewing the submitted record.

10. Select **Add behavior**.

Deactivate a question from the App Intake form

Deactivate a question from the App Intake form if it's no longer needed.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **Service Catalog** > **Catalog Builder**.
2. On the **Catalog Items** tab, select **Apply for Citizen Development** from the list of catalog items.
3. Select **Edit**.
4. On the **Questions** tab, point to the question you want to remove.
5. Remove the question from the form by selecting the Deactivate question icon ( ).
6. Select **Deactivate**.

Manage user groups for Application Intake

Control which user groups are available for admins to give Creator Studio and App Engine Studio app development permissions to during the Application Intake process. These groups are managed on the User Groups Permission Types [sn_app_intake_permission_type] table.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **sn_app_intake_permission_type.list**.
2. Select a record in the list to update it.
3. On the group record, change the status of the **Active** option to control whether the group is available in the Application Intake guided setup.

What to do next

Confirm that you see the correct active groups in an application request in App Engine Management Center.

Configure Pipelines and Deployments

Use the App Engine Studio (AES) Pipelines and Deployments guided setup to step through the initial configuration of Pipelines and Deployments. Detailed instructions for each step are provided in subsequent sections of the product documentation.

Before you begin

Role required: admin

About this task

The Pipelines and Deployments guided setup provides a sequence of tasks that help you configure Pipelines and Deployments on the ServiceNow AI Platform. For more information on each task, see Pipelines and Deployments configuration tasks.

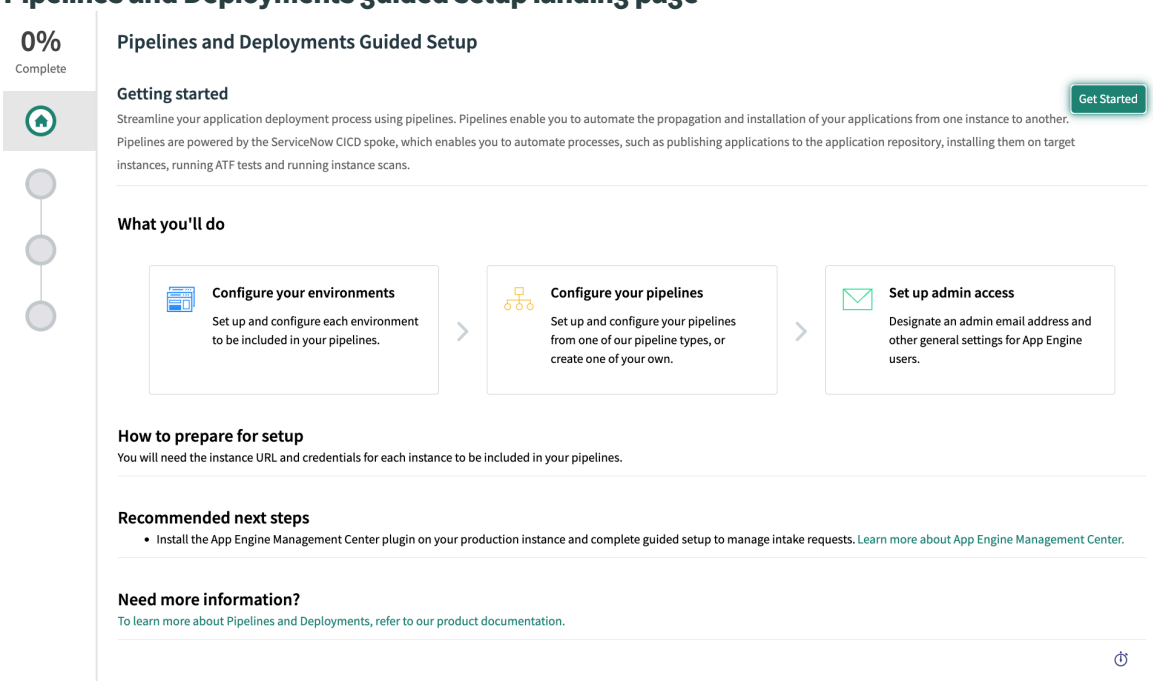
For general information about guided setup, see Using guided setup.

Procedure

1. Navigate to **All** > **App Engine** > **Pipelines and Deployments** > **Guided Setup**.

   The landing page provides information on the different categories, tools, and user access.

Pipelines and Deployments guided setup landing page

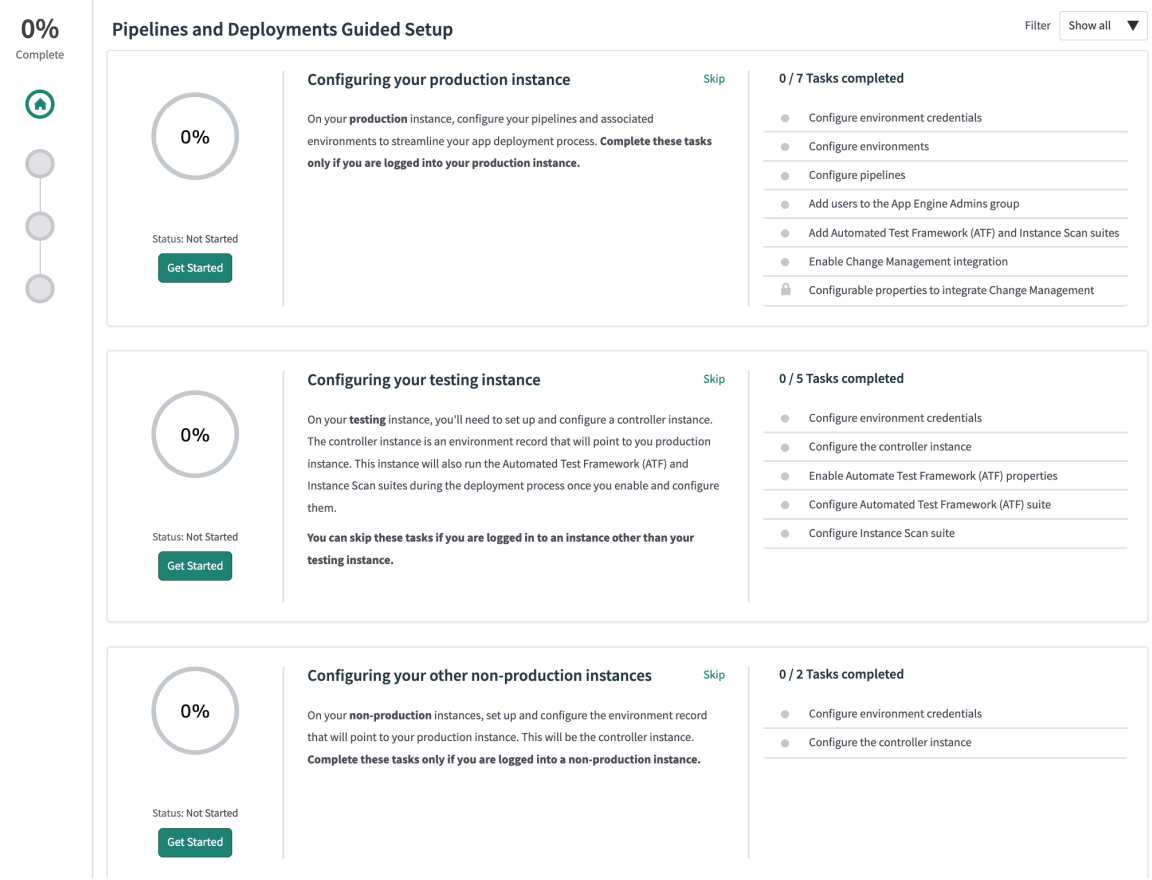


2. To initiate guided setup, select **Get Started**.

   The Pipelines and Deployments Guided Setup category page provides a list of different categories of tasks you must complete before you can use Pipelines and Deployments.

> **Note:**
> If you have previously started any of the guided setup tasks, and then exited without completed them, the **Get Started** button will be labeled **Continue**.

Pipelines and Deployments guided setup category page



Important:
Please be aware that each of the categories of tasks shown here are to be performed on different environments in your pipeline.

3. Select the first **Get Started** button to begin performing configuration tasks on your production environment.
   a. Configure environment credentials.
   b. Configure your pipeline environments.
   c. Configure your pipeline.
   d. Add users to the App Engine Admin group.
   e. Add ATF and instance scan suites for testing.

      Note:
      You can add ATF and instance scan suites for testing only if you've already created them and set up your testing instance. If you have not done this yet, you must come back to this step.

   f. Enable Change Management integration.
   g. Configure properties to integrate Change Management.

   When you have completed all of the tasks in this category, the Guided Setup screen reappears.

4. Select the next **Get Started** button to begin performing configuration tasks on your testing environment.
   a. Configure environment credentials.
   b. Configure your controller instance.
   c. Enable Automated Test Framework (ATF) properties.
   d. Configure Automated Test Framework (ATF) suite.
   e. Configure Instance Scan suite.

   When you have completed all of the tasks in this category, the Guided Setup screen reappears.

5. Select the last **Get Started** button to begin performing tasks for all of the other non-production (that is, test, development, staging, and so forth) instances in your pipeline.
   a. Configure environment credentials.
   b. Configure your controller instance.
   Congratulations! You have completed guided setup for the Pipelines and Deployments application.

Pipelines and Deployments configuration tasks

As you work through the Pipelines and Deployments guided setup, you must perform different configuration tasks on each of your instances.

Setting up environments, pipeline records, and user access

The tasks for configuring Pipelines and Deployments are listed below, along with links to detailed instructions for completing them. These tasks build on each other, so it's important to complete them in the sequence listed in the Guided Setup.

Pipelines and Deployments configuration tasks

Instance	Task	Description
Production	Configure environment credentials.	Configure credentials for each instance in your pipeline so that the flows used by AEMC can access the instances in your pipelines. For more information, see Configure environment credentials.
	Configure environments.	Set up your production and non-production (that is, development, test, and/or staging) environments by adding the URLs and credentials used to access each instance. For more information, see Configure your pipeline environments.
	Configure pipelines.	Set up and configure each pipeline by naming the pipeline, identifying each of the environments associated with the pipeline, and specifying the order in which apps are deployed between the environments in that pipeline.

Pipelines and Deployments configuration tasks (continued)

Instance	Task	Description
		For more information, see Configure your pipeline.
	Add users to the App Engine admin group.	Add users involved with this flow of tasks to the App Engine admin group. **Note:** If already added users on the production instance using AES guided setup or Configure Application Intake guided setup, you can skip this step. For more information, see Add users to the App Engine Admin group.
	Add Automated Test Framework (ATF) and Instance Scan suites.	Add your customized test suites as part of an application deployment in addition to the instance scan and ATF testing suites that are standard in AEMC. For more information, see Add ATF and instance scan suites for testing.
	Enable Change Management integration	Connect an existing Change Management program with AEMC so that deployments through the pipeline to the production environment are automatically scheduled based on the Change request state and planned change window. For more information, see Enable Change Management integration.
	Configure properties to integrate Change Management	Change requests and configuration items (CIs) are created using specifications set up in your existing Change Management program. If you want to change the change model, change template, or CI creation subflows used during app deployments, you must configure these properties during Guided Setup. For more information, see Configure properties to integrate Change Management.
Testing	Configure environment credentials	For more information, see Configure environment credentials.
	Configure the controller instance	Identify the controller instance that contains the pipeline deployment flows, as well as all of the data associated with the pipeline. As each instance completes its assigned tasks, it communicates with

Pipelines and Deployments configuration tasks (continued)

Instance	Task	Description
		the controller instance to determine where the next stage of deployment takes place. **Note:** Your production instance should be identified as the controller. For more information, see Configure your controller instance.
	Enable ATF properties on instances of type Testing.	Enable system properties that allow the ATF suite to run during the deployment process. If you do not enable these properties, a warning displays during the deployment process, but you can continue with the deployment. **Note:** If you plan on cloning your production instance to one or more non-production instances, you should either create a data preserver for these settings or enable these settings on your production instance. For more information, see System clone. For more information, see Enable Automated Test Framework (ATF) properties.
	Configure Automated Test Framework (ATF) suite	For more information, see Test your apps with the ATF.
	Configure Instance Scan suite	For more information, see Configuring Instance Scan.
Non-production instances that aren't of type Testing	Configure environment credentials	For more information, see Configure environment credentials.
	Configure the controller instance in your non-production instances	For more information, see Configure properties to integrate Change Management.

Configure environment credentials

Configure credentials in your production instance so that the flows used by the App Engine Management Center (AEMC) can access different instances.

Before you begin

You must use a functional user account with admin permissions to create the credentials for use in your pipelines. This user account should have the following characteristics:

- It should not be associated with a person in your organization.
- It should not need its password reset regularly.
- It should be controlled by your administrator.
- It should have admin privileges on all instances in the pipeline.

You can create a new functional user account specifically for pipeline credentials, or use one that you have previously created with these characteristics. Either way, this user account must be created before you can create the pipeline credentials. For more information about creating this user, see Create a user.

Role required: admin

About this task

Create the pipeline credentials with a functional user account that you created on the Users [sys_users] table. For example, you create a user account called Pipeline Runner and give it admin permissions. This account user name and password are used to set up the credential record.

Procedure

1. Navigate to **All** > **Connections & Credentials** > **Connection & Credential Aliases**.
2. Select **New**.
3. On the form, change the **Type** to **Credential**.
   Changing the type removes several fields from the form.
4. On the form, fill in the remaining fields.

Field	Description
Name	Unique name for the credential record. For example, Pipeline Credentials.
Application	Scope the record is being created in, which should be **Global**.
ID	Unique ID for the credential record. This field populates automatically based on the name. IDs are written in the form [scope_name.credential_name], unless the application is in the Global scope, then the ID is just the credential name.

5. Select **Submit**.
   The Connections & Credentials Aliases table reopens.
6. Open the credential that you created to add credential records to it.
7. On the **Credentials** tab, select **New**.
8. In the **What kind of Credentials would you like to create?** list, select **Basic Auth Credentials**.

   Note:
   If you want to use OAuth credentials in your pipelines instead, see Configure OAuth credentials for use in Pipelines and Deployments.

9. On the form, fill in the fields.

Field	Description
Name	Unique name for the basic auth credential record. For example, Pipeline Credentials.
Order	Order in which the flows you're using read the credential records. For example, if you have multiple credentials for backups, order them like 100, 200, 300, and so on. Flows read from the top of the list.
User name	User name for the functional account. For example, pipeline.runner.
Password	Password for the functional user account.
Active	Option to make the record active.
Credential alias	Alias for the record, repeated from the credential record ID. The alias exists so that only certain users can see what the user name and password are. When flows execute, they use the credential alias so they can install the apps.

10. Select **Submit**.
11. Repeat this process on each non-production instance in your pipeline.
    For the credential record, use the user name and password that the pipeline should use to access the production instance.

Configure OAuth credentials for use in Pipelines and Deployments

Use OAuth credentials in your pipelines to add another level of security to your pipeline. This task helps you create and configure OAuth credentials for use in your pipelines.

Before you begin

In the top right corner of your instance, make sure you set the application scope to **Global**. Open all your instances (development, test, production, and the like) in separate browser tabs. The following tasks lay out the steps for a three-instance environment. However, if you use any other non-production instances (stage, and the like), there are steps that detail where you may need to repeat a task on that instance.

Role required: admin

About this task

Creating and connecting OAuth credentials consists of several tasks that must be completed in the order laid out. Pay close attention to make sure you're in the correct instance for each step. For more information, see OAuth 2.0 .

Procedure

1. Complete all of the steps under Create OAuth API endpoints for external clients on the specified instances.
2. Complete all of the steps under Create third-party OAuth provider records on the specified instances.
3. Complete all of the steps under Use OAuth to create pipeline credentials on the specified instances.

What to do next

When you Configure your pipeline environments, make sure you select the correct instance credential records for the instance you're configuring.

Create OAuth API endpoints for external clients

Create OAuth API endpoints to enable your controller instance to have two-way communication with your non-production instances. Follow and complete each step carefully on the specified instances before moving on to create your third-party OAuth provider records.

Demonstration of how to create OAuth API endpoints for external clients

Before you begin

In the top right corner of your instance, make sure you set the application scope to **Global**.

Open all of your instances (development, test, production, and the like) in separate browser tabs.

Role required: admin

About this task

To create OAuth API endpoints for external clients and use OAuth in your pipelines, you must create several records, each on different instances in your pipeline. Begin on your production instance, which should be your controller instance.

Procedure

1. On your production instance, navigate to **All** > **System OAuth** > **Application Registry**.
2. Select **New**.
3. Select **Create an OAuth API endpoint for external clients**.
4. On the form, fill in the fields.

Application Registries form

Field	Action
Name	Enter `Pipeline Controller Client`.
Redirect URLs	**a.** Unlock the field. **b.** Enter the URL for your production, development, and test instances, each with `oauth_redirect.do` after the backslash. **c.** Lock the field. Separate each of the three URLs with a comma and a space. For example: `https://<production instance`

Field	Action
	`name>.service-now.com/oauth_redirect.do, https://<development instance name>.service-now.com/oauth_redirect.do, https://<test instance name>.service-now.com/oauth_redirect.do.`

5. Select **Submit**.

> **Important:**
> Complete the next steps on your development instance.

6. On your development instance, navigate to **All** > **System OAuth** > **Application Registry**.
7. Select **New**.
8. Select **Create an OAuth API endpoint for external clients**.
9. On the form, fill in the fields.

Application Registries form

Field	Action
Name	Enter `Pipeline Development Client`.
Redirect URLs	**a.** Unlock the field. **b.** Enter the URL for your production and development instances, each with `oauth_redirect.do` after the backslash. **c.** Lock the field. Separate the two URLs with a comma and a space. For example: `https://<production instance name>.service-now.com/oauth_redirect.do, https://<development instance name>.service-now.com/oauth_redirect.do.`

10. Select **Submit**.

> **Important:**
> Complete the next steps on your test instance.

11. On your test instance, navigate to **All** > **System OAuth** > **Application Registry**.
12. Select **New**.
13. Select **Create an OAuth API endpoint for external clients**.
14. On the form, fill in the fields.

Application Registries form

Field	Action
Name	Enter `Pipeline Test Client`.
Redirect URLs	**a.** Unlock the field. **b.** Enter the URL for your production and test instances, each with `oauth_redirect.do` after the backslash. **c.** Lock the field. Separate the two URLs with a comma and a space. For example: `https://<production instance name>.service-now.com/oauth_redirect.do, https://<test instance name>.service-now.com/oauth_redirect.do`.

15. Select **Submit**.
16. **Optional:** Repeat this process from steps 11–15 for any other non-production instances (staging, and the like) you have.

What to do next

Follow the steps in Create third-party OAuth provider records on the specified instances.

Create third-party OAuth provider records

Create third-party OAuth provider records to enable each of your instances to access the API endpoints you've created.

Before you begin

Complete the tasks in Create OAuth API endpoints for external clients.

In the top right corner of your instance, make sure you set the application scope to **Global**.

Open all of your instances (development, test, production, and the like) in separate browser tabs. Begin the steps below on the production instance.

Role required: admin

Procedure

1. On your production instance, navigate to **All** > **System OAuth** > **Application Registry**.
   You must create three records here, one for each of the three instances (development, test, and production). If you have any additional non-production instances (staging, and the like), create a record for each of those following the method shown here.

   Demonstration of how to create the first three records on your production instance
2. Select **New**.
3. Select **Connect to a third party OAuth Provider** to create a record for your development instance.

4. On the form, fill in the fields.

Application Registries form

Field	Action
Name	Enter `Dev Instance Connection`.
Client ID	**a.** On your development instance, open the Application Registry list (**All** > **System OAuth** > **Application Registry**). **b.** Open the **Pipeline Development Client** record. **c.** Copy the **Client ID**. **d.** On your production instance, paste the Client ID from your development instance into the **Client ID** field.
Client Secret	**a.** On your development instance, open the Application Registry list (**All** > **System OAuth** > **Application Registry**). **b.** Open the **Pipeline Development Client** record. **c.** Unlock the **Client Secret** field and copy the text. **d.** On your production instance, paste the Client Secret from your development instance in the **Client Secret** field.
Default Grant type	Change to **Authorization code**.
Authorization URL	**a.** Unlock the field. **b.** Enter the URL for your development instance followed by `oauth_auth.do`. **c.** Lock the field. For example: `https://<development instance name>.service-now.com/oauth_auth.do`.
Token URL	**a.** Unlock the field. **b.** Enter the URL for your development instance followed by `oauth_token.do`. **c.** Lock the field. For example: `https://<development instance name>.service-now.com/oauth_token.do`.

5. Select **Submit**.
6. Select **New**.
7. Select **Connect to a third party OAuth Provider** to create a record for your test instance.

8. On the form, fill in the fields.

Application Registries form

Field	Action
Name	Enter `Test Instance Connection`.
Client ID	**a.** On your test instance, open the Application Registry list (**All** > **System OAuth** > **Application Registry**). **b.** Open the **Pipeline Test Client** record. **c.** Copy the **Client ID**. **d.** On your production instance, paste the Client ID from your test instance into the **Client ID** field.
Client Secret	**a.** On your test instance, open the Application Registry list (**All** > **System OAuth** > **Application Registry**). **b.** Open the **Pipeline Test Client** record. **c.** Unlock the **Client Secret** field and copy the text. **d.** On your production instance, paste the Client Secret from your test instance in the **Client Secret** field.
Default Grant type	Change to **Authorization code**.
Authorization URL	**a.** Unlock the field. **b.** Enter the URL for your test instance followed by `oauth_auth.do`. **c.** Lock the field. For example: `https://<test instance name>.service-now.com/oauth_auth.do`.
Token URL	**a.** Unlock the field. **b.** Enter the URL for your test instance followed by `oauth_token.do`. **c.** Lock the field. For example: `https://<test instance name>.service-now.com/oauth_token.do`.

9. Select **Submit**.
10. On the Application Registry list, select the **Pipeline Controller Client** record.
11. Copy the **Client ID** and paste it somewhere like a notes application.
12. Unlock the **Client Secret** field, copy the text, and paste it in a note with the Client ID.
13. Go back to the Application Registry list and select **New**.

14. Select **Connect to a third party OAuth Provider** to create a record for your production instance.
15. On the form, fill in the fields.

Application Registries form

Field	Action
Name	Enter `Prod Instance Connection`.
Client ID	**a.** Copy the **Client ID** that you noted down. **b.** Paste the Client ID into the **Client ID** field.
Client Secret	**a.** Copy the **Client Secret** that you noted down. **b.** Unlock the **Client Secret** field, and paste the text in the field.
Default Grant type	Change to **Authorization code**.
Authorization URL	**a.** Unlock the field. **b.** Enter the URL for your production instance followed by `oauth_auth.do`. **c.** Lock the field. For example: `https://<production instance name>.service-now.com/oauth_auth.do`.
Token URL	**a.** Unlock the field. **b.** Enter the URL for your production instance followed by `oauth_token.do`. **c.** Lock the field. For example: `https://<production instance name>.service-now.com/oauth_token.do`.

16. Select **Submit**.

> **Important:**
> Complete the next steps on your development instance.

17. On your development instance, navigate to **All** > **System OAuth** > **Application Registry**.

    Demonstration of how to create records for your production instance on both your development and test instances

18. Select **New**.
19. Select **Connect to a third party OAuth Provider** to create a record for your production instance.
20. On the form, fill in the fields.

Application Registries form

Field	Action
Name	Enter `Prod Instance Connection`.
Client ID	**a.** Copy the **Client ID** that you noted down. **b.** Paste the Client ID into the **Client ID** field.
Client Secret	**a.** Copy the **Client Secret** that you noted down. **b.** Unlock the **Client Secret** field, and paste the text in the field.
Default Grant type	Change to **Authorization code**.
Authorization URL	**a.** Unlock the field. **b.** Enter the URL for your production instance followed by `oauth_auth.do`. **c.** Lock the field. For example: `https://<production instance name>.service-now.com/oauth_auth.do`.
Token URL	**a.** Unlock the field. **b.** Enter the URL for your production instance followed by `oauth_token.do`. **c.** Lock the field. For example: `https://<production instance name>.service-now.com/oauth_token.do`.

21. Select **Submit**.

> **Important:**
> Complete the next steps on your test instance.

22. On your test instance, navigate to **All** > **System OAuth** > **Application Registry**.
23. Select **New**.
24. Select **Connect to a third party OAuth Provider** to create a record for your production instance.
25. On the form, fill in the fields.

Application Registries form

Field	Action
Name	Enter `Prod Instance Connection`.
Client ID	**a.** Copy the **Client ID** that you noted down. **b.** Paste the Client ID into the **Client ID** field.

Field	Action
Client Secret	**a.** Copy the **Client Secret** that you noted down. **b.** Unlock the **Client Secret** field, and paste the text in the field.
Default Grant type	Change to **Authorization code**.
Authorization URL	**a.** Unlock the field. **b.** Enter the URL for your production instance followed by `oauth_auth.do` **c.** Lock the field. For example: `https://<production instance name>.service-now.com/oauth_auth.do`.
Token URL	**a.** Unlock the field. **b.** Enter the URL for your production instance followed by `oauth_token.do` **c.** Lock the field. For example: `https://<production instance name>.service-now.com/oauth_token.do`.

26. Select **Submit**.
27. **Optional:** Repeat steps 22–26 for any other non-production instances that you have (staging, and the like).

What to do next

Now that you've completed the pre-work for using OAuth, complete all the steps in Use OAuth to create pipeline credentials on the specified instances.

Use OAuth to create pipeline credentials

Create credential records on each of your instances to enable OAuth use in your pipeline.

Before you begin

Complete the tasks in Create OAuth API endpoints for external clients and Create third-party OAuth provider records.

In the top right corner of your instance, make sure you set the application scope to **Global**.

Open all of your instances (development, test, production, and the like) in separate browser tabs.

Role required: admin

About this task

To configure credentials correctly, you must create records for each of your production and non-production instances on your production instance. Then, you must create a record for your production instance on each of your non-production instances connecting them all together. Use the videos in each section to follow along with the steps. Begin on your production instance.

Procedure

1. On your production instance, navigate to **All** > **Connections & Credentials** > **Connection & Credential Aliases**.
   Create credential records on your production instance
2. Select **New**.
3. Change the **Type** to **Credential**.
4. In the **Name** field, enter `Pipeline Dev OAuth`.
5. Select **Submit**.
6. Reopen the record you just created (Pipeline Dev OAuth).
7. On the Credentials related list, select **New**.
8. Select **OAuth 2.0 Credentials**.
9. On the form, fill in the fields.

OAuth 2.0 Credentials form

Field	Action
Name	Enter `Dev OAuth`.
OAuth Entity Profile	Search for and select the **Dev Instance Connection default profile**.

10. Select **Submit**.
11. Reopen the **Dev OAuth** credential record.
    An expected error appears, directing you to verify the OAuth configuration and select **Get OAuth Token** to request a new token.
12. Under Related Links, select **Get OAuth Token**.
    A development instance opens requesting to connect to your ServiceNow account.
13. Select **Allow**.
14. Go back to the Connection & Credential Aliases list (**All** > **Connections & Credentials** > **Connection & Credential Aliases**).
15. Select **New**.
16. Change the **Type** to **Credential**.
17. In the **Name** field, enter `Pipeline Test OAuth`.
18. Select **Submit**.
19. Reopen the record you just created (Pipeline Test OAuth).
20. On the Credentials related list, select **New**.
21. Select **OAuth 2.0 Credentials**.
22. On the form, fill in the fields.

OAuth 2.0 Credentials form

Field	Action
Name	Enter `Test OAuth`.

Field	Action
OAuth Entity Profile	Search for and select the **Test Instance Connection default profile**.

23. Select **Submit**.
24. Reopen the Test OAuth credential record.
    An expected error appears, directing you to verify the OAuth configuration and select **Get OAuth Token** to request a new token.
25. Under Related Links, select **Get OAuth Token**.
    A test instance opens requesting to connect to your ServiceNow account.
26. Select **Allow**.
27. Go back to the Connection & Credential Aliases list (**All** > **Connections & Credentials** > **Connection & Credential Aliases**).
28. Select **New**.
29. Change the **Type** to **Credential**.
30. In the **Name** field, enter `Pipeline Prod OAuth`.
31. Select **Submit**.
32. Reopen the record you just created (Pipeline Prod OAuth).
33. On the Credentials related list, select **New**.
34. Select **OAuth 2.0 Credentials**.
35. On the form, fill in the fields.

OAuth 2.0 Credentials form

Field	Action
Name	Enter `Prod OAuth`.
OAuth Entity Profile	Search for and select the **Prod Instance Connection default profile**.

36. Select **Submit**.
37. Reopen the Prod OAuth credential record.
    An expected error appears, directing you to verify the OAuth configuration and select **Get OAuth Token** to request a new token.
38. Under Related Links, select **Get OAuth Token**.
    A production instance opens requesting to connect to your ServiceNow account.
39. Select **Allow**.
40. **Optional:** If you have any other non-production instances (staging, etc.), create a credential record for each of them on your production instance using the methods above.

> **Important:**
> After you've completed the steps above on your production instance, follow the steps below to create one credential record on each of your non-production instances connecting them to your production instance. Complete the next steps on your development instance.

41. On your development instance, navigate to **All** > **Connections & Credentials** > **Connection & Credential Aliases**.
    Create a credential record on your development instance and connect it to production.
42. Select **New** to create a record to connect your development instance to production.
43. Change the **Type** to **Credential**.
44. In the **Name** field, enter `Pipeline Prod OAuth`.
45. Select **Submit**.
46. Reopen the record you just created (Pipeline Prod OAuth).
47. On the Credentials related list, select **New**.
48. Select **OAuth 2.0 Credentials**.
49. On the form, fill in the fields.

OAuth 2.0 Credentials form

Field	Action
Name	Enter `Prod OAuth`.
OAuth Entity Profile	Search for and select the **Prod Instance Connection default profile**.

50. Select **Submit**.
51. Reopen the Prod OAuth credential record.
    An expected error appears, directing you to verify the OAuth configuration and select **Get OAuth Token** to request a new token.
52. Under Related Links, select **Get OAuth Token**.
    A production instance opens requesting to connect to your ServiceNow account.
53. Select **Allow**.

> **Important:**
> Complete the next steps on your test instance.

54. On your test instance, navigate to **All** > **Connections & Credentials** > **Connection & Credential Aliases**.
    Create a credential record on your test instance and connect it to production.
55. Select **New** to create a record to connect your test instance to production.
56. Change the **Type** to **Credential**.
57. In the **Name** field, enter `Pipeline Prod OAuth`.
58. Select **Submit**.
59. Reopen the record you just created (Pipeline Prod OAuth).
60. On the Credentials related list, select **New**.
61. Select **OAuth 2.0 Credentials**.
62. On the form, fill in the fields.

OAuth 2.0 Credentials form

Field	Action
Name	Enter `Prod OAuth`.
OAuth Entity Profile	Search for and select the **Prod Instance Connection default profile**.

63. Select **Submit**.
64. Reopen the Prod OAuth credential record.
    An expected error appears, directing you to verify the OAuth configuration and select **Get OAuth Token** to request a new token.
65. Under Related Links, select **Get OAuth Token**.
    A production instance opens requesting to connect to your ServiceNow account.
66. Select **Allow**.
67. **Optional:** If you have any other non-production instances (staging, etc.), create a credential record on that instance for production using steps 54-66.

What to do next

Now that you've created all of the credential records connecting your instances, you can use those records to configure your pipeline environments. For more information, see Configure your pipeline environments.

Configure your pipeline environments

Set up your App Engine Studio (AES) production and non-production (for example, development, test, and/or staging) environments by adding the URLs and credentials used to access each instance.

Before you begin

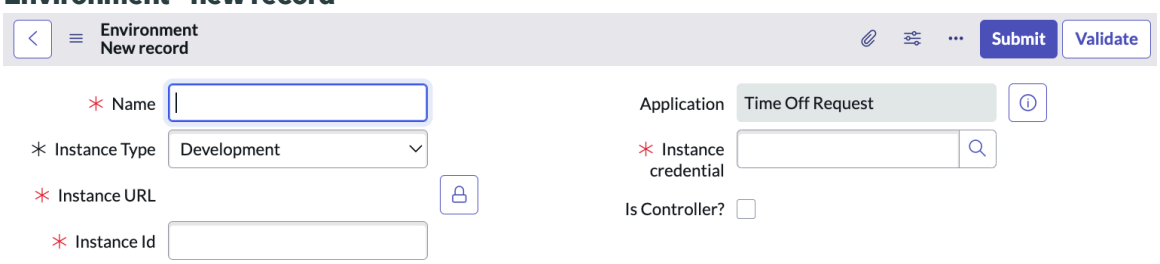
Role required: admin

> **Note:**
> Only users with the System Administrator (admin) role can define instance credentials for environments. Users with the App Engine Administrator [sn_app_eng_notify.app_engine_admin] role can view environment records; however, the **Instance credential** field is not visible.

Procedure

1. In your production instance, navigate to **All** > **App Engine** > **Pipelines and Deployments** > **Environments**.
2. Select **New**.

Environment - new record



3. On the form, fill in the fields.

Environment form

Field	Description
Name	Name of the environment. Enter a name that distinguishes the instance as a development, test, or production environment. For example, if you're defining a development environment, you could include `Dev` in the name.
Instance Type	The type of environment you are defining: ○ Sandbox ○ Development ○ Testing ○ Staging ○ Production
Instance URL	Web address of your ServiceNow instance.
Instance ID	The instance_id sys_property for the instance. **Note:** The Instance ID is automatically generated when you complete all of the other fields and select **Validate**.
Application	Application scope of the environment.
Instance credential	Authentication data related to the instance that you're configuring as an environment. **Note:** If you are using OAuth 2.0 credentials, select the appropriate credential for the instance record. For example, on your development instance record, select the OAuth Dev credential, and for your test instance record, select the OAuth Test credential.
Is Controller?	Identifies if this instance is a controller. This should be selected for the production environment record where you plan to manage deployment requests. For more information, see Configure your controller instance.

4. Select **Submit**.
5. Repeat the previous steps as you define each remaining environment in your pipeline. For example, if you defined a production environment, repeat the procedure again to define a development environment. Then repeat the procedure once more to define a test environment.

Configure your pipeline

Configure your App Engine Studio (AES) pipeline so that your administrator can quickly move an application from one environment to another.

Before you begin

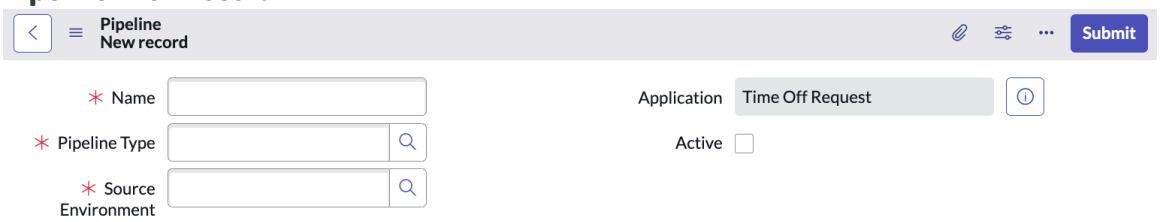
You must create all of your pipeline environment records before completing these steps. For more information, see Configure your pipeline environments.

Role required: admin or app_engine_admin

Procedure

1. In your production instance, navigate to **All** > **App Engine** > **Pipelines and Deployments** > **Pipelines**.
2. Select **New**.

Pipeline - new record



3. On the form, fill in the fields.

Pipeline form

Field	Description
Name	Name of the pipeline. Enter a name that distinguishes this pipeline from other pipelines.
Pipeline Type	The type of pipeline you are defining. The most common use case for pipelines is to select Application Deployment; however, you can define other types as needed.
Source Environment	The environment for this pipeline, usually the development environment.
Application	Application scope of the pipeline.
Active	Select the check box to activate this pipeline.

4. Select **Submit**.
5. Reopen the pipeline record you just created.
6. In the Pipeline Environments Order section, select the environments in the pipeline, excluding the **Source Environment**, and specify the order in which apps should be deployed through the pipeline.
7. Select **Update**.

Configure your controller instance

You must identify one of the instances in your App Engine Studio (AES) pipeline as the controller instance. All communication between the instances in your pipeline, including the deployment order for applications in the pipeline, takes place in the controller instance.

Before you begin

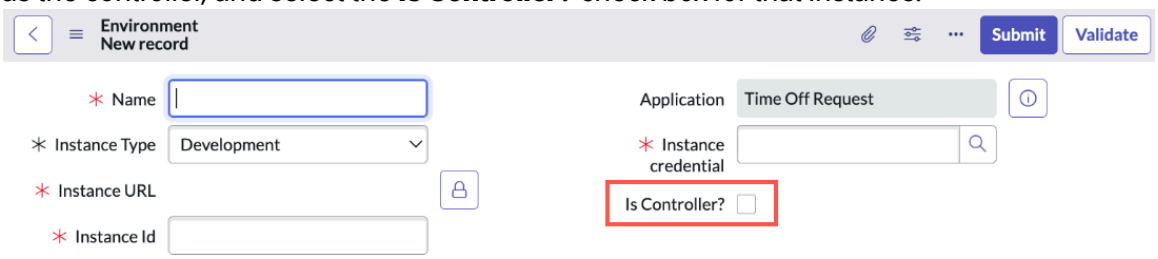
Role required: admin

About this task

Typically, your production instance should be identified as the controller instance. All request and approval records are stored on your controller instance.

Procedure

1. Navigate to **All** > **App Engine** > **Pipelines and Deployments** > **Environments**.
2. When you are configuring your pipeline environments, decide which instance you want to use as the controller, and select the **Is Controller?** check box for that instance.



3. Select **Save**.
   When the workflow for the pipeline runs, the non-production instances communicate with the controller instance to determine to which instance the app should next be deployed.

Add ATF and instance scan suites for testing

Add your customized test suites as part of an application deployment in addition to the instance scan and Automated Test Framework (ATF) testing suites that are standard in AEMC.

Before you begin

Because the ATF and instance scan suites exist only on testing instances, you must have your pipeline set up to complete this task. For more information, see Configure your pipeline environments.

Role required: admin

Procedure

1. On your production instance, navigate to **All** > **App Engine** > **Pipelines and Deployments** > **Scan Suites**.
2. If this is the first time you are looking at this table, select **Populate default suites** to load the instance and ATF scan suites that ServiceNow provides.
3. Add a new scan suite by selecting **New**.
4. On the form, fill in the fields.

Scan Suite form

Field	Description
Suite	Table with all the ATF and Instance scan suites that exists on your testing instance.
Error severity	Error options that indicate the severity of a scan suite not passing testing.
Instance Type	Type of instance the scan suite will run on. This field should be set to **Testing**.

5. Select **Submit**.
   The suites you add will run on apps installed on the instance you selected during testing.

Enable Automated Test Framework (ATF) properties

The App Engine Studio (AES) Pipelines and Deployments app includes an Application Test Framework (ATF) suite called the Application Deployment Test Suite. Two system properties control whether the test runs automatically whenever an app is deployed to a Test environment.

Before you begin
Role required: admin

About this task
The tests in the Application Deployment Test Suite can be run on production instances; however, the flows included in the base system run only on non-production (Test) instances. Additionally, the tests should run only on an instance defined with an **Instance Type** of **Testing**. For more information, see Configure your pipeline environments.

> **Note:**
> If you plan on cloning your production instance to one or more non-production instances, you should either create a data preserver for these settings or enable these settings on your production instance. For more information, see System clone.

Procedure

1. Navigate to the System Properties table by typing `sys_properties.list` in the navigation filter and pressing **Enter**.
2. Locate the following two properties:
   - sn_atf.runner.enabled
   - sn_atf.schedule.enabled
3. If you want these tests to run automatically on non-production instances whenever an app is deployed to a Test instance, open each property record, change the **Value** field to true, and select **Update**.

Enable Change Management integration

Enable Change Management integration with AEMC so that deployments through the pipeline to the production environment are automatically scheduled based on the Change request state and planned change window.

Before you begin
Role required: admin

About this task
This task can be completed within the Guided Setup system. This task opens within Guided Setup when you select **Configure**. If you prefer, you can follow the instructions below to access the same configuration task outside of the Guided Setup.

Procedure

1. Navigate to the System Properties table by typing `sys_properties.list` in the navigation filter and pressing **Enter**.
2. Locate the following property: sn_deploy_pipeline.change_management.enabled.
3. Connect your existing Change Management processes to AEMC by changing the **Value** field to `true`.

4. Select **Update**.
5. In the Pipelines and Deployments Guided Setup, select **Mark Complete**.
   Marking the enablement step complete enables you to configure properties related to Change Management integration including the change model, change template, and CI creation subflow. For more information, see Configure properties to integrate Change Management.

Configure properties to integrate Change Management

Configure predefined properties through the Pipelines and Deployments Guided Setup to tailor your Change Management and AEMC integration to your company's needs. Configuring these properties is optional.

Configure the change model

Configure the change model you would like to use during application deployments. If not configured, the **Normal** change model is used by default.

Before you begin
Role required: admin

About this task
There are several change models you can use to tailor change activities and flows for specific use cases. For more information about each type of model, see Change models. Use the steps below to select which change model you would like to use during app deployments.

This task can be completed within the Guided Setup system. This task opens within Guided Setup when you select **Configure**. If you prefer, you can follow the instructions below to access the same configuration task outside of the Guided Setup.

Procedure
1. Navigate to the System Properties table by typing `sys_properties.list` in the navigation filter and pressing **Enter**.
2. Locate the following property: sn_deploy_pipeline.change_management.default.model.
   To change the value of this property, and therefore the model that's being used, you must identify the **sys_id** of the change model you want to switch to.
   a. Navigate to **All** > **Change** > **Administration** > **Change Models**.
   b. Select the Change model you want to make the default for app deployments (for example, **Standard**).
   c. Right-click in the header bar for the record, and select **Copy sys_id**.
3. In the **sn_deploy_pipeline.change_management.default.model** property, paste the sys_id you have copied into the **Value** field, replacing its current contents.
4. Select **Update**.
5. In the Pipelines and Deployments Guided Setup, in the **Configure Change Model** section, select **Mark Complete**.

Configure the change template

Configure the default change template used to create change requests during application deployments.

Before you begin
Role required: admin

About this task

There are several templates that you can choose from to use as the default template for creating change requests. For more information, see Create a change request template. Use the following steps to select which change template you would like to use during app deployments.

This task can be completed within the Guided Setup system. This task opens within Guided Setup when you select **Configure**. If you prefer, you can follow the instructions below to access the same configuration task outside of the Guided Setup.

Procedure

1. Navigate to the System Properties table by typing `sys_properties.list` in the navigation filter and pressing **Enter**.
2. Locate the following property: sn_deploy_pipeline.change_management.default.template.
   To change the value of this property, and therefore the default template used for change requests, you must identify the **sys_id** of the referenced template in the change template that you want to switch to.
   a. Navigate to **All** > **Change** > **Standard Change** > **All Templates**.
   b. Select the update personalized list icon ( ) from the list header.
   c. In the Available column, select **Template** and use the move icon ( ) to add it to the Selected column.
   d. Select **OK**.
   e. Select the value in the **Template** field of the change template that you want to make the default for app deployments.
   f. Right-click in the header bar for the template record and select **Copy sys_id**.
3. In the **sn_deploy_pipeline.change_management.default.template** property, paste the sys_id you have copied into the **Value** field, replacing its current contents.
4. Select **Update**.
5. In the Pipelines and Deployments Guided Setup, in the **Configure Change Template** section, select **Mark Complete**.

Configure CI creation subflow

Customize how Configuration Items (CIs) are created during application deployment using AEMC.

Before you begin

Role required: admin

About this task

By default, the **Create CMDB CI if not present** subflow will run automatically during deployment to create a CI if for your application if it doesn't exist. However, you can change how Configuration Items (CIs) are created during deployment by using the following steps.

Procedure

1. Navigate to the System Properties table by typing `sys_properties.list` in the navigation filter and pressing **Enter**.
2. Locate the following property:
   sn_deploy_pipeline.change_management.config_ci_creation_subflow.
   To change the value of this property, and therefore the template that's being used, you must identify the **sys_id** of the subflow you want to switch to.

a. Navigate to **All** > **Flow Designer**.

b. Select the **Subflows** tab.

c. In the **Name** column, search for and open `Template: Create CMDB CI if not present` or `Create CMDB CI if not present`.

d. Select the More Actions menu icon ( ), and select **Copy subflow**.

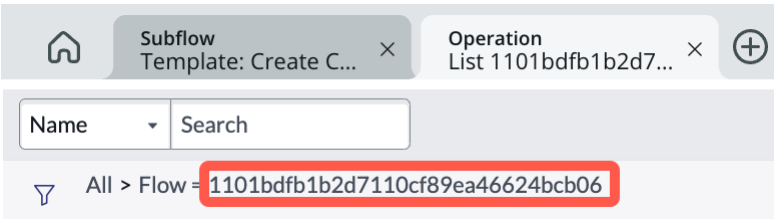
e. Enter a name for the subflow, and make sure the application is in the **Deployment Pipeline** scope.

f. Select **Copy**.

g. Modify the subflow, and select **Save**.

h. To find the sys_id of the subflow, select the More Actions menu icon ( ), and select **Executions**.

i. Copy the sys_id.
The sys_id is listed at the top of the page, after **All> Flow=**.



3. In the **sn_deploy_pipeline.change_management.config_ci_creation_subflow** property, paste the sys_id you have copied into the **Value** field, replacing its current contents.
4. Select **Update**.
5. In the Pipelines and Deployments Guided Setup, in the **Configure CI Creation Subflow** section, select **Mark Complete**.

Cloning instances with AES

Learn how to protect the data, tables, and templates you've created in App Engine Studio when using System Clone to copy instances from production to non-production.

Preserving data and tables when cloning

The following are requirements for cloning instances with AES:

1. Ensure that all of your AES plugins are installed across all instances.
2. If you're cloning a production instance, verify that you've set up a data preserver on the target instances to preserve the Automated Test Framework (ATF) and Instance Scan properties. For more information about data preservers, see Data preservation on cloning target instances.

> **Important:**
> By default, the ATF system property is disabled to prevent you from accidentally running these tests on a production system. Running ATF on a production instance is neither recommended nor supported due to the potential for data corruption or outages.

3. If you're collecting development and deployment data, the App Engine Management Center (AEMC) plugin must be installed on all instances.

Cloning data and tables from a production instance over a non-production instance can overwrite data in your non-production tables. To ensure that data isn't lost in development environments, create a cloning strategy for collaboration.

1. The following tables have data preservation to ensure that the tables are correctly cloned between instances:

   > **Note:**
   > For the following tables, preservation is for global scope only.

   - Collaboration Descriptor tables:
     - App Collaboration Descriptors (sys_appcollab_descriptor)
     - App Collaboration Descriptor Permissions (sys_appcollab_permission_m2m)
   - Collaboration Users and Groups tables:
     - App Collaboration Users (sys_appcollab_user)
     - App Collaboration Groups (sys_appcollab_group)

   The data preservation ensures data is retained on the tables in the development instances.
2. The following tables have clone exclusions:
   - Collaboration Descriptor tables:
     - App Collaboration Descriptors (sys_appcollab_descriptor)
     - App Collaboration Descriptor Permissions (sys_appcollab_permission_m2m)
   - Collaboration Users and Groups tables:
     - App Collaboration Users (sys_appcollab_user)
     - App Collaboration Groups (sys_appcollab_group)

   Clone exclusions ensure data from production instances isn't copied to development instances.
3. If AES is the only application using the Credentials table, consider creating data preservers for Credential Alias, Basic Auth, and Discovery credentials. Otherwise, you must ensure that these tables aren't overwritten when the production instance is cloned to non-production instances.
4. The following users must be reassigned their roles after cloning:
   - Users in the AES Users group
   - Users in the AES User Limited group
   - Users who have the sn_app_eng_studio.user role in non-production instances
5. After cloning, a **ReSync Collaborations Permissions** post-clone clean-up script runs automatically, so any applications that were the same on production and development instances are automatically have collaborators synced. Developers can resume development on them immediately.

Note:
The cloned instance must have the collaboration plugin enabled.

6. If some applications were backed up prior to cloning and retrieved after cloning, you can use the **Resync collaboration permissions** related link on the sys_app record to reassign users and groups to their appropriate delegated development permissions.
7. If a collaboration descriptor is no longer associated with a user or group after cloning (in the event that development apps were wiped out during cloning as they were not in the source instance), select the **Clean up records with empty references** related link to remove the unreferenced user or group from the collaboration table. You should run this UI action after the cloning is done and all preserved applications have been retrieved (with **Resync collaboration permissions** already run on them).

The following tables have data preservation to ensure that the tables are correctly cloned between instances:

- Pipeline Instance
- Request Authorization Key
- Deployment Request
- Deployment Environment Request

Preserving app templates when cloning

Admins must protect custom templates from being overwritten during the cloning process. Without protection, templates created in AES (both from existing applications and from scratch) are in danger of disappearing during a clone.

When you create a template in AES, a scoped app is automatically generated on the Custom Applications [sys_app.list] table in your instance. Though they have different contents, template applications and standard custom applications are treated similarly on the ServiceNow AI Platform. So, preserving app templates during a system clone works the same way as preserving an application.

To protect app templates on your non-production instances, follow the process in Preserve applications and customizations in development during a system clone.

More information on cloning and data preservation

See the following topics for more information on cloning and data preservation:

- For more information on using the System Clone tool, see System clone.
- For more information on data preservation, see Data preservation on cloning target instances.
- For more details on cloning instances with AES, see the App Engine Studio System Administrator Guide on ServiceNow University.

Learn more about cloning instances with AES	Additional ServiceNow resources
ServiceNow provides several additional resources on cloning instances with App Engine Studio.	 Cloning basics knowledge article
	 Cloning instances tips and tricks knowledge article
	 Extensive FAQ knowledge article on cloning instances
	 App Engine Enterprise - Data Preservation During System Clone Whitepaper **Note:** You must log in to ServiceNow University to access this resource.

Building apps in App Engine Studio

Meet the business needs of your organization by building custom applications in App Engine Studio (AES).

App Engine Studio build process

At a high level, there are seven main steps for building a new application in App Engine Studio. Select each step to see more information.

Application building process

1. Plan your app development
2. Submit your idea for app development
3. Delegate developers using AES
4. Create your app or Generate your app with Now Assist for Creator
5. Collaborate with other developers

6. Submit your app for approval and publishing
7. Managing app deployments using Pipelines and Deployments

Considerations

There are several considerations to keep in mind as you build applications in App Engine Studio:

- You access App Engine Studio by navigating to **App Engine Studio** > **App Engine Studio** or with the following web address: `<instance-name>.service-now.com/now/appenginestudio`. If you don't have a working link, contact your administrator.
- When you navigate back and forth between different applications in App Engine Studio, the application scope changes accordingly.

Working with data, experiences, automation, and security

App Engine Studio enables you to add as many data tables, types of experiences, logic and automation, and security roles as needed. However, you do not need to add all four categories to your app if the app doesn't need them.

App creation tutorial

This tutorial series guides you through the entire process of building an application using the low-code tools available in the ServiceNow AI Platform.

If you want to build an application that can solve your business problems, the app creation tutorial is your guide to building an application from start to finish. The tutorial walks you through each step of app development, from planning to building and testing. You also learn to use the powerful low-code app development tool, App Engine Studio, and its suite of integrated tools to build and enhance applications easily.

The use case for our tutorial is simple: We want to create an application that automates the employee travel request process for an organization. To do so, we build the application step-by-step, including the data model, user experience, logic and automation, and security for the application.

Each step in the tutorial demonstrates one aspect of app development. You can follow along with the tutorial sequentially or skip to the steps most relevant to you. Use the app creation tutorial and related resources to build your own application.

Complete the tutorial

Step 1	Planning your application
Step 2	Create an app
Step 3	Building a data model
Step 4	Creating user experiences
Step 5	Adding logic and automation
Step 6	Test your application

Planning your application

Plan out your application before building it to streamline the development process.

When you want to build an application, the first thing you should do is develop a plan for the app. Planning involves outlining what the application does and identifying who will use the

application. By planning your application, you can better understand the purpose of your application and address potential issues before development begins.

This is the first step in the app creation tutorial. Follow along to develop a plan for an application that manages employee travel requests for an organization.

Step 1		Planning your application
Step 2		Create an app
Step 3		Building a data model
Step 4		Creating user experiences
Step 5		Adding logic and automation
Step 6		Test your application

https://player.vimeo.com/video/964898255?badge=0&autopause=0&player_id=0&app_id=58479

Video sections

Timestamp	Section
0:17	Identify the use case for your application.
0:21	Visualize the workflow for your application.
0:42	Outline the functional requirements for your app.

Step 1: Identify the use case for your application

Start by identifying the use case for your application. In our tutorial, we're creating an application to manage employee travel requests for an organization. The app automates the entire travel request process and involves users such as employees, managers, regional heads, and travel desk agents as needed.

Consider the problem that you aim to solve with your application. To determine if your use case is a good fit for the tools featured in this tutorial, see Plan your app development.

Step 2: Visualize the workflow for your application

Once you have identified the use case for your application, visualize the end-to-end workflow for the application. The end-to-end workflow is the sequence of actions an application follows to complete a task or process.

The following visualization demonstrates the end-to-end workflow for the employee travel request application in our tutorial:

To visualize the end-to-end workflow for your application, map out what your application does from start to finish. Include any loops that lead back to earlier steps in the workflow and any actions that cause you to exit the workflow.

Step 3: Outline the functional requirements for your app

It's also important to consider the functional requirements of your application. The functional requirements of an application include who uses the application, what tasks they perform in the app, and what data the app must collect to support these tasks.

Use the following prompts to outline the functional requirements for your application:

1. Identify the personas who use your application and consider what tasks they perform.

   In our tutorial, the personas who use the employee travel request application and their associated tasks are:

Persona	Tasks
Employees	Submit travel requests
Managers and regional heads	Approve or deny travel requests
Travel desk agents	Book travel and complete follow-up tasks

2. Determine the data that your application must collect.

   For example, the application in our tutorial must collect details about employees' travel requests, including:

   - Where the employee is traveling to and from
   - When the travel occurs
   - How much the travel costs

3. Choose the process or processes that you want to automate in the workflow for your application using logic and automation.

   Review the following table to see how we plan to use logic and automation in our tutorial application.

	What it does	Tutorial application example
Logic	Controls how and when certain actions occur in your application workflow.	Verifies that an employee's travel request is only forwarded to the travel desk after it has been approved by both the manager and the regional head.
Automation	Sets up automatic processes based on the logic that you define.	Updates the employee's travel request case status and sends email notifications whenever a travel request is rejected.

4. Consider the roles and permissions for your application.

   Roles determine whether users have access to application content. Permissions are the individual abilities that a user has within application content, such as creating, reading, updating, and deleting. You can determine the roles for your application and configure permissions for each role at a later step in the tutorial. For more information about roles in App Engine Studio, see Configure AES personas and roles.

5. Think about how you want to organize the data collected by your application.

   The data that your application collects is automatically stored in tables. You might want to organize the data across different tables, depending on the processes that your application completes. You can decide how you want to organize the data for your application a later step in the tutorial. For more information about data organization in App Engine Studio, see Create a data model for your application.

Next steps

Once you've developed a plan for your application, you can begin building the application in App Engine Studio. Proceed to the next step: Create an app.

Create an app

When you're ready to start building your application, begin creating your application in App Engine Studio.

Before you begin

Note:
To create apps in AES, you must be an admin or in the App Engine Studio Users group. If you are in the App Engine Studio User Limited group, you can only edit existing apps, not create new ones.

Role required: admin, sn_app_eng_studio.user

About this task

Once you've developed a plan for your application, you can start building it in App Engine Studio. App Engine Studio is a powerful tool that enables you to develop robust applications in a low-code environment.

This is the second step in the app creation tutorial. Follow along to create an application in App Engine Studio.

Step 1	✓	Planning your application
Step 2	○	Create an app
Step 3	○	Building a data model
Step 4	○	Creating user experiences
Step 5	○	Adding logic and automation
Step 6	○	Test your application

https://player.vimeo.com/video/953654187?badge=0&autopause=0&player_id=0&app_id=58479

Video sections

Timestamp	Section
0:06	Open App Engine Studio.
0:10	Create an application.
0:14	Name your application and provide additional details.
0:24	Review, add, and customize roles.
0:36	Go to your application dashboard.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. On the App Engine Studio home page, select **Create app**.
3. Enter a unique **Name** and **Description** for your application.
   We enter the following details for the application in our tutorial:

Field	Value
Name	`Travel request`
Description	`Enables employees to submit travel requests`

4. **Optional:** Upload a logo for your application by dragging an image into the **Drag app logo or browse to upload** field, or by selecting the field, selecting the image from your file directory, then selecting **Open**.

   Tip:
   You can add a logo for your application later using the **Application Properties** menu.

5. Select **Continue**.

6. Control who has access to content in your application by reviewing, adding, or customizing roles.
   By default, App Engine Studio provides admin and user roles. To add more roles to your application, select **Add a role**, and enter a name and description for the role.
   In our tutorial, we use the default roles and permissions. We don't add additional roles.
7. Select **Continue**.
8. On the summary screen, select **Go to app dashboard** to continue setting up your application.
   Your application dashboard displays each part of your application, including **Data**, **Experience**, **Logic and automation**, and **Security**. You'll configure these parts of your app in later steps of the tutorial.

Result

You've created an application in App Engine Studio. The application is empty and ready for configurations.

What to do next

Now that your application is created, you can start building the data model for the application. Proceed to the next step in the tutorial: Building a data model.

Building a data model

After creating an application, start building the data model for the application. Learn about what a data model is and how we build the data model for the tutorial application.

Once you have created an application in App Engine Studio, you can start configuring each part of the application. In our tutorial, we start with the data model for the application.

This is the third step in the app creation tutorial. In this step, you learn about data models and build the data model for the employee travel request application.

Step 1	✓	Planning your application
Step 2	✓	Create an app
Step 3	○	Building a data model
Step 4	○	Creating user experiences
Step 5	○	Adding logic and automation
Step 6	○	Test your application

Overview of data models

The data model for an application describes the data that the application reads and writes and how the data is stored. To build a data model, you can create tables to store the data for your application. Each table contains rows that represent individual records and columns that define the type of information within each record. You can create as many tables as needed for your application and define relationships between the tables.

For more information about building data models in App Engine Studio, see Create a data model for your application.

Tutorial data model

In our tutorial, the data model for our application stores information about employees' travel requests, such as where the employee is traveling to and how much the travel costs.

We build the data model for the application in our tutorial in three phases:

Phase 1	Create a data table
Phase 2	Configure a data table
Phase 3	Share data between tables

Next steps

Start building the data model for your application. Proceed to the next step Create a data table.

Create a data table

Create a table to store the data for your application.

Before you begin

Role required: admin, sn_app_eng_studio.user, or delegated developer

About this task

The first phase in building the data model for the employee travel request application is to create a table. In our tutorial, we create a blank table that extends the Task table.

Phase 1		Create a data table
Phase 2		Configure a data table
Phase 3		Share data between tables

This procedure covers the first part of the task demonstrated in the tutorial video. The rest of the procedure is outlined in the next topic Configure a data table.

Follow along to create a table for the employee travel request application.

https://player.vimeo.com/video/967024604?badge=0&autopause=0&player_id=0&app_id=58479

Video sections

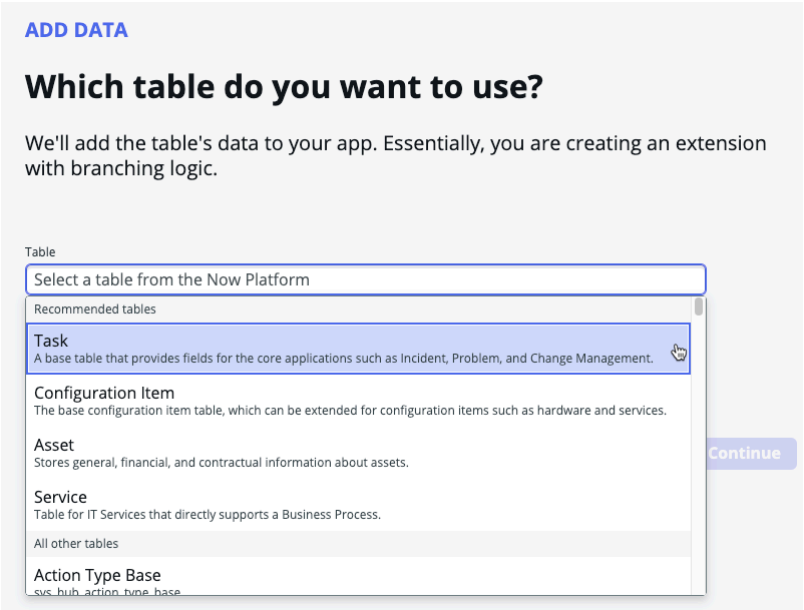
Timestamp	Section
0:10	Create a data table.
0:14	Create a table by extending the Task table.

Video sections (continued)

Timestamp	Section
0:21	Define table properties.
0:33	Add permissions to your table.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. Select the travel request application.
3. On the application dashboard, select **+ Add** next to **Data**.
4. Select **Create a blank table**.
   A blank table enables you to create the fields, rows, and columns in your table from scratch. You can import existing data, such as spreadsheets or PDFs, to create the table for your application.
5. Select **Continue**.
6. Select **Create from an extensible table**.
   Extensible tables are tables with built-in capabilities and functionality that you can use to create new tables. If you don't want to create a table from an extensible table, you can create a table from scratch by selecting **Create new table**.
7. Select **Task** from the **Table** list.



   The Task table is one of the commonly extended tables in the ServiceNow AI Platform. The Task table includes built-in functionality for approvals and assignments, which we need for the table in our tutorial application.

8. Select **Continue**.
9. Define the properties of your new table.
   We define the following properties for the table in our tutorial:

Table properties

Field	Description	Tutorial table properties
Table label	Unique label to identify the table.	`Travel request`
Table name	Database name for the table. Auto-populates based on the **Table label** that you provided.	No action needed
Table name prefix	Database prefix for the table. Auto-populates based on the application that you created the table in.	No action needed
Make extensible	Option to enable other tables to share data from this table. For more information, see Table extension.	No action needed
Auto-number	Option to track table records with a unique number.	Select the check mark
Prefix	Abbreviated name of the table to append to the beginning of record numbers.	`TRA`
Starting number	Number to identify the first record created for your table.	No action needed
Number of digits	Maximum number of digits for the record number.	No action needed

10. Select **Continue**.
11. Control who has access to content in the table by adding permissions to each existing role, or creating new roles.

> **Note:**
> At least one role must have read access for you to be able to preview the table.

In our tutorial, we grant the following permissions to the admin and user roles.

Role	Permissions
Admin	**All**
User	○ **Create** ○ **Read**

12. Select **Continue** to add the roles to your new table.
13. Select **Edit table** to continue setting up your new table.

Result
You have created a table for your application.

What to do next
Complete the setup of your table by proceeding to the next phase: Configure a data table

Configure a data table

Configure a table to capture the necessary data for your application.

Before you begin
Role required: admin, sn_app_eng_studio.user, or delegated developer

About this task
The second phase in building the data model for the employee travel request application is to configure the table so that it collects the necessary data for employee travel requests, such as location, dates of travel, and estimated airfare.

Phase 1		Create a data table
Phase 2		Configure a data table
Phase 3		Share data between tables

This procedure covers the second part of the task demonstrated in the tutorial video. The first part of the procedure is outlined in the previous topic Create a data table.

Follow along to configure a table to store the necessary data for the employee travel request application.

https://player.vimeo.com/video/967024604?badge=0&autopause=0&player_id=0&app_id=58479

Video sections

Timestamp	Section
0:48	Begin editing your table.

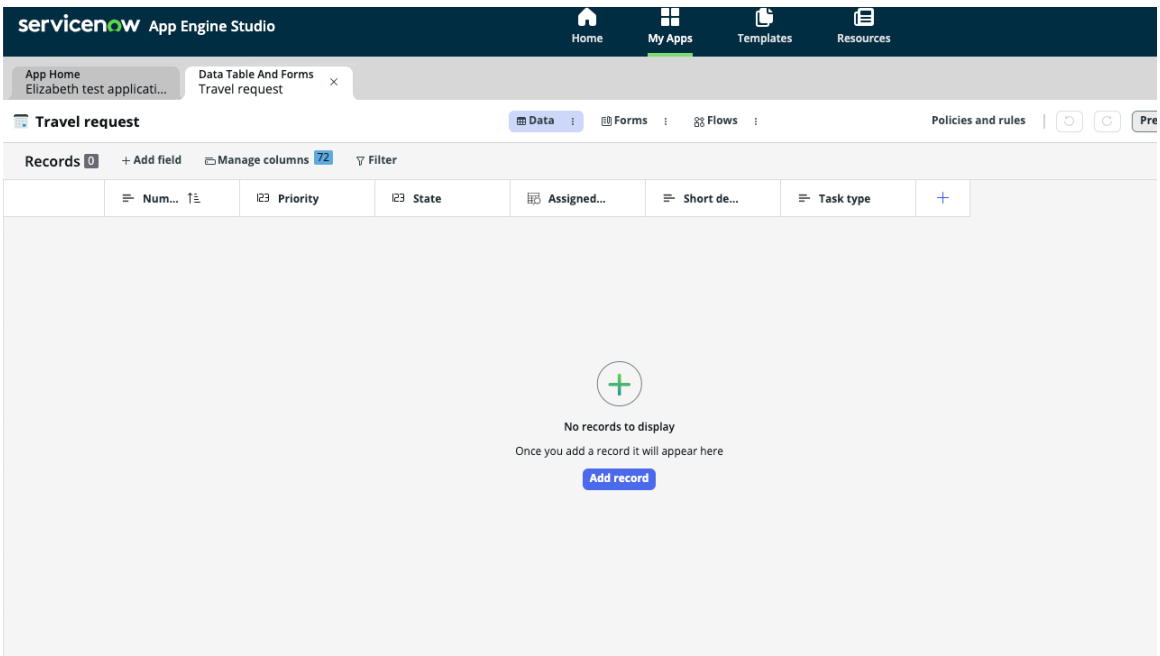
Video sections (continued)

Timestamp	Section
0:54	Add and configure fields in your table.
2:32	Change the data form layout.
2:39	Save your changes.

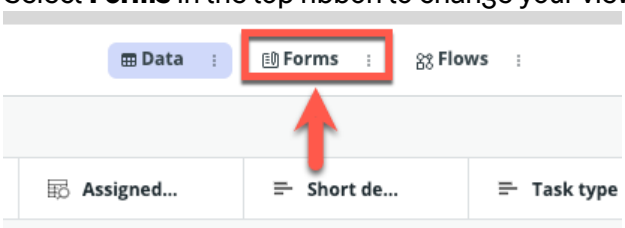
Procedure

1. Ensure that the employee travel request table is open.
   - To open a table from your instance home page, navigate to **All** > **App Engine** > **App Engine Studio**. Then select your application and select the table that you want to configure.
   - To open a table that you have just created, select **Edit table**.

   When your table is open, it appears in the canvas space.



2. Select **Forms** in the top ribbon to change your view to **Forms** view.



   Forms view enables you to create your table as a form. You can choose other views depending on the type of data that your table stores.

3. Remove any unnecessary fields from your table by hovering over the field, then selecting the delete field icon ( ).

> **Note:**
> When you create a table from an extended table, the table comes with preconfigured fields. You can choose to keep or remove any preconfigured fields in your own table.

In our tutorial, we remove the following fields:

- **Priority**
- **Configuration item**
- **Parent**
- **Assigned to**

4. Select **Save**.
5. Create new fields for your table.
   We must create several fields to collect details about employees' travel requests.

   a. Select **+ Add a field in the table** in the **Add form elements** panel.

   b. In the **Column label** field, enter `Departure Date`.

   > **Tip:**
   > The **Column name** field auto-populates based on the column label that you enter.

   c. In the **Type** field, select **Date**.

   d. **Optional:** To set additional properties for the field, select **Advanced settings**.
   We don't set additional properties for the fields in our tutorial. For more information about field types and advanced settings, see Field configuration in Table Builder.

   e. Select **Add**.

   f. **Optional:** Select **Add another one** to create additional fields.

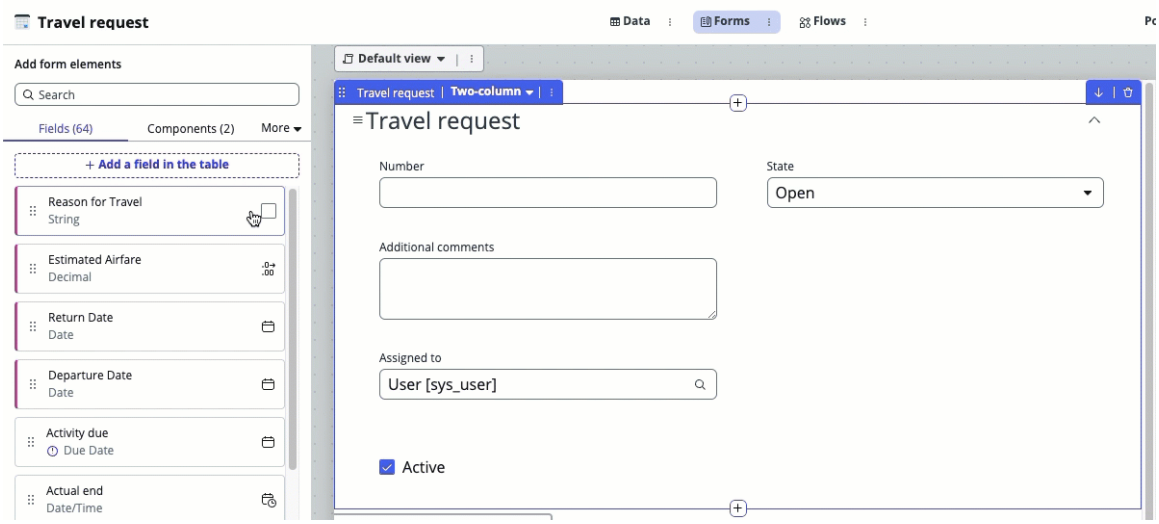
   g. Repeat the process in steps a-f to add as many fields as you need.
   We add more fields to collect additional information about employees' travel requests. The following table outlines the additional fields that we add to our table:

Column label	Type
`Return Date`	Date
`Estimated Airfare`	Decimal
`Reason for Travel`	String

   h. When you have added all of the new fields that you need, select **Done**.
   The fields that you have added will appear in the **Add form elements** panel.

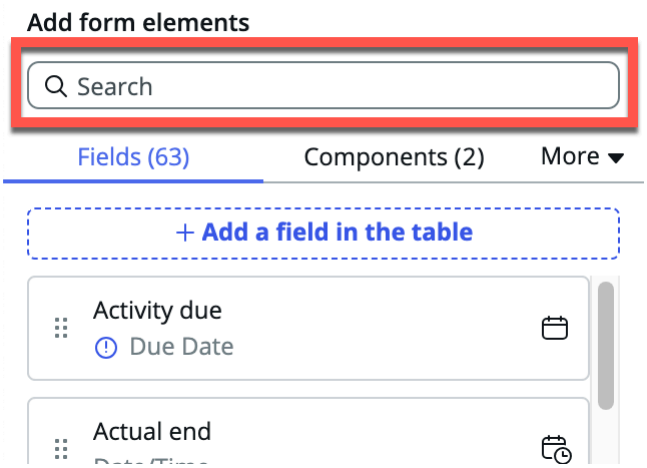
   i. To add the fields to your form, select the fields, then drag them into the **Default view** panel with your existing fields.

Tip:
To add multiple fields to your table at once, hover over the field, select the check mark in each field, then drag the fields into the **Default view** panel.



6. Select **Save**.
7. Add pre-existing fields to your table.

   a. Search for the fields that you want to add by entering the names of the fields in the search bar.



   In our tutorial, we add the following fields to our table:

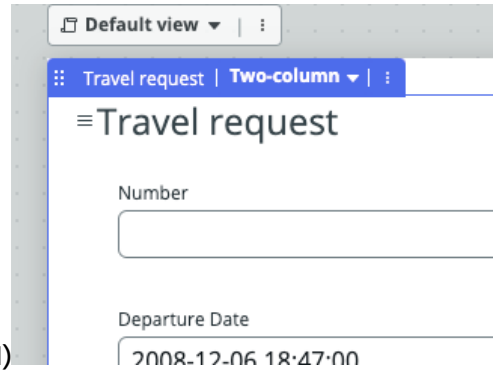
   - `Approval`
   - `Opened by`

   b. Select the field in the **Add form elements** panel, then drag the field into the **Default view** panel.

8. **Optional:** Change the layout of the form.

   a. Select your table in the **Default view** panel.
   When selected, your table appears with a blue border.

(Optional)



b. In the **Section** customization panel, select **One column**, then select **Two columns** to arrange the form fields evenly into a two-column layout.

> **Tip:**
> For more information about the customization options available in the **Section** panel, see Customize your form layout in Table Builder.

9. **Optional:** Arrange the table fields into the order that you want them to appear by moving the fields.
   In our tutorial, we arrange the fields into the following order:
   a. Number
   b. Opened by
   c. Departure Date
   d. Return Date
   e. Reason for Travel
   f. Estimated Airfare
   g. Approval
   h. Status
10. Select **Save**.
11. **Optional:** Select **Preview** to review how your form appears.

Result
Your table is configured to capture the necessary data for employee travel requests.

What to do next
To complete the data model for the employee travel request application, we must connect the travel request table to a separate table that stores airport data. Proceed to the next step: Share data between tables.

Share data between tables

Share data between tables to avoid data duplication.

Before you begin
Role required: admin, sn_app_eng_studio.user, or delegated developer

Select the following link to download the material needed for this step in the app creation tutorial.

App creation tutorial airport spreadsheet

About this task

The final phase in building the data model for the employee travel request application is to link the travel request table to a separate table that stores airport data. Doing so enables employees to choose which airports they want to use when they complete employee travel requests.

Phase 1	✓	Create a data table
Phase 2	✓	Configure a data table
Phase 3	○	Share data between tables

Storing data in multiple tables enables you to keep your data organized and modular. When you want to share data between tables you can create reference fields, which pull records from one table into another without duplicating the data.

Download the airport spreadsheet and follow along with the tutorial to create references fields between the employee travel request and airport tables.

https://player.vimeo.com/video/967025312?badge=0&autopause=0&player_id=0&app_id=58479

Video sections

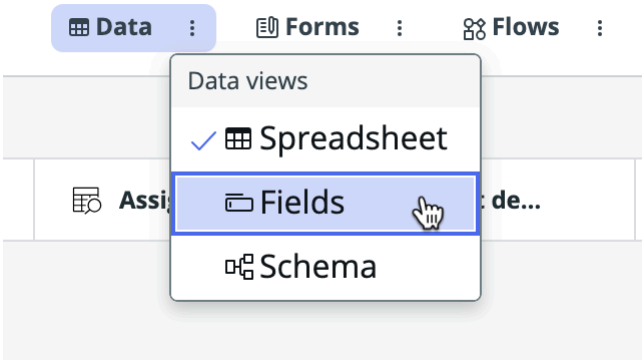
Timestamp	Section
0:03	Create a table to store the data that you want to reference.
0:26	Configure the table.
1:05	Open your existing table.
1:13	Add new fields to the existing table.
1:19	Set the field type to **Reference** and select the table that you want to pull data from.
1:36	Save your changes.
1:38	Add the new reference fields to your table in **Form** view.
1:59	Customize the form layout.
2:10	Save your changes.

Procedure

1. Create a new table to store airport information.
   In our tutorial, we create a new table from a spreadsheet that stores airport information. To follow along with the tutorial, download the airport spreadsheet. Then use the procedures in Import a spreadsheet and Create new table from spreadsheet import to create a new table from the airport spreadsheet.
2. Customize the table so that it displays the information that you want to reference.

a. Select the airport table from your application dashboard in App Engine Studio.

b. On the table landing page, select the More actions icon (⋮) next to **Data** in the top ribbon, then select **Fields**.



c. On the **Fields** page, in the **Name** field, toggle on the **Display** switch ( ) to enable other tables to reference airport names.

> **Important:**
> You can select only one field to be referenced by other tables. If you attempt to select multiple fields, Table Builder keeps only one of your selections when you save your changes.

d. Select **Save**.

3. Navigate to your application dashboard in App Engine Studio.
4. Select the employee travel request table, where you'll create the reference fields that pull data from the airport table.
5. On the table landing page, select the More actions icon (⋮) next to **Data**, then select **Fields**.
6. Create the reference fields.
   In our tutorial, we create two reference fields to collect airport choices for employee travel requests.

   a. On the **Fields** page, select **+ Add a field in the table**.

   b. In the **Column label** field, enter `Travel From`.
   The **Column name** field auto-populates based on the label that you entered.

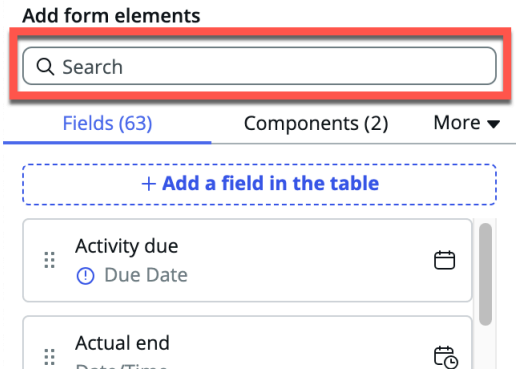
   c. In the **Type** column, select **Reference**, then select the airport table from the list.
   In our tutorial, we create another reference field to capture where the employee is traveling to. We label this field `Travel To`.

7. Select **Save**.
8. Add the reference fields to the form view of the table.

   a. Select **Forms** in the top ribbon.

   b. Enter the label of the reference field in the **Search** bar.

In our tutorial, the reference fields are labeled `Travel From` and `Travel To`.



c. Drag the reference fields from the **Add form elements** panel to the **Default view** panel.

9. **Optional:** Repeat the process in steps 6–8 to add as many reference fields as you need.
10. Select **Save**.

Result

The data model for the employee travel request application is complete.

What to do next

Continue building the employee travel request application by proceeding to the next step in the app creation tutorial: Creating user experiences.

Creating user experiences

Create the user experiences for your application. Learn about experiences in App Engine Studio and how we build the user experience for the employee travel request application.

We continue building our application by creating user experiences.

This is the fourth step in the app creation tutorial. In this step, you learn about user experiences and add an experience to the employee travel request application.

Step 1	✓	Planning your application
Step 2	✓	Create an app
Step 3	✓	Building a data model
Step 4	○	Creating user experiences
Step 5	○	Adding logic and automation
Step 6	○	Test your application

Overview of user experiences

User experiences, or interfaces, determine the options for users to access and engage with application content. There are several kinds of experiences that you can add to your application,

such as record producers and portals. For more information about experiences in App Engine Studio, see Add an application experience.

Tutorial user experience

In our tutorial, we add a record producer, which is one of the application experiences available in App Engine Studio. The record producer creates the form that employees will use to submit travel requests.

We build the experience for our application in two phases:

Phase 1	Add a record producer
Phase 2	Configure a record producer

Next steps

Start building the user experience for the employee travel request application: Add a record producer.

Add a record producer

Create a record producer to start building user experiences for the employee travel request application.

Before you begin
Role required: admin or catalog_admin

About this task
The first phase in building the employee travel request application's user experience is to select an experience. In our tutorial, we add a record producer, which enables us to create the form that employees will use to submit travel requests.

Phase 1		Add a record producer
Phase 2		Configure a record producer

This procedure covers the first part of the task demonstrated in the tutorial video. The rest of the procedure is outlined in the next topic Configure a record producer.

Follow along with the tutorial to add a record producer to the employee travel request application.

https://player.vimeo.com/video/953207342?badge=0&autopause=0&player_id=0&app_id=58479

Video sections

Timestamp	Section
0:06	Add an experience.
0:10	Select the record producer experience.

Video sections (continued)

Timestamp	Section
0:14	Enter a name and description for the record producer.
0:20	Create the record producer.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. Select the travel request application.
3. On the application dashboard, select **+ Add** next to **Experience**.
4. Select **Record producer**, then select **Begin**.
5. Define the properties of the record producer.
   a. In the **Name** field, enter `Raise a travel request`.
   b. **Optional:** In the **Short description** field, enter the following description of the record producer: `This request is for employees who are seeking approvals for travel requests`.
6. Select **Continue**.
7. Select **Edit record producer** to continue setting up your record producer.

Result

You have added a record producer to your application. The record producer is ready for configurations.

What to do next

Complete the setup of the record producer by proceeding to the next phase: Configure a record producer.

Configure a record producer

Configure the record producer so that it contains the necessary questions for employee travel requests.

Before you begin

Role required: admin or catalog_admin

About this task

The second phase in building the user experience for the employee travel request application is to configure the record producer. We configure the record producer so that it contains the necessary questions for employee travel requests, such as travel dates, location, and estimated airfare.

Phase 1		Add a record producer
Phase 2		Configure a record producer

This procedure covers the second part of the task demonstrated in the tutorial video. The first part of the procedure is outlined in the previous topic Add a record producer.

Follow along to configure the record producer for the employee travel request application.

https://player.vimeo.com/video/953207342?badge=0&autopause=0&player_id=0&app_id=58479

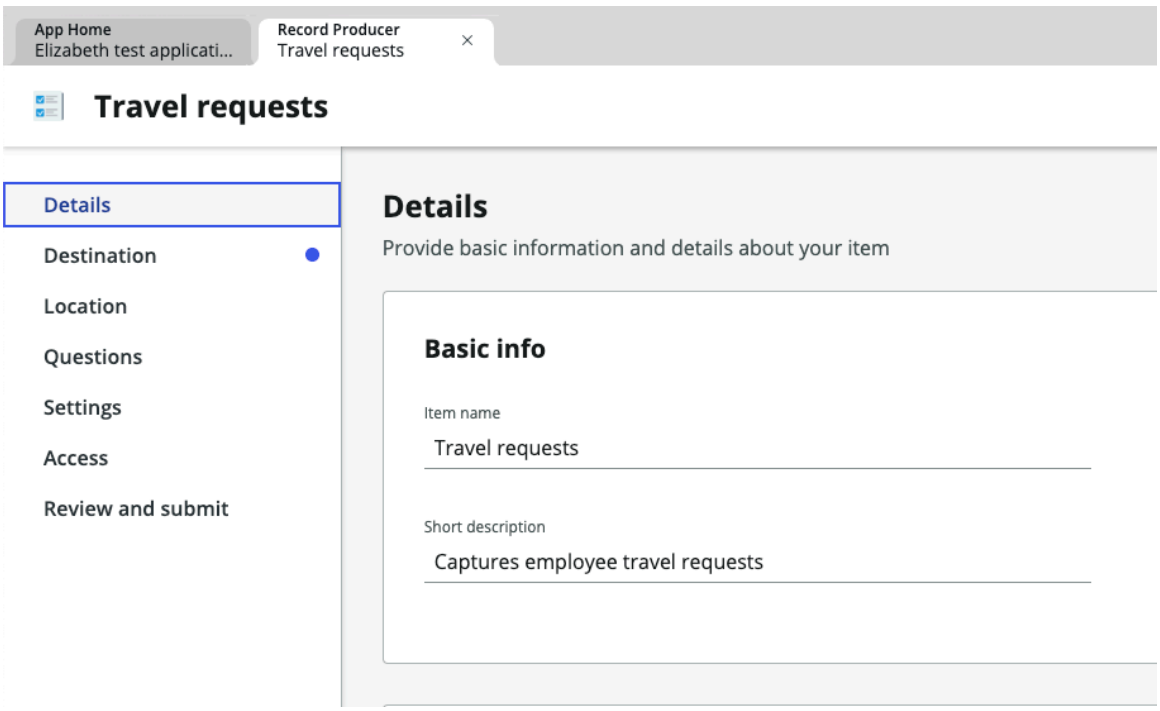
Video sections

Timestamp	Section
0:21	Specify a destination table.
0:32	Choose the location.
0:42	Create question containers.
1:01	Insert questions.
3:49	Preview the record producer.
3:58	Configure settings.
4:05	Select user access.
4:10	Review and submit the record producer.

Procedure

1. Ensure that the travel request record producer is open.
   - To open a record producer from your instance home page, navigate to **All** > *App Engine* > *App Engine Studio*. Then select your application and select the record producer that you want to configure.
   - To open a record producer that you have created, select **Edit record producer**.

   When your record producer is open, it appears in the canvas space.



2. **Optional:** Edit or add details about the record producer in the **Item name** or **Short description** fields.

3. Select **Continue to Destination** to specify the table in which the record producer creates records.
4. Enter `Travel request` in the **Record submission table** field, then select the employee travel request table from the list.
   By setting the travel request table as the destination table, travel requests submitted through the record producer become records in the employee travel request table.
5. Select **Continue to location** to specify the location of your record producer.

   Note:
   Location determines where users can find the record producer. You can locate your record producer in a catalog, such as the Service Catalog, which is a set of business and technical products, services, and offerings that users can order. You can also locate your record producer with a topic, such as a knowledge base article. For more information about locations for record producers, see Creating or editing catalog item template.

   In our tutorial, we locate our record producer in the Service Catalog, so that employees can access the employee travel request form easily.
6. In the **Catalogs** card, select **Browse**.
7. From the **Available options** list, select **Service Catalog** and use the move icon () to add it to the **Selected catalogs** list.
8. Select **Save selections**.
9. Select **Continue to questions** to start building the form questions.
10. Create containers, or sections, for the form questions.
    a. Select the arrow icon () in the **Insert new question** button to expand the menu of available question types.
    b. Select **Single column container** from the list.
    c. In the **Title** field, enter `General Information`.
    d. Select **Submit**.
    e. Repeat the procedure in steps a-d to create as many containers as needed.
    In our tutorial, we create another container for our form questions with the following specifications:

Container type	Title
Two column container	`Travel Details`

11. Add a question to the form.

    Note:
    You can build different types of questions for your record producer. To see the full list of available question types in Catalog Builder, see Create a question for a catalog item in Catalog Builder.

    In our tutorial, we create several questions to collect details about employees' travel requests.

    a. In the correct question container, hover over the plus icon (), then select **+ Insert**.
    In our tutorial, we insert this question in the General Information container.

b. Select **New question**.

c. Select and enter the following values for the new question.

Question type	Question subtype	Map to specific field on table	Table field	Question label	Name	Mandatory
Choice	**Dropdown (fixed values)**	Yes	**Reason for travel**	`What is the reason for travel?`	`reason_for_travel`	Yes

d. Select **Continue to Choices** to complete additional configurations for the question.

e. In the **Available choices** section, hover over the plus icon (), then select **+ Insert**.

f. In the **Display name** field, enter `Customer meeting`.

g. Repeat the process in steps e and f to add additional choices.
In our tutorial, we add two additional choices. The following list includes the names for each additional choice that we add:

- `Internal meeting`
- `Training`

h. Select **Insert question**.

12. Repeat the procedure outlined in step 11 to create as many questions as needed.
In our tutorial, we add the following additional questions to our form in the Travel Details container:

a. We add a question to determine the employee's departure date.

Question type	Question subtype	Map to specific field on table	Table field	Question label	Name	Mandatory	Additional configuratio
Date / Time	**Date**	Yes	**Departure date**	`When are you leaving?`	`departure_date`	Yes	None, select **Insert question**.

b. We add a question to determine the employee's return date.

Question type	Question subtype	Map to specific field on table	Table field	Question label	Name	Mandatory	Additional configuratio
Date / Time	**Date**	Yes	**Return date**	`When are you returning?`	`return_date`	Yes	None, select **Insert question**.

c. We add a question to determine where the employee is traveling from.

Question type	Question subtype	Map to specific field on table	Table field	Question label	Name	Mandatory	Additional configuratio
Choice	**Record reference**	Yes	**Travel from**	`Where are you traveling from?`	`travel_from`	Yes	Select **Source table** > **Airport**.

d. We add a question to determine where the employee is traveling to.

Question type	Question subtype	Map to specific field on table	Table field	Question label	Name	Mandatory	Additional configuratio
Choice	**Record reference**	Yes	**Travel to**	`Where are you traveling to?`	`travel_to`	Yes	Select **Source table** > **Airport**.

e. We add a question to determine the estimated cost of the travel.

Question type	Question subtype	Map to specific field on table	Table field	Question label	Name	Mandatory	Additional configuratio
Text	**Single line**	Yes	**Estimated airfare**	`What is your estimated airfare?`	`estimated_airfare`	Yes	Select **Text validation** > **Number**.

13. Optional: Preview your form by selecting **Preview**.

14. When you're ready to finalize your form, select either **Continue to settings** or the **Settings** tab in the side panel.
15. Select the check box next to any of the following portal settings.
    In our tutorial, we select the check box next to **Hide 'Add to wishlist' button** and **Hide attachment button**. You can select or deselect these settings in your record producer.
    - To prevent users from saving your record producer item to their wish list, select the check box next to **Hide 'Add to wishlist' button**.
    - To hide the component that enables users to provide attachments in the record producer form, select the check box next to **Hide attachment button**.
    - To require users to provide attachments, such as receipts or screenshots for service requests, select the check box next to **Make attachment mandatory**.
16. Select **Continue to access**.
17. **Optional:** Choose from the following options to define which users or groups have access to the form.

    Important:
    If you don't select any users or groups to grant or deny access, your form is available to everyone.

    In our tutorial, we don't grant or deny access to any users or groups.
    - To make the form available to specific users or groups, select **Browse** in the **Available for** card, then search for the users or groups that you want to grant access to. Move the users or groups from the **Available options** list to the **User criteria granted access** list using the move icon ( ).
    - To make the form unavailable for certain users or groups, select **Browse** in the **Unavailable for** card, then search for the users or groups that you want to deny access to. Move the users or groups from the **Available options** list to the **User criteria denied access** list using the move icon ( ).
18. Select **Continue to review and submit** to review and finalize your form.
19. Select **Submit**.

Result
Your record producer is configured and ready for testing.

What to do next
Continue building the employee travel request application by proceeding to the next step: Adding logic and automation.

Adding logic and automation

Add logic and automation to your application. Learn about logic and automation in App Engine Studio and how we automate the employee travel request application.

We continue building the employee travel request application by adding logic and automation.

This is the fifth step in the app creation tutorial. In this step, you learn about logic and automation in App Engine Studio. You build a decision table and create a flow for the employee travel request application.

Step 1	✓	Planning your application

Step 2		Create an app
Step 3		Building a data model
Step 4		Creating user experiences
Step 5		Adding logic and automation
Step 6		Test your application

Overview of logic and automation

Logic and automation enable you to define the actions that your application performs and the conditions that trigger those actions. Logic helps you define when certain actions should occur. For example, your application logic can determine whether support tickets are marked as "critical." Automation then defines the actions that the application performs. For example, if support tickets are marked as "critical", your application automation can move those tickets to the top of the queue.

There are several ways to implement logic and automation in App Engine Studio. Our tutorial focuses on decision tables and flows. To learn about the full range of logic and automation capabilities in App Engine Studio, see Add logic and automation.

Tutorial logic and automation

In our tutorial, we create a decision table and a flow to store the logic and automation for the employee travel request application. Decision tables enable you to set and manage business logic in your application separate from your code. The decision table in our tutorial determines which regional heads receive travel requests based on employees' geographic location.

Flows are automated processes that use a trigger, actions, and logic to define the behavior of an application. The flow in our tutorial defines the sequence of actions that the employee travel request application completes once a travel request is created.

We add logic and automation to our application in two phases:

Phase 1	Build a decision table
Phase 2	Create a flow

Next steps

Start adding logic and automation to the employee travel request application: Build a decision table.

Build a decision table

Build a decision table for the employee travel request application that determines which regional head to route travel requests to.

Before you begin
Role required: admin, decision_table_admin, or delegated developer permissions

About this task

The first phase in adding logic and automation to the employee travel request application is to build a decision table. Our decision table evaluates where an employee is located and uses that information to determine which regional head to route their travel request to.

Phase 1		Build a decision table
Phase 2		Create a flow

For more information about using decision tables, see Create decision tables in Workflow Studio.

Follow along with the tutorial to build a decision table for the employee travel request application.

https://player.vimeo.com/video/953653842?badge=0&autopause=0&player_id=0&app_id=58479

Video sections

Timestamp	Section
0:05	Create a decision table.
0:12	Create a new decision table.
0:13	Enter a unique name and choose the scope for the decision table.
0:28	Add an input.
0:36	Add a condition column.
0:52	Add conditions.
0:59	Add a result column.
1:09	Add results.
1:16	Add more conditions.
1:34	Save your changes.
1:36	Verify that the decision table has been added to your application.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. Select the travel request application.
3. On the application dashboard, select **+ Add** next to **Logic and automation**.
4. Select **Decision**, then select **Begin**.
5. Select **Create a new decision table**.
6. Define the properties of the decision table.

a. Enter `Regional head approvals` in the **Name** field.

b. In the **Accessible from** field, select **This application scope only** to have the logic in the decision table apply to only the selected application.
If you want the logic in your decision table to apply to all applications, select **All application scopes**.

c. Select **Continue**.

7. Select **Edit decision table** to continue setting up your decision table.
8. Add an input to the decision table.

Note:
Inputs define the data that is evaluated in the decision table.

In our tutorial, we set the input as the travel request table so that the decision table can evaluate employee information associate with each travel request.

a. Select **+ Add an input**.

b. In the **Type** field, select **Reference**, then select the travel request table from the list.

c. In the **Label** field, enter `Employee travel requests`.

d. **Optional:** Repeat this procedure to create as many inputs as you need for your decision table.
In our tutorial, we don't create additional inputs.

9. Add a condition column to the decision table.

Note:
Condition columns act as filters that refine the input data by specifying which values should be evaluated by the decision table.

In our tutorial, the condition column refines the input data to evaluate only the employees' country codes. Employees' country codes determine which regional head their travel requests should route to.

a. Select **+ Add condition column**.

b. In the **Condition column label**, enter `Employee regions`.

c. In the **Input** field, select the travel request table input.

d. For **Data to evaluate**, select **Field**.

e. In the **Field** field, select **Opened by** > **Country code**.

f. In the **Default operator** field, select **is one of**.

g. Select **Done**.

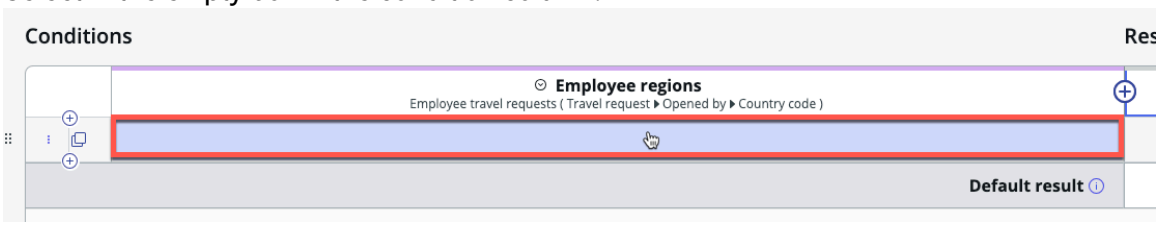
h. **Optional:** Repeat this procedure to create as many condition columns as you need for your decision table.
In our tutorial, we don't create additional condition columns.

10. Define individual conditions for the condition column.

> **Note:**
> Conditions represent the individual values that the decision table evaluates to return results.

In our tutorial, the conditions that we define represent each country that corresponds to a different regional head. We define individual conditions for the countries that make up the Americas (AMS) region, the Asia Pacific (APAC) region, and the Europe, Middle East, and Africa (EMEA) region.

a. Select in the empty box in the condition column.



b. Select **is**, then select **is one of** from the list.

c. In the empty space beneath **is one of**, select **Brazil** and **United States**.

d. Select **OK**.

e. Select **+ Add a new decision row** to add additional conditions to the condition column.

f. **Optional:** Repeat the process in steps a-e to define additional conditions.
In our tutorial, we define two additional conditions for the countries that make up the APAC and EMEA regions. The following table outlines the condition column values that we set:

Operator	Value
is one of	▪ **China** ▪ **Japan**
is one of	▪ **France** ▪ **Germany** ▪ **Italy** ▪ **Spain** ▪ **United Kingdom**

11. Add a result column to the decision table.

Note:
Result columns determine the type of data to return after evaluating the input and condition values.

In our tutorial, the result column determines which regional head to route travel requests to. As the regional heads are individual users in the system, the result column references the User [sys_user] table.

a. Select **+ Add result column**.

b. In the **Result column label** field, enter `Regional heads`.

c. Optional: In the **Description** field, enter a description for the result column.

d. In the **Type** field, enter `Reference`, then select **Reference** from the list.

e. In the **Reference table** field, enter `User`, then select **User [sys_user]** from the list.

f. Select **Done**.

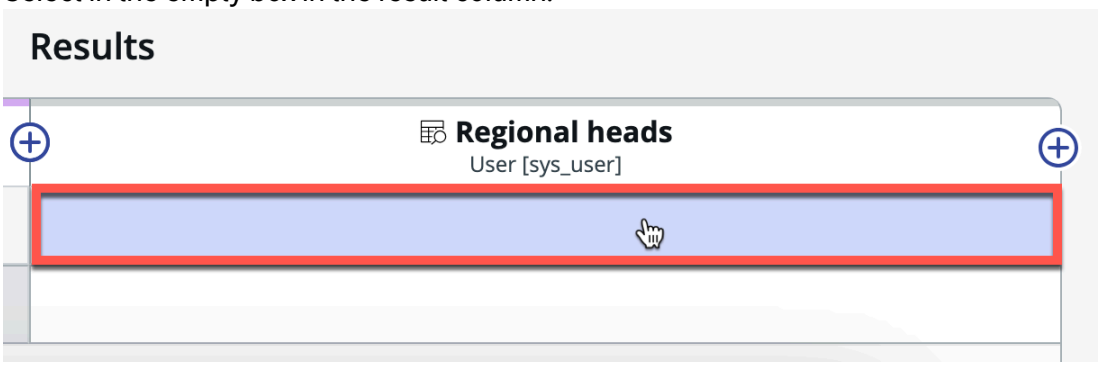
g. Optional: If you want to define additional results for your decision table, repeat this procedure to create as many result columns as needed.
In our tutorial, we don't create additional result columns.

12. Define the results for the result column.

Note:
Results are the individual values, or decisions, that are generated based on inputs, conditions, and results defined in the decision table.

In our tutorial, the results represent the regional heads for each region.

a. Select in the empty box in the result column.



b. In the empty field, enter the name of the regional head.
For the Americas (AMS) region, enter `Lucius Bagnoli`.

c. Select the regional head from the list, then select the Enter key.

d. Optional: Repeat the process in steps a-c to create as many results as needed for the decision table.

In our tutorial, we create two additional results for the APAC and EMEA regional heads. The following table outlines the regional head user information for each region:

Region	User
Asia Pacific (APAC)	`Rob Phil`
Europe, Middle East, and Africa (EMEA)	`Abel Tuter`

13. Select **Save**.
14. **Optional:** Verify that your decision table has been added to your application.
    a. Navigate to your application home page.
    b. Confirm that your decision table appears in the **Logic and automation** section.

Result
You have built a decision table for your application.

What to do next
Proceed to the next phase in adding logic and automation to the employee travel request application: Create a flow.

Create a flow

Automate the workflow for the employee travel request application by creating a flow.

Before you begin
Role required: admin or flow_designer

About this task
The second phase in adding logic and automation to the employee travel request application is to create a flow. The flow that we create automates the workflow for the application after an employee travel request is created.

Phase 1	✓	Build a decision table
Phase 2	○	Create a flow

For more information about flows, see Building flows.

Use the following tutorial to create a flow for the employee travel request application.

https://player.vimeo.com/video/953653961?badge=0&autopause=0&player_id=0&app_id=58479

Video sections

Timestamp	Section
0:09	Add a flow to your application.
0:14	Build the flow from scratch.

Video sections (continued)

Timestamp	Section
0:17	Enter a unique name and description for the flow.
0:22	Continue setting up your flow.
0:25	Create a trigger.
0:41	Add an Ask for Approval action.
1:09	Add If flow logic.
1:25	Integrate decision tables.
1:48	Duplicate repeating actions.
2:17	Add an Update Record action.
2:38	Add Else flow logic.
2:50	Add a Send Email action.
4:14	Add End Flow flow logic.
4:19	View your flow as a diagram.
4:22	Save your changes.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. Open the employee travel request application.
3. On the application dashboard, select **+ Add** next to **Logic and automation**.
4. Select **Flow**, then select **Begin**.
5. Select **Build your flow from scratch**.
   You can also choose to build your flow with Now Assist. For more information, see Create a flow with Now Assist.
6. Define the properties of the flow.
   a. In the **Name** field, enter `Approvals workflow`.
   b. In the **Description** field, enter `Approval workflow for managers and regional heads`.
   c. Select **Continue**.
7. Select **Edit this flow** to continue setting up your flow.
8. Add triggers to the flow.

> **Note:**
> A trigger specifies when to run the flow. When the trigger conditions are met, the system runs the flow using the data provided by the trigger.

In our tutorial, we create a trigger that runs the flow whenever an employee submits a travel request.

a. In the **Trigger** section, select **+ Add a trigger**.

b. In the **Trigger** field, select **Record** > **Created**.

For more information about trigger types, see Workflow Studio flow trigger types.

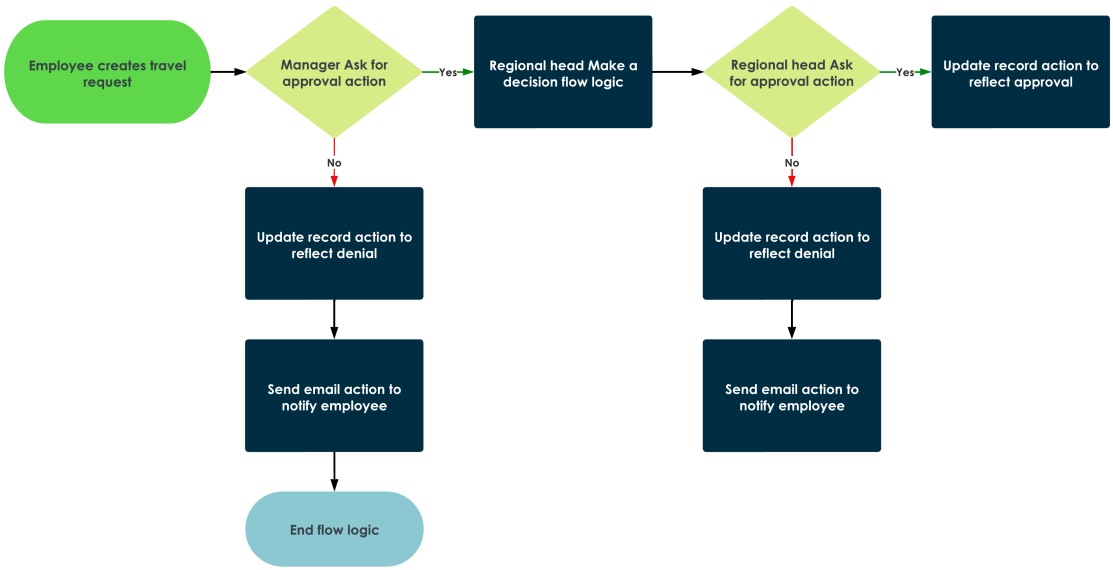
c. In the **Table** field, enter the name of the travel request table and select it from the list.

d. Select **Done**.

9. Add actions, flow logic, and sublows to the flow.

> **Note:**
> You can add actions, flow logic, and subflows to enable your flow to perform desired actions. For example, you can create an action that enables your application to send emails automatically.

In our tutorial, we add several actions and branches of flow logic to automate the employee travel request process. The following diagram demonstrates the flow that we create in our tutorial.



Instructions for adding the actions and flow logic demonstrated in the tutorial are outlined in the following topics.

10. **Optional:** See how your flow appears as a diagram by toggling the switch next to **View**.
11. Select **Save** to save your changes.

Result

You have created a flow that will automate the end-to-end workflow for the employee travel request application.

What to do next

Continue building the flow for the employee travel request application. Proceed to the next step: Add an Ask for Approval action.

Add an Ask for Approval action

Request approvals for any record by adding the Ask for Approval action to your flow.

Before you begin

Role required: admin or flow_designer

About this task

The Ask for Approval action is a ServiceNow Core action template that enables you to request approvals easily. You can specify which records require approval and assign users to approve or reject the record.

In our tutorial, we use the Ask for Approval action to request approvals from both managers and regional head for employee travel requests. This procedure outlines the Ask for Approval action that routes to the manager. The procedure for creating the Ask for Approval action that routes to the regional head is outlined in Duplicate repeating actions.

For more information about the Ask for Approval action, see Ask for Approval action.

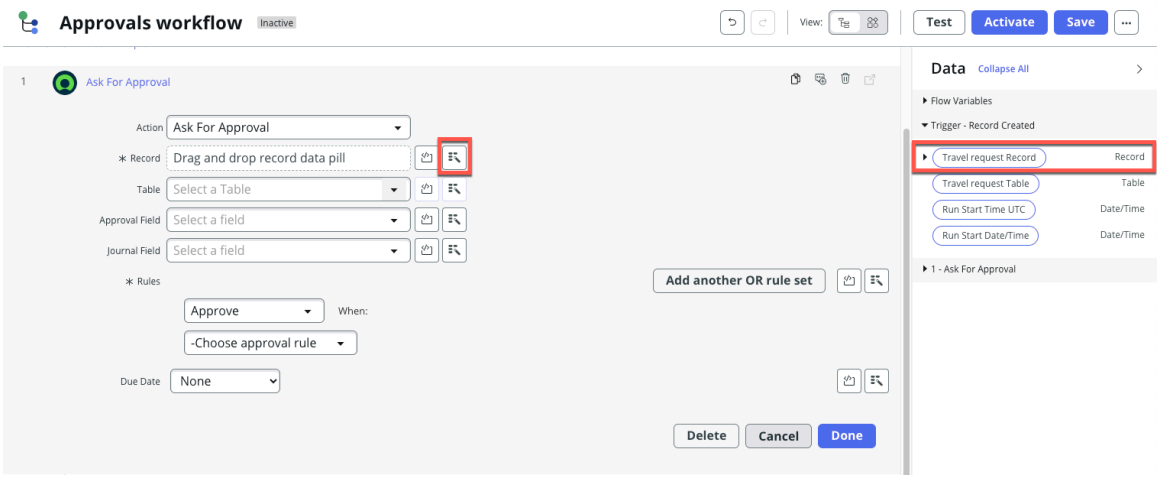
Procedure

1. At the correct location in your flow, select **+ Add an Action, Flow Logic, or Subflow**.
   In our tutorial, the Ask for Approval action that routes to the manager occurs just after the trigger. To see where the Ask for Approval action occurs in the tutorial flow, see the diagram in Create a flow.
2. Select **Action**.
3. From the **ServiceNow Core** options, select **Ask for Approval**.
4. Choose from one of the following options to connect the action to the appropriate trigger.

   Note:
   Some flows have multiple triggers, so you must specify which trigger corresponds to the action.

   In our tutorial, we add the travel request record trigger to the **Record** field, so that the action runs whenever an employee submits a travel request.

   - To add your trigger within the Ask for Approval dialog, select the **Data Pill Picker** (), then select the trigger from the list.
   - To add your trigger from the **Data** panel, drag the data pill that contains the trigger into the **Record** field.



5. Set the rules for the action.

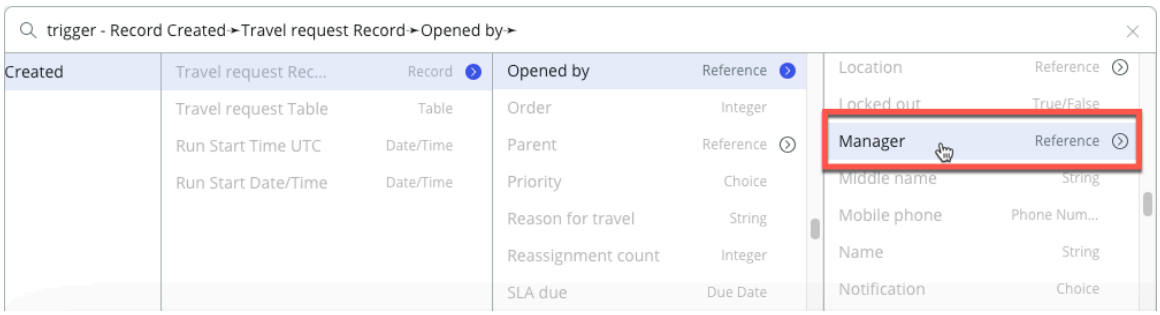
   Note:
   Rules determine which users can approve or reject requests, and what happens after approval or rejection. For more information about the rules available for the Ask for Approval action, see Ask for Approval action.

In our tutorial, we configure the rules for the Ask for Approval action to send travel requests to the manager of the user who submitted the request. The manager can then approve or deny the request.

a. Select the **Approve** field, then select **Approve or Reject** from the list.

b. Select the **-Choose approval rule** field, then select **All users approve or reject**.

c. Choose which users or groups can complete the approval actions using the **Data Pill Picker** ( ), the add user icon ( ), add group icon ( ), the add manual approvers icon ( ), or by dragging data pills from the **Data** panel into the field.
In our tutorial, we want the Ask for Approval action to route to the manager of the user who submitted the travel request. So we select **Trigger - Record Created** > **Travel request Record** > **Opened by** > **Manager**.



The second Ask for Approval action, which you create in Duplicate repeating actions, routes the approval to the regional head.

6. Select **Done**.

7. Select **Save**.

What to do next
Proceed to the next step: Add If flow logic.

Add If flow logic

Use If flow logic to direct your flow to perform actions based on specific conditions.

Before you begin
Role required: admin or flow_designer

About this task
Add If flow logic to your flow to have your application perform actions only when certain criteria are met. If flow logic creates a specific path, or branch, within a flow that runs only when the conditions are met.

In our tutorial, we use If flow logic to define what the application does if a travel request has been approved. For example, if a travel request has been approved by the regional head, the **State** of the travel request record should be changed to **Closed complete**.

There are two instances of If flow logic in the flow in the app creation tutorial: One that represents manager approval and another that represents regional head approval. This procedure covers creating the branch that represents manager approval. You can use this procedure to create the branch that represents the regional head approval and adjust the **Column label** and **Condition** field values accordingly.

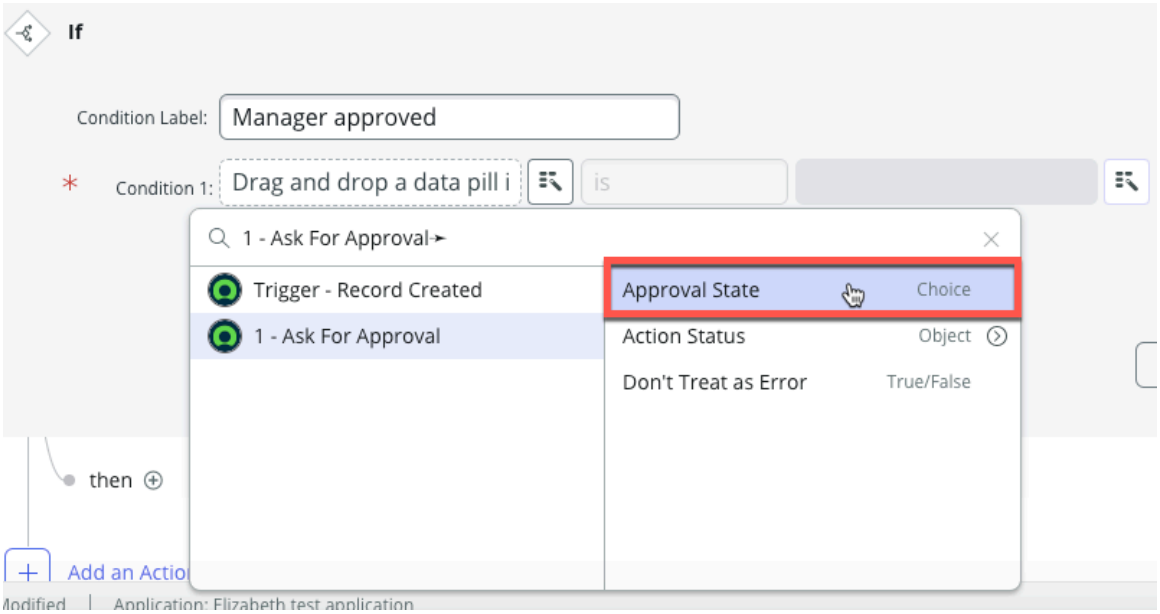
For more information about using If flow logic, see If flow logic.

Procedure

1. At the correct location in your flow, select **+ Add an Action, Flow Logic, or Subflow**.
   The If flow logic that represents manager approval occurs just after the first Ask for Approval action. To see where If flow logic occurs in the tutorial flow, see the diagram in Create a flow.
2. Select **Flow Logic**.
3. Select **If**.
4. **Optional:** In the **Condition label** field, enter a description to describe the conditions that the branch represents.
   In our tutorial, we enter `Manager Approved`, as this branch represents that a manager has approved the travel request.
5. Select the condition under which the branch runs using the **Data Pill Picker** ( ) or by dragging the data pill from the **Data** panel into the **Condition 1** field.

   Note:
   The condition is the data container that the flow evaluates when determining whether to run the branch. You can define one or multiple conditions in If flow logic.

   In our tutorial, we want this branch of the flow to represent what happens if a manager has approved a travel request, so we set the condition to **Approval State**.



6. Define the condition state that enables the branch to run by selecting the fields beside **Condition 1**, then choosing the appropriate values.

   Note:
   The condition state is the actual value of the condition that defines when the branch should run.

   In our case, this branch should run when a travel request has been approved by a manager. So we set the condition state to **is** and **Approved**.
7. Select **Done**.
8. Select **Save**.

Result
By default, If flow logic has no output. It simply evaluates whether a condition is met or unmet. To have your flow complete actions when the conditions in the If flow logic are met, you must add branching actions or flow logic.

What to do next
Proceed to the next step: Integrate decision tables.

Integrate decision tables

Integrate decision tables into your flow using Make a decision flow logic.

Before you begin
Role required: admin or flow_designer

About this task
Make a decision flow logic enables you to integrate decision tables within flows easily. Make a decision flow logic runs the decision table and returns results, which can be displayed as branches in the flow or data pills in the **Data** panel.

In our tutorial, employee travel requests are sent to the regional heads after receiving manager approval. We use Make a decision flow logic to connect the flow to the regional head decision table and retrieve the correct regional information.

For more information about Make a decision flow logic, see Make a decision flow logic.

Procedure

1. At the correct location in your flow, select **+ Add an Action, Flow Logic, or Subflow**.
   We add Make a decision flow logic beneath the branch of If flow logic that represents manager approval. To see where we add Make a decision flow logic in the tutorial flow, see the diagram in Create a flow.
2. Select **Flow logic**.
3. Select **Make a decision**.
4. Select the **Decision table** field, then select the correct decision table from the list.
   In our tutorial, we select the regional head approvals decision table.
5. Choose to select or deselect the check box for **Use Branches**.

   Note:
   If you select the check box for **Use Branches**, the flow displays each possible decision table result as a separate branch.

   In our tutorial, the approval process is the same for each of the regional heads, so we deselect **Use Branches**. If the employee travel request process was different for employees of a certain region, we could select **Use Branches** to represent the different processes based on region.
6. Add the correct trigger to the **Decision table inputs** field using either the **Data Pill Picker**( ) or by dragging the data pill from the **Data** panel into the field.

   Tip:
   When you integrate a decision table into a flow, you must specify the data values for the decision table to evaluate within the context of the flow.

   In our tutorial, we need the decision table to evaluate the user information that's associated with the travel request trigger. So we add our trigger to the **Decision table inputs** field.
7. Select **Done**.

Result
The Make a decision flow logic generates results from the decision table, which you can use in your flow.

What to do next
Proceed to the next step: Duplicate repeating actions.

Duplicate repeating actions

For actions that repeat within your flow, save time by duplicating the actions and adjusting their values as needed.

Before you begin
Role required: admin or flow_designer

About this task
Duplicating actions enables you to build flows more quickly and efficiently.

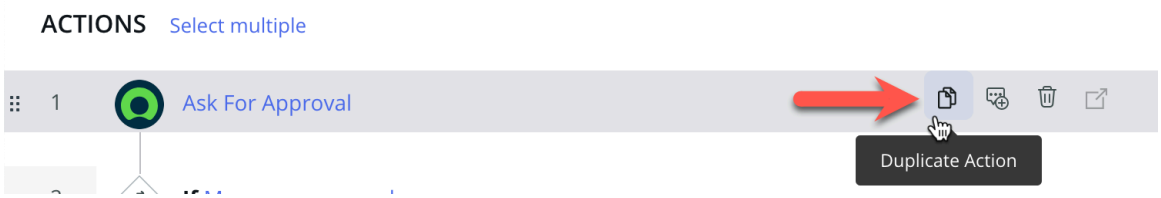
In our tutorial, there are several actions that repeat within the employee travel request flow.

- The Ask for Approval action repeats to request both the manager and regional head approvals for employee travel requests.
- The Update Record action updates the travel request record at several places in the flow, such as when the request is rejected by a manager.
- The Send Email action occurs in two places to notify employees that their travel requests have been rejected by either the manager or the regional head.

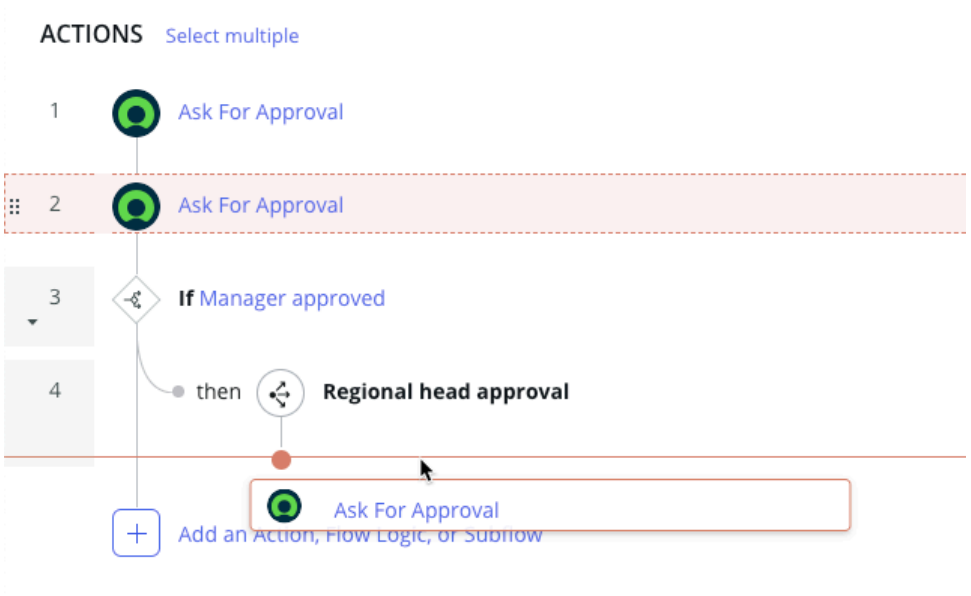
This procedure outlines duplicating the Ask for Approval action to request regional head approval. You can use the steps outlined here to duplicate the other actions that repeat and adjust the action location and field values accordingly.

Procedure

1. Hover over the action that you want to duplicate and select the Duplicate action icon ( ).
   In our tutorial, we duplicate the Ask for Approval action.



2. Move the duplicated action into the correct place in the flow by hovering over the action, selecting the drag icon ( ), and dragging the action into the correct place.
   We move the duplicated Ask for Approval action beneath the Make a decision flow logic that returns the regional head information.



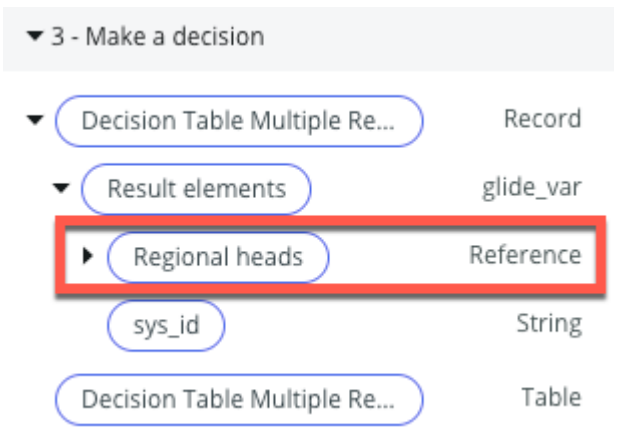
3. Adjust values in the action as needed.

   a. Select the duplicated action to open the action dialog.

   b. Select the remove icon ( ) to remove any field values that you must change.
   In our tutorial, we must change the user that the Ask for Approval routes to. So we select the remove icon ( ) in the data pill that contains the manager user information.

   c. Provide the correct field values by selecting the **Data Pill Picker** ( ) or by dragging data pills from the **Data** panel into the fields.
   To route the Ask for Approval action to the correct regional head, we must add the data pill that contains the regional head information.

   - Using the **Data Pill Picker** ( ), we select **Make a decision** > **Decision Table Multiple Result Record** > **Result elements** > **Regional heads**.
   - Using the **Data** panel, we select and drag the regional heads data pill into the field.



4. Select **Save**.

5. **Optional:** Repeat this procedure to duplicate as many actions as needed in your flow.
   In our tutorial, we also duplicate the Update Record and Send Email actions.

What to do next
Proceed to the next step: Add an Update Record action.

Add an Update Record action

Automate record updates in a flow using the Update Record action.

Before you begin
Role required: admin or flow_designer

About this task
The Update Record action is a ServiceNow Core action template that you can add to your flow to update existing records.

In our tutorial, we use the Update Record action to update the travel request record after it has been approved or rejected. This procedure outlines creating the Update Record action if a travel request has been approved by the manager and regional head. You can use this procedure to create the other Update Record actions and change the fields and field values as needed.

For more information about the Update Record action, see Update Record action.

Procedure

1. At the correct location in your flow, select **+ Add an Action, Flow Logic, or Subflow**.
   We add the Update Record action that represents manager and regional head approvals beneath the If flow logic branch for regional head approval. To see where we add the Update Record action in the tutorial flow, see the diagram in Create a flow.
2. Select **Action**.
3. From the **ServiceNow Core** options, select **Update Record**.
4. Select the record that you want to update by selecting the **Data Pill Picker** ( ) or by dragging the data pill from the **Data** panel into the **Record** field.
   In our tutorial, we want to update the travel request record, so we add the travel request trigger to the **Record** field.
5. Choose the field or fields that you want to update in the record by selecting **Add a field value**, then selecting the field that you want to update from the list.
   In our tutorial, we must update several fields in our travel request record after the request has been approved by the manager and regional head. We select the following fields:
   - **Approval**
   - **Assignment group**
   - **State**
6. Set the desired value of each field that you added by selecting anywhere in the **Select** field, then selecting the correct value from the list.
   In our tutorial, we set the following values for the fields that we added:

Field	Value
Approval	Approved
Assignment group	Travel desk
State	Closed complete

7. Select **Done**.
8. Select **Save**.

What to do next
Proceed to the next step: Add Else flow logic.

Add Else flow logic

In flows with If flow logic, add Else flow logic to create alternate paths when conditions aren't met.

Before you begin
Role required: admin or flow_designer

About this task
Add Else flow logic to define the actions that your flow performs when If flow logic conditions aren't met. Else flow logic creates a path in the flow that runs only when the If flow logic conditions aren't met.

In our tutorial, we add Else flow logic to specify what happens when a travel request is rejected by either a manager or regional head.

Procedure
1. At the correct location in your flow, select **+ Add an Action, Flow Logic, or Subflow**.
   We add Else flow logic at two places in the flow: One beneath the branch of the flow that represents manager approval and another beneath the branch that represents regional head approval. To see where we add Else flow logic in the tutorial flow, see the diagram in Create a flow.
2. Select **Flow logic**.
3. In the **Search flow logics** list, select **Else**.
4. Select **Save**.

Result
By default, Else flow logic has no output. It just creates a branch for the flow to run when the If flow logic conditions aren't met. To have your flow complete actions in the Else flow logic block, you must add branching actions or flow logic.

What to do next
Proceed to the next step: Add a Send Email action.

Add a Send Email action

Automate sending email notifications using the Send Email action.

Before you begin
Role required: admin

About this task
The Send Email action is a ServiceNow Core action template that you can add to your flow to automate sending emails to specified users or groups.

In our tutorial, we add Send Email actions to notify employees that their travel requests have been rejected. This procedure covers the Send Email action that notifies employees that their travel request has been rejected by the regional head. You can use this procedure to create the other Send Email action for the manager rejection and adjust the field values accordingly.

For more information about the Send Email action, see Send Email action.

Procedure

1. At the correct location in your flow, select **+ Add an Action, Flow Logic, or Subflow**.
   We add the Send Email action beneath the Else flow logic that represents regional head rejection. To see where we add Send Email actions in the tutorial flow, see the diagram in Create a flow.
2. Select **Action**.
3. From the **ServiceNow Core** options, select **Send Email**.
4. Add the record that the email is associated with to the **Target Record** field using the Data pill picker ( ), or by dragging the data pill from the **Data** panel.
   In our tutorial, we want the email to be associated with the travel request record in our trigger.
5. Add recipients for the email in the **To** field using the Data pill picker ( ), or by dragging the data pill from the **Data** panel.
   In our tutorial, we want the user who created the travel request to receive the email. The user information is stored under **Trigger - Record Created** > **Travel request Record** > **Opened by**.
6. **Optional:** Add additional recipients to the email in the **CC** and **BCC** fields.
   We don't add additional recipients to the Send Email action in our tutorial.
7. Enter a subject for the email in the **Subject** field.
   In our tutorial, we enter the following subject for the email: `Travel request rejected`.
8. Enter the content for the email message in the **Body** field.

   Tip:
   You can add dynamically created content, or content that is generated within the flow, to the Send Email action. To add dynamic content, drag the data pills from the **Data** panel into the **Body** of the email.

   In our tutorial, the body of our email notifies the employee that their travel request has been rejected and to contact their manager for additional information.
9. Select **Done**.
10. Select **Save**.

What to do next

Proceed to the next step: Add End Flow flow logic.

Add End Flow flow logic

Add End Flow flow logic to stop running the current flow.

Before you begin

Role required: admin

About this task

If you want your flow to end when certain conditions are met, you can add End Flow flow logic. End Flow flow logic can be contained within a conditional flow logic block, such as an If, Else If, or Else flow logic. End Flow flow logic stops running the current flow.

In our tutorial, we add End Flow flow logic to end the flow if a travel request has been rejected.

For more information about End Flow flow logic, see End Flow flow logic .

Procedure

1. At the correct location in your flow, select **+ Add an Action, Flow Logic, or Subflow**.
   We add End Flow flow logic in the branch that represents manager rejection of a travel request.
   To see where we add End Flow logic in the tutorial flow, see the diagram in Create a flow.
2. Select **Flow Logic**.
3. In the **Search flow logics** list, select **End Flow**.
4. Select **Save**.

What to do next

The logic and automation for the employee travel request application are complete. Proceed to the final step in the app creation tutorial: Test your application.

Test your application

Verify that your application works as intended before publishing by testing the application experience and flow.

Before you begin

Role required: admin or flow_designer

About this task

The final step in building an application is testing. There are several ways to test an application, including simulating the user experience and running flow executions to verify that everything works as expected.

This is the last step in the app creation tutorial. In this step, we create a test record in the record producer to verify that it triggers the flow to run. We also verify that approvals route to the correct users and that the flow performs the actions that we defined, such as updating records.

Step 1	✓	Planning your application
Step 2	✓	Create an app
Step 3	✓	Building a data model
Step 4	✓	Creating user experiences
Step 5	✓	Adding logic and automation
Step 6	○	Test your application

For more information about testing your application, see Testing and debugging applications.

Follow along with the tutorial to test the employee travel request application.

https://player.vimeo.com/video/953653889?badge=0&autopause=0&player_id=0&app_id=58479

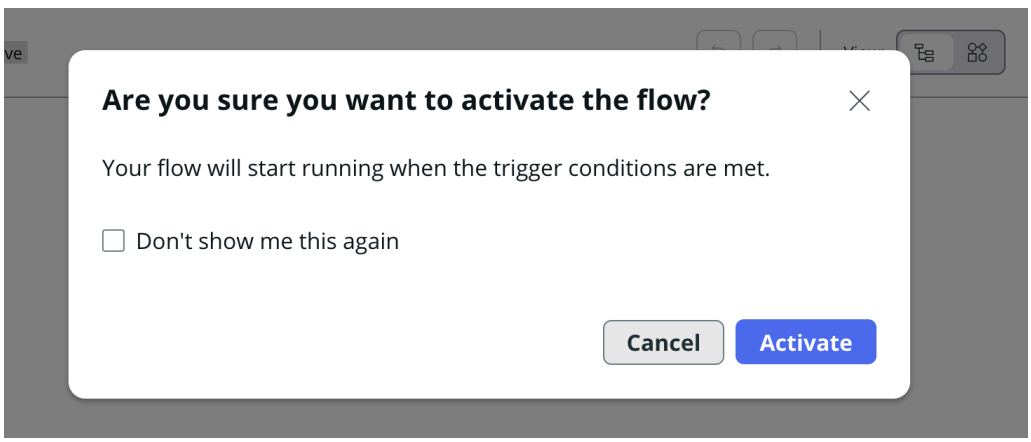
Video sections

Timestamp	Section
0:07	Activate your flow.
0:10	Navigate to your instance home page.
0:15	Impersonate a user to test your user experience.
0:20	Find and open your record producer.
0:26	Create a test record.
0:42	End the impersonation.
0:45	Navigate back to your flow.
0:48	Test your flow using the test record that you created.
0:56	Review the flow execution details.
1:00	Perform additional tests to verify that branching actions and flow logic run correctly.

Procedure

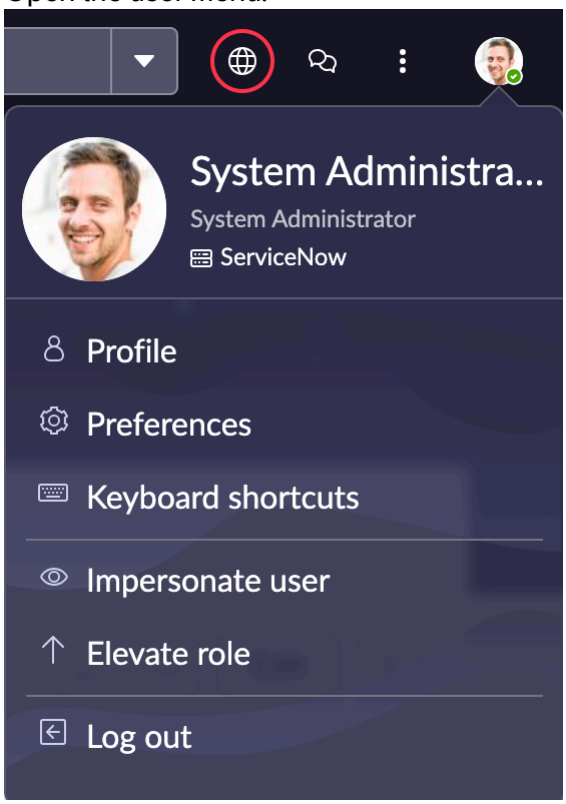
1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. Open your application's flow.
   - If your application isn't already open, navigate to **All** > **App Engine** > *App Engine Studio*. Then open your application and select the flow under the **Logic and automation** heading.
   - If your flow is already open, verify that your changes are saved.
3. Select **Activate**, then select **Activate** again to make your flow available to run.

> **Note:**
> You can also test your flow without activating it by creating a test record within Workflow Studio. For more information about testing before activating your flow, see Test a flow.



4. Navigate to your ServiceNow AI Platform instance home page.
5. Impersonate a user to test that your record producer functions as intended.

a. Open the user menu.



b. Select **Impersonate user**.

c. Select a user from the **Recent Impersonations** list or enter a different user's name in the user selection field.
When testing your application, you should impersonate users who have the permissions required to access the features that you're testing.
In our tutorial, we select a user with regular user permissions, as the record producer that we're testing is accessible to all users.

d. Select **Impersonate user**.
The system updates the user menu and dashboard to reflect the correct information for the user that you're impersonating.

6. On the home page, navigate to **All** > **Self Service** > **Service Catalog**.
7. Select **Services**.
8. Select your record producer from the list.
In our tutorial, we select our record producer called `Raise a travel request`.
9. Complete the record producer fields to generate a test record.
10. Select **Submit**.
11. End the impersonation.
    a. Open the user menu.
    b. Select **End impersonation**.
12. Reopen your application's flow.

a. Navigate to **All** > **App Engine** > *App Engine Studio*.

b. Open your application.

c. Select your flow under the **Logic and automation** heading.

13. Select **Test**.

14. In the **Test Flow** dialog, select the record field, then select the record that you created during the impersonation.

15. Select **Run Test**.
The system runs the flow based on the test record that you selected.

16. Select **Your test has finished running. View the flow execution details**.

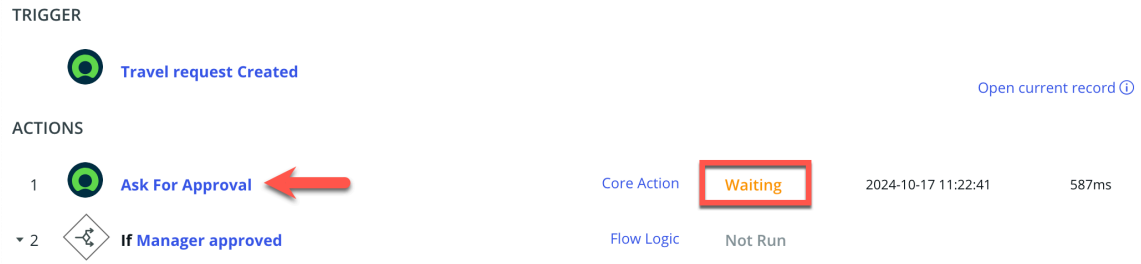
17. Review the flow execution details to verify that your flow runs as intended.
The flow execution details page enables you to view detailed information about an action or flow, such as the run-time, current state, items run, and values produced. For more information about flow execution details, see Flow execution details.

18. **Optional:** Perform additional tests to verify that the flow completes all desired branches of actions and flow logic.

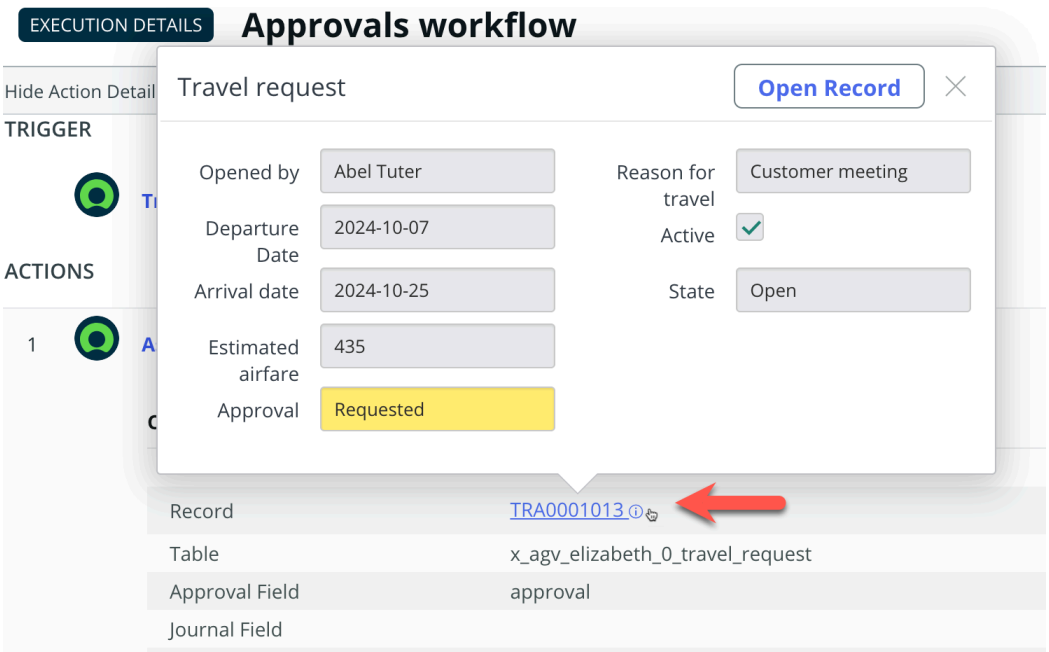
(Optional) For example, if your flow performs an action only after certain conditions are met, you can create those conditions to verify that the branching actions and logic run correctly.

In our tutorial, we simulate the manager's approval to test the branching actions and flow logic.

a. Select the action that is in the **Waiting** state to view the action details.



b. In the action details, select the test record.



c. Select **Open Record**.

d. Open the approval record by selecting the list item in the **Approvers** list.

> **Tip:**
> If your test record page doesn't display the **Approvers** list, you can adjust the page configurations to display any lists that you want to appear. Complete the following steps to configure the lists that appear on your record page.
>
> i. On the record page, select the Additional actions icon ( ).
>
> ii. Select **Configure** > **Related lists**.
>
> iii. Select the **Approvers** list item or any other lists that you want to appear on the record page and move them from the **Available** list to the **Selected** list.
>
> iv. Select **Save**.

e. On the approval record page, select **Approve** to simulate the manager's approval.

f. Navigate back to your flow.

g. Select the refresh icon ( ) to update your flow.

h. Verify that your flow has run branching actions or flow logic based on the additional test that you performed.

19. **Optional:** Complete this procedure to perform as many additional tests as needed to verify that all branches of your flow run correctly.

Result

Your have tested the employee travel request application and verified that it functions as expected.

What to do next

Congratulations! You have completed the app creation tutorial by planning, building, and testing an application.You are ready to start building your own applications in *App Engine Studio*.

To see what tasks you could perform after building an application, such as publishing or managing app deployments, see Publish your app.

Plan your app development

The application development process in App Engine Studio (AES) begins with your idea for the next effective app. Careful planning is the key to creating a successful app that is beneficial to your business.

Fit for the ServiceNow AI Platform

Consider the characteristics of your application to help determine whether it is suited for the ServiceNow AI Platform.

Good fit	Poor fit
• Simple forms • Task management • Request management • Spreadsheet-driven processes • Repeatable processes • Third-party integrations • Orchestration of multiple systems • Single experience from functions in multiple systems • Web and mobile access to the same apps and data simultaneously	• Unstructured data • Unrepeatable processes • Content that needs graphics processing • Streaming audio or video • Highly customized user interface

Plan before you build

In order to build your application, you need access to a ServiceNow instance and an admin or delegated developer role in that instance. If you have delegated developer permissions, you can develop applications but have fewer privileges than an admin.

Essentially, an application is a digital program that supports user tasks. As you plan your app, consider the questions in the following table.

Questions	Answers and considerations
What are the goals, objectives, and outputs of your application? What business problem are you trying to solve?	Without a clear business objective, you might have difficulty measuring the success of your application or justifying its continued use within the organization.
Are you turning a spreadsheet into a ServiceNow application or does the application exist somewhere else?	The answer to this question helps determine the tools you would use within the ServiceNow AI Platform to support your efforts.
Who will use your application?	Identifying your target audience has a direct impact on the features you include in your application, the data you choose to capture, and the interface you provide for the application.
Do you want all users to have the same ability to see and edit parts of your application or should some people have more or less access than others?	Identifying who has access to what information during the planning stage is a critical step in application development.
What is the main purpose of the application? Is the application used to provide information, collect information, route information, look up information, request something, or collaborate on information?	Identifying the application's purpose helps to establish the features and functions you need to build into your application.

Questions	Answers and considerations
Will your application require entering information or does the data in your application exist in your ServiceNow instance (that is, user data)? Will data be imported from an external source?	Use available data sources as much as possible to avoid duplication of data and to ensure your application has the data it needs to meet business objectives.
How will users interact with your application? Will users access your application using their computer or mobile device?	Understanding how users access your application affects how you build your application's functionality.
How will you report on your application to stakeholders?	If your application is meeting a business purpose, you might need to provide reports showing usage, adoption, and key business objectives related to your application.
How will you maintain this application?	Planning for the maintenance of your application at the start can prevent roadblocks further into the development process.

The answers to these questions can help you decide whether you want to base your application on a predefined template or create your app from scratch.

Irreversible considerations

Some actions you take when building an application might be irreversible. Be aware of these actions and plan for them in advance.

Action	Considerations
Creating a scoped or global application	When you create an application, you can choose to create it in a private scope (scoped application) or in a global scope (global application). Scoped applications have extra functionality for managing development, application deployment, and data security. By default, all applications are created in a private scope. Citizen developers should generally work with scoped applications.
Using different instances	Proof of concept (PoC) applications can be built in a personal developer instance (PDI) obtained from the ServiceNow Developer site. • PDI names are similar to dev12345.service-now.com. • PoC applications can be rebuilt in your PDI. Don't import the PoC app into your PDI. Production applications that your organization uses should be created in a development instance so they can be subject to your organization's testing and deployment process. Ask your ServiceNow administrator which instance to use for developing

Action	Considerations
	applications that will eventually be deployed to your organization's production instance.
Selecting an application name	Based on your application's display name, auto-complete displays suggestions for an internal name also known as the application scope. Application scopes are written in the form x_[company code]_[app_name], for example, x_acme_legal_request. Every property created in your application inherits the application scope name so be sure to choose it carefully before starting to develop your application. **Note:** You can always change the application's display name but the application scope name can't be changed.
Selecting table and field names	After your application is created, begin creating tables and fields. Tables and fields have internal database names that should be edited only at creation time.

Application Intake process

Any member of your organization can submit ideas for app development through a process called Application Intake. After the idea has been submitted and approved, developers can begin building the app with a guided, intuitive app development environment in App Engine Studio.

The Application Intake request process benefits your organization as follows:

- Simplifies the process to request, approve, track, and store app ideas.
- Improves planning by providing visibility into all app ideas.
- Provides an efficient process to filter out repetitive requests.

For more information, see Submit your idea for app development.

- The intake request form is available through the standard Service Catalog. For information about how you can modify this form, see Service catalog items.
- The intake request process uses Flows in Workflow Studio to determine the approval flow. To adjust the approval flow, see Flows.
- Admins review intake application requests in App Engine Management Center . For more information, see Managing app development using the App Engine Management Center.

Application templates

An application template provides predefined content to support a certain purpose. For example, the Travel Request template provides application content for submitting and approving employee travel requests. Choose the template that most closely fits your application goals.

If the available templates don't fit your application goals, you can create your app from scratch and control all aspects of the development process.

Additional resources for planning your app development

Learn more about planning app development	ServiceNow resources
ServiceNow provides additional resources on planning your app development	Launching an effective Citizen Development program

Create your app

Use a template in App Engine Studio (AES) to build data, experience, logic and automation, and security into your application automatically. Or, create your app from scratch and customize it as needed.

Note:
Before creating an app, Plan your app development.

To build an application in App Engine Studio, add the following content:

Data

Information that is stored in your application. For example, employee phone numbers or office locations. You configure application data using tables.

Experience

Graphical interface that your users interact with. For example, you can create a portal where users find information, submit requests, or complete business tasks.

Logic and automation

Automate all the work in your application by adding logic and automation. For example, you can build a flow that sends a notification to the admin when someone makes a request.

Security

Roles and access controls to limit who can use your application.

Note:
To create apps in AES, you must be an admin or in the App Engine Studio Users group. If you are in the App Engine Studio User Limited group, you can only edit existing apps, not create new ones.

If you use an application template, the template automatically adds one or more of these items to your application. You can then edit them to tailor to your specific business needs. You can also add more data, experience, logic and automation, or security to an application that you created using a template.

To build an application from scratch, add data, experience, logic and automation, and security to your application and then edit each.

Now Assist for app generation in AES

Note:
If you using a Xanadu Store Release 2 instance, Now Assist for app generation works with ServiceNow Studio (not App Engine Studio). For detailed information, see Now Assist for app generation in ServiceNow Studio.

Get started

Important:

- Not all model providers are available for customers with in-country SKUs, and some Now Assist products/features are currently unavailable for in-country customers. For more information, see the KB1584492 article in the Now Support Knowledge Base. Be sure to check for model provider availability updates in future releases.
- Some Now Assist products/features are currently unavailable for customers in the FedRAMP, NSC DOD IL5, or Australia IRAP-Protected data centers, self-hosted customers, or in other restricted environments. For more information, see the KB0743854 article in the Now Support Knowledge Base . Be sure to check for availability updates in future releases.
- Some Now Assist products/features are currently available only for customers in some regions. Be sure to check for availability updates in future releases.
- Some AI products and skills are not available in Regulated Markets. For more information, see KB2593939: Regulated Markets AI Products/Skills Not Available. Be sure to check for availability updates in future releases.

Troubleshoot and get help

AI limitations

This application uses artificial intelligence (AI) and machine learning, which are rapidly evolving fields of study that generate predictions based on patterns in data. As a result, this application may not always produce accurate, complete, or appropriate information. Further, there is no guarantee that this application has been fully trained or tested for your use case. To mitigate these issues, it is your responsibility to test and evaluate your use of this application for accuracy, harm, and appropriateness for your use case, employ human oversight of output, and refrain from relying solely on AI-generated outputs for decision-making purposes. This is especially important if you choose to deploy this application in areas with consequential impacts such as healthcare, finance, legal, employment, security, or infrastructure. You agree to abide by ServiceNow's AI Acceptable Use Policy, which may be updated by ServiceNow.

Data processing

This application requires data to be transferred from ServiceNow customers' individual instances to a centralized ServiceNow environment, which may be located in a different data center region from the one where your instance is, and potentially to a third-party cloud provider, such as Microsoft Azure. This data is handled per ServiceNow's internal policies and procedures, including our policies available through our CORE Compliance Portal.

Data collection

ServiceNow collects and uses the inputs, outputs, and edits to outputs of this application to develop and improve ServiceNow technologies including ServiceNow models and AI products. In addition, this application will collect information about applications (and associated application files) in which App generation was utilized. Customers can opt out of future data collection at any time, as described in the Now Assist Opt-Out page.

Exploring app generation in AES

Note:
If you using a Xanadu Store Release 2 instance, Now Assist for app generation works with ServiceNow Studio (not App Engine Studio). For detailed information, see Now Assist for app generation in ServiceNow Studio.

App generation overview

For more information about using app generation, see Generate apps with Now Assist for Creator for use with AES.

App generation benefits

Benefit	Feature	Users
Quickly reach a starting point and establish a base for further development.	Generate apps with Now Assist for Creator for use with AES	Developer

General guidelines for using app generation in AES

Learn about general guidelines for using the app generation skill.

Note:
If you using a Xanadu Store Release 2 instance, Now Assist for app generation works with ServiceNow Studio (not App Engine Studio). For detailed information, see Now Assist for app generation in ServiceNow Studio.

Configuring app generation in AES

Note:
If you using a Xanadu Store Release 2 instance, Now Assist for app generation works with ServiceNow Studio (not App Engine Studio). For detailed information, see Now Assist for app generation in ServiceNow Studio.

Install Now Assist for app generation to use with AES

Before you begin

Review the Now Assist for Creator application listing in the ServiceNow Store to learn about dependencies, licensing, and subscription requirements, and release compatibility. Now Assist for Creator installs the Now Assist for app generation application.

Note:
If you using a Xanadu Store Release 2 instance, Now Assist for app generation works with ServiceNow Studio (not App Engine Studio). For detailed information, see Now Assist for app generation in ServiceNow Studio.

Role required: admin

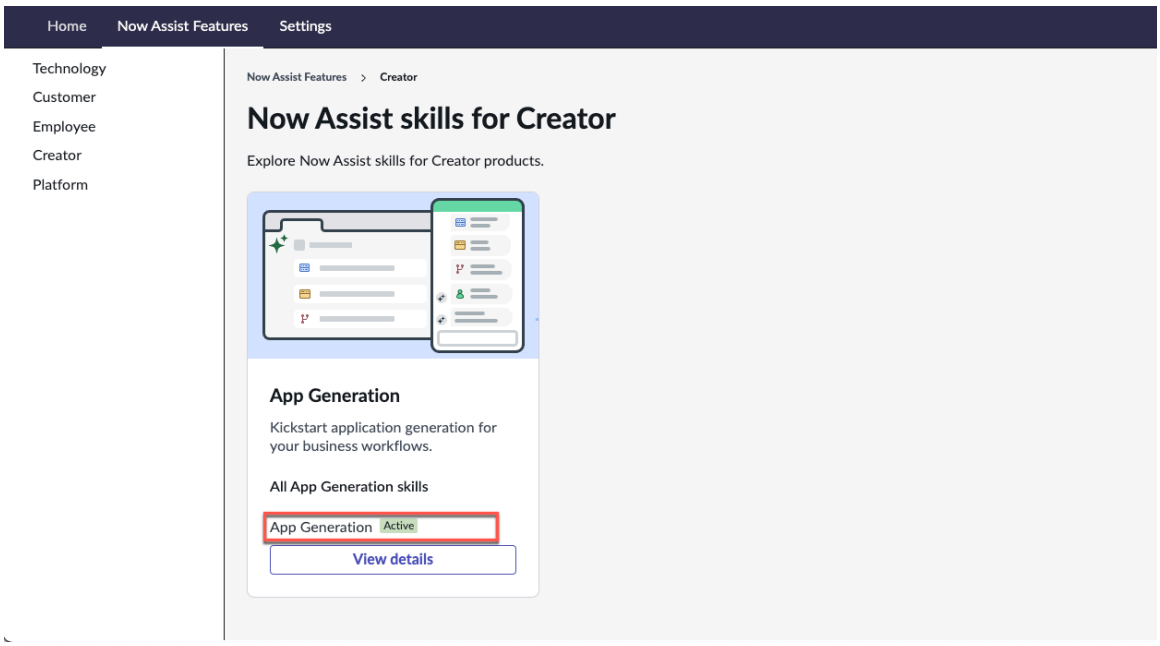
Procedure

1. Navigate to the Now Assist for Creator application on the ServiceNow Store .
2. On the Now Assist for Creator application page, select **Request App**.
3. After your request is approved, navigate to **All** > **System Applications** > **All Available Applications** > **All**.
4. Find the Now Assist for Creator application (sn_now_creator) by using the filter criteria and search bar.
5. Select **Install**.
6. Verify that Now Assist for Creator is installed.

a. Navigate to **All** > *Now Assist Admin* > **Features**.

b. In the workflow list, select **Creator**.

c. On the App Generation card, verify that the app generation skill is active.



For more information about using the Now Assist Admin console to access information about setting up, configuring, and monitoring Now Assist applications, see Now Assist Admin console.

What to do next

Grant the admin and now.assist.creator roles to each user that you want to create and edit applications using app generation.

To begin generating applications, see Generate apps with Now Assist for Creator for use with AES.

Related topics

Install Now Assist for Creator

Generate apps with Now Assist for Creator for use with AES

Have a conversation with the Now Assist for Creator application to start building applications.

Note:
If you using a Xanadu Store Release 2 instance, Now Assist for app generation works with ServiceNow Studio (not App Engine Studio). For detailed information, see Now Assist for app generation in ServiceNow Studio.

This video shows you how to perform the following procedure in App Engine Studio.

https://player.vimeo.com/video/996001913?h=35af70a157&badge=0&autopause=0&player_id=0&app_id=58479

Before you begin

To use app generation, you must activate the skill in Now Assist for Creator. For more information, see Install Now Assist for app generation to use with AES.

Role required: admin and now.assist.creator

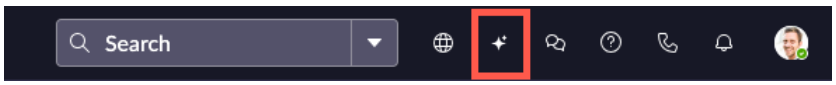
Note:
As of Xanadu Store Release 2, if you have users that only need to edit (not create) apps, they can be assigned the delegated_developer, now_assist_panel_user, and now.assist.creator roles.

About this task

When app generation is enabled on an instance, the Now Assist icon ( ) appears on the home page banner.

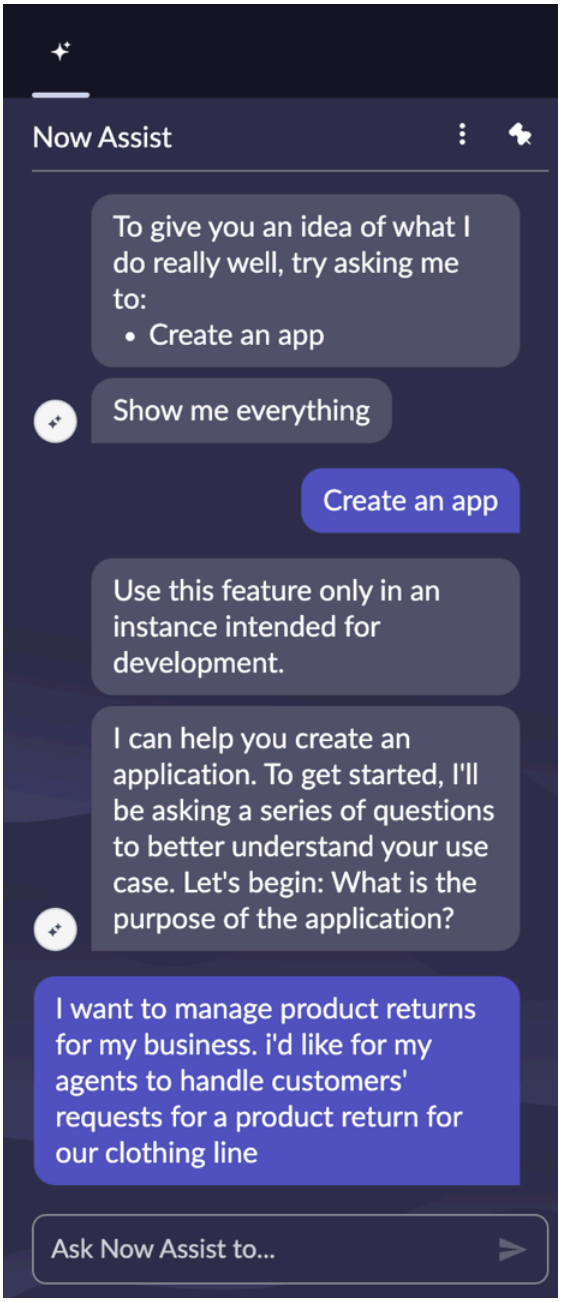
Procedure

1. Open the Now Assist panel by selecting the Now Assist icon.



2. In the Now Assist panel, select **Create an app**.
   Now Assist asks you to describe your application.
3. Summarize your current business process.

   Describe your application's use case so Now Assist understands the purpose of your application. You can be highly detailed if you have a clear idea of what you want. If you don't, start with a broad description. Now Assist narrows down your app's requirements.



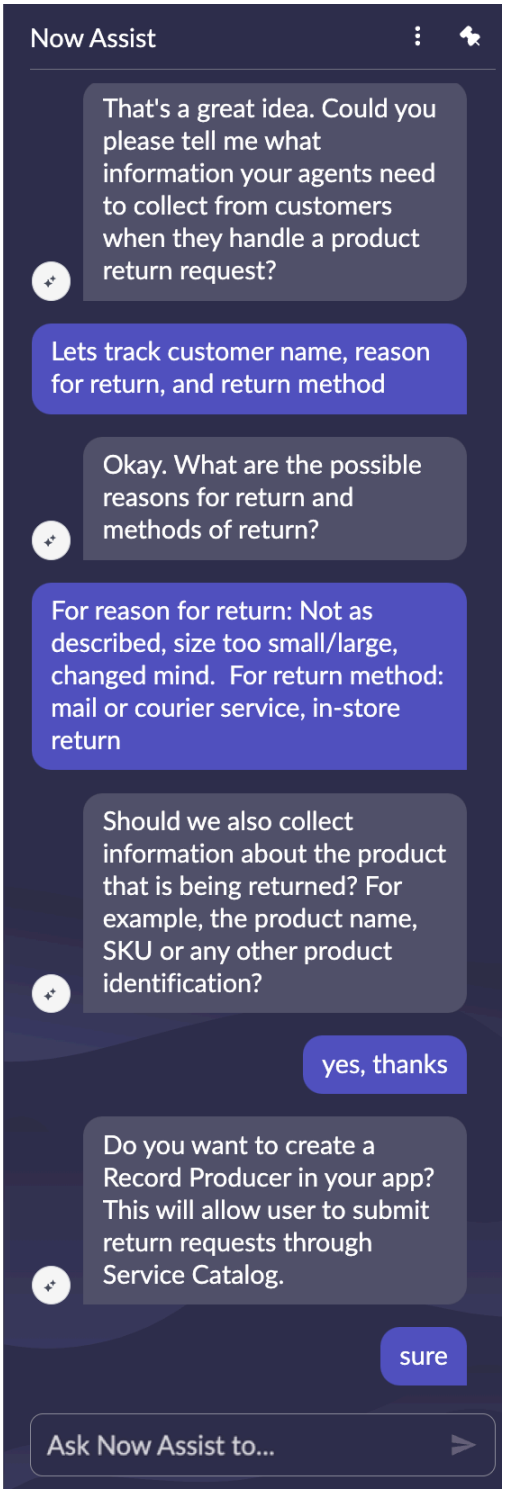
4. Continue the conversation with Now Assist by providing details about how you want the app to work.

   Example

   Now Assist asks you questions to understand the data to be collected, the users involved and their permissions, and the desired interface. Your answers help Now Assist create the correct tables, roles, access control lists (ACLs), forms, and record producers for your application.

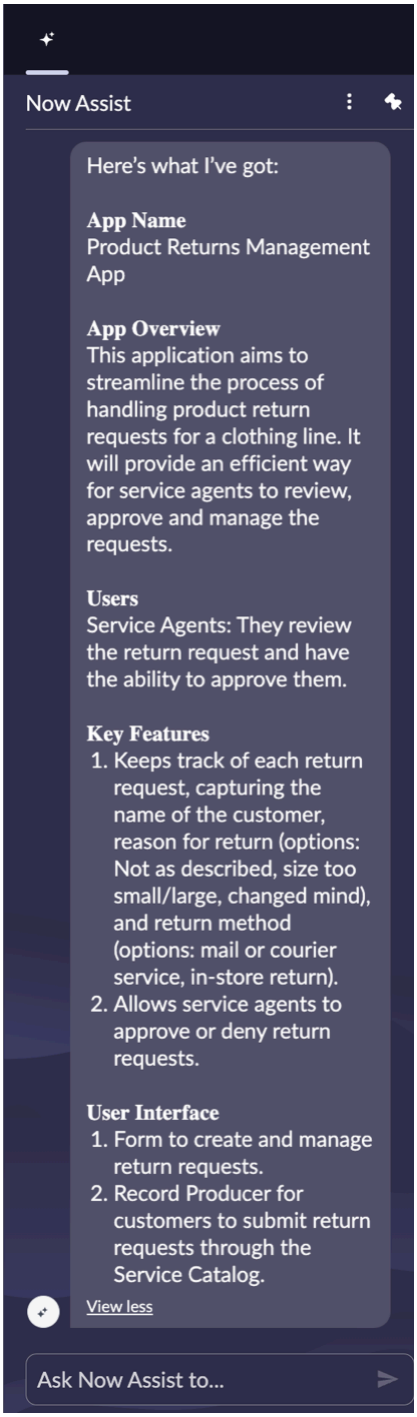
   If you know exactly how you want your app to work, be specific about its functionality. If you don't, interact with Now Assist to determine the correct application requirements. For more information, see General guidelines for using app generation in AES.

   When Now Assist collects enough information, it provides a detailed summary of the application requirements.



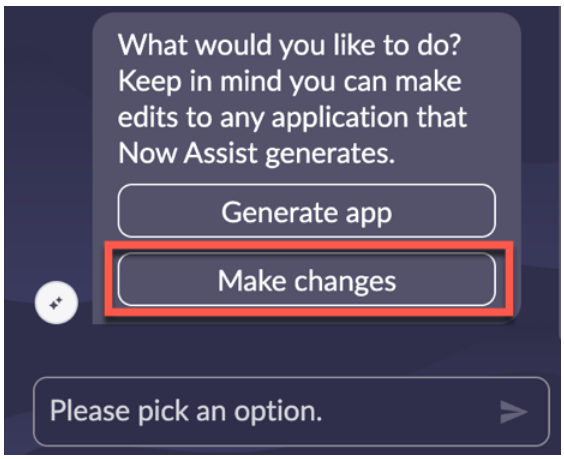
5. Review the summary of your application requirements.

Note:
Because the information is generated by AI, review the text to make sure it's accurate.



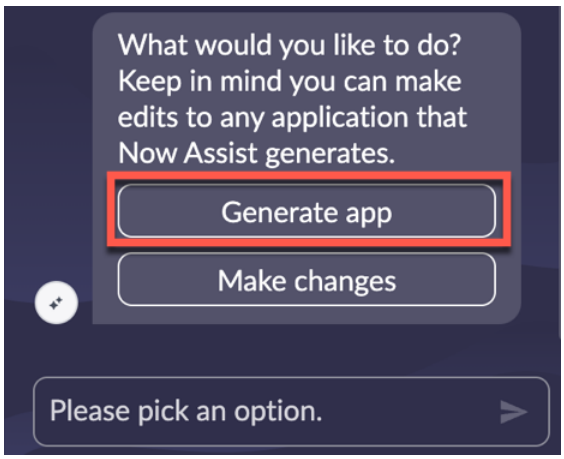
6. Finalize and generate the app.

a. If necessary, select **Make changes** to keep the Now Assist panel open so that you can keep editing.

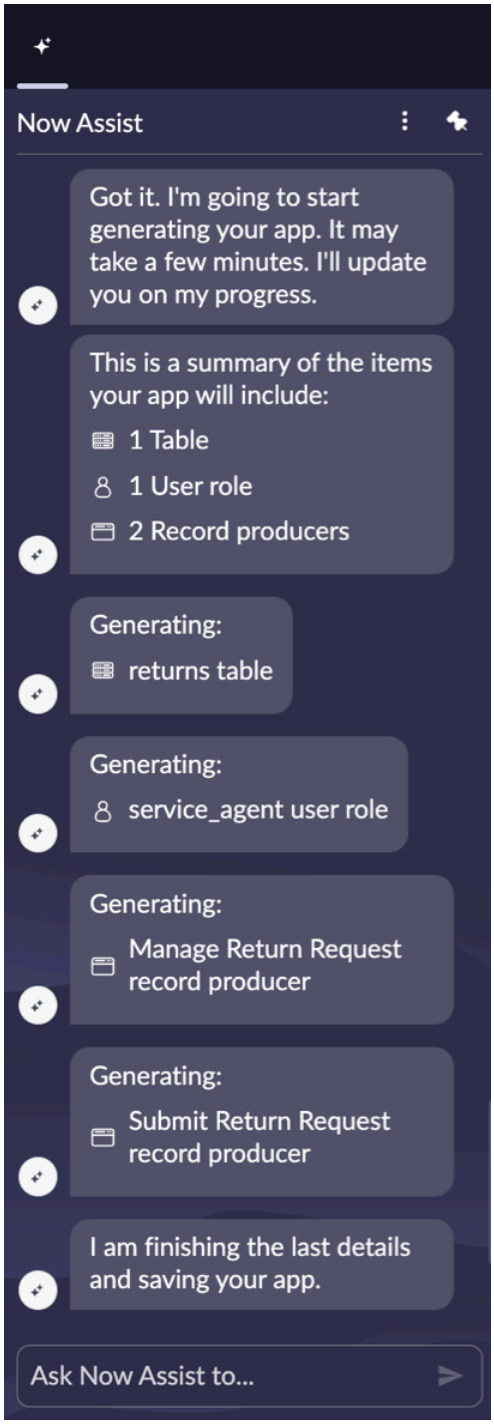


Continue to interact with Now Assist until the summary it produces matches your app's requirements.

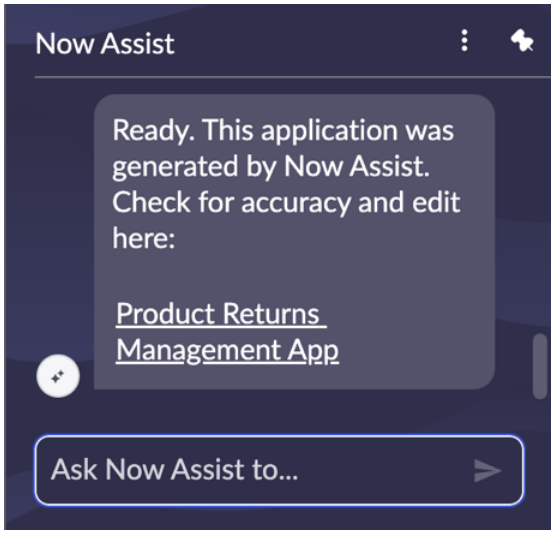
b. After you're satisfied with the application requirements, select **Generate app**.



Now Assist tells you what it's generating.



Now Assist provides a link to open the app and its supporting components in App Engine Studio:



For more information about App Engine Studio, see Build apps using App Engine Studio.

Result
Now that your app is created, review its contents and verify for accuracy. Then, use the tools in App Engine Studio to add more features and enhance your app further.

Review the apps generated by Now Assist for Creator in App Engine Studio

In Xanadu Store Release 1, use the App Engine Studio development environment to verify and modify the application that you created with Now Assist for Creator.

> **Note:**
> If you using a Xanadu Store Release 2 instance, Now Assist for app generation works with ServiceNow Studio (not App Engine Studio). For detailed information, see Now Assist for app generation in ServiceNow Studio.

This video shows you how to perform the following procedure in App Engine Studio.

https://player.vimeo.com/video/995459091?badge=0&autopause=0&player_id=0&app_id=58479

Before you begin
Check that you have an application created using Now Assist for Creator.

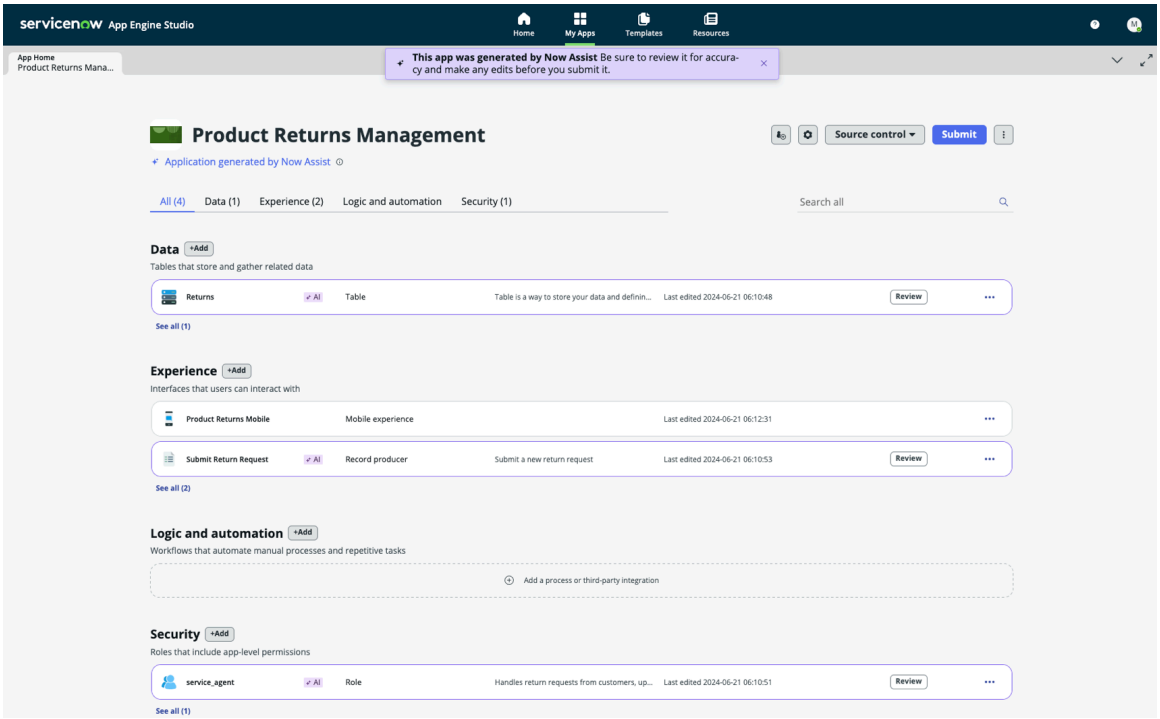
Role required: admin and now.assist.creator (as of Xanadu Store Release 2, if you have users that only need to edit, not create, apps, they can be assigned the delegated_developer, now_assist_panel_user, and now.assist.creator roles)

About this task
The application generated by Now Assist for Creator provides a starting point for further development. Examine the contents of the application and confirm that they align with your business process. Enhance the application by adding functionality as needed in App Engine Studio.

In Xanadu Store Release 1, apps generated with Now Assist automatically open in App Engine Studio.

Procedure

1. If the application is not already open, navigate to App Engine > App Engine Studio and select the application tile.



   Notice the banner prompt to review and that the line items generated by Now Assist have the following:

   - AI indicator AI
   - **Review** button

   In contrast, observe that in the **Experience** section, the **Product Returns Mobile** experience was created without Now Assist and does not include these elements.

   To learn more about the AES home page, see AES user interface.

2. Review the generated tables.

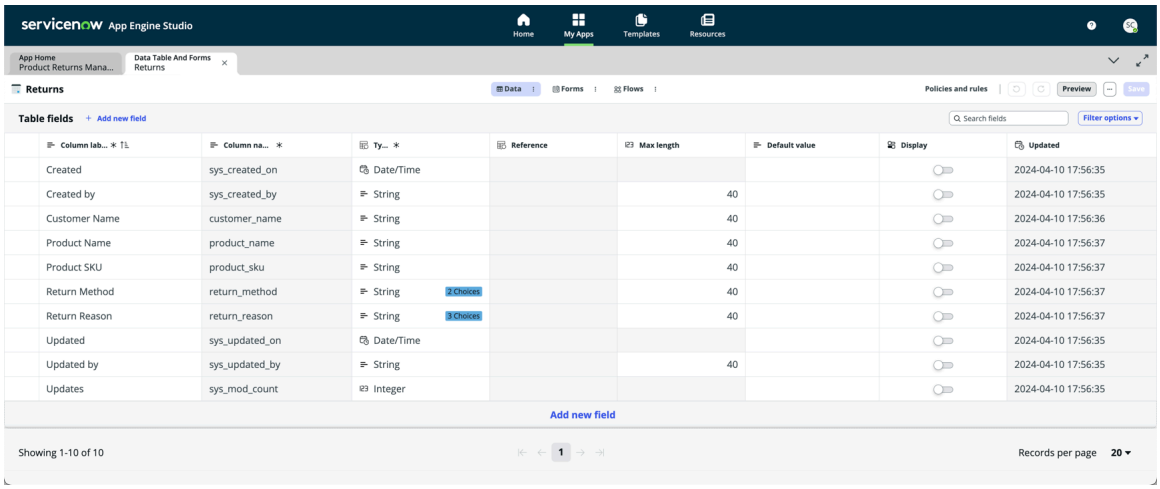
   a. Under the **Data** heading, select the Additional actions icon (•••) for an existing table in your application, and then select **Edit**.
   When you select a data table or select **Edit** to edit data for your application, Table Builder launches. Table Builder is a tool for editing the data tables that you added to your application.

   b. Check the table data for accuracy and completeness.

   On the **Data** tab:

   - Manage table fields.
   - Edit the table and field column properties.
   - Verify that the generated choices are implemented correctly.

Now Assist created tables based on your instructions. Confirm that the tables capture the desired data to be collected. For example, the following table includes the fields discussed during the conversation.



For more information on editing tables in Table Builder, see Modify application data tables.

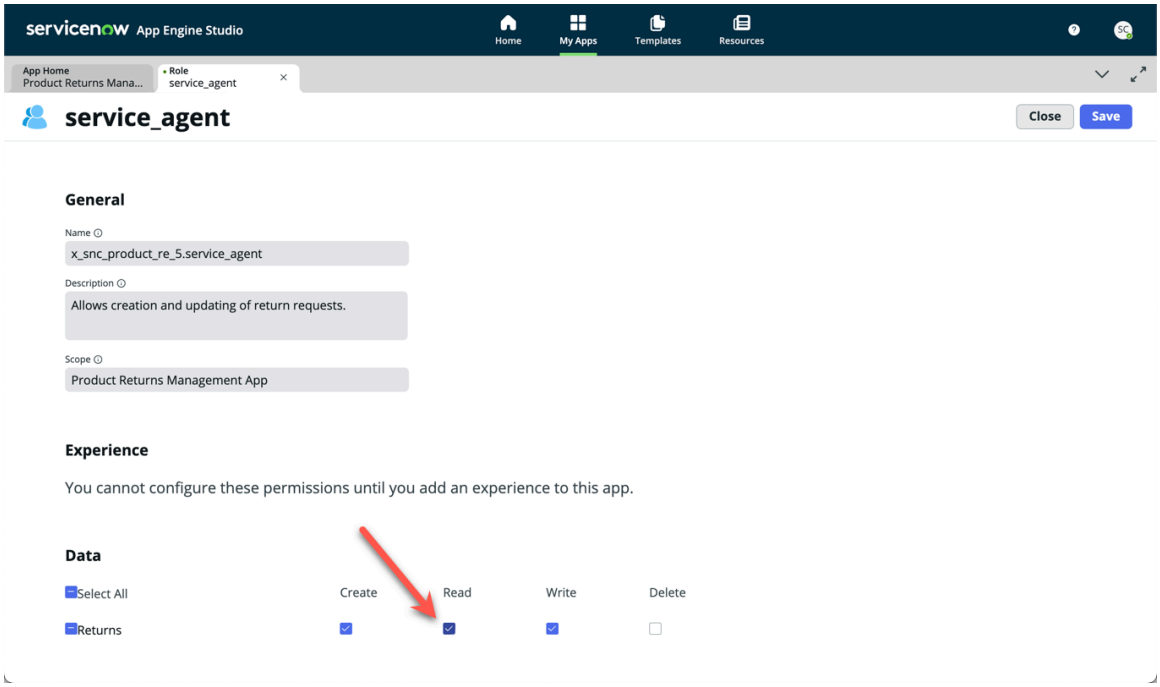
3. Select the **App Home** tab to review the generated roles and access control lists (ACLs).

   a. Under the **Security** heading, select the Additional actions icon (•••) next to a role, and then select **Edit**.

      You can also select the bar that lists the role you want to view.

   b. Verify that the created roles and their assigned permissions accurately represent the users of your application.

      You can configure the roles and their permissions to align with your workflow. For example, the role in the following example was granted permissions to create and update return requests. You want this role to read the requests, so you would configure that access on the Role page.



For more information on configuring roles and access, see Working with roles.

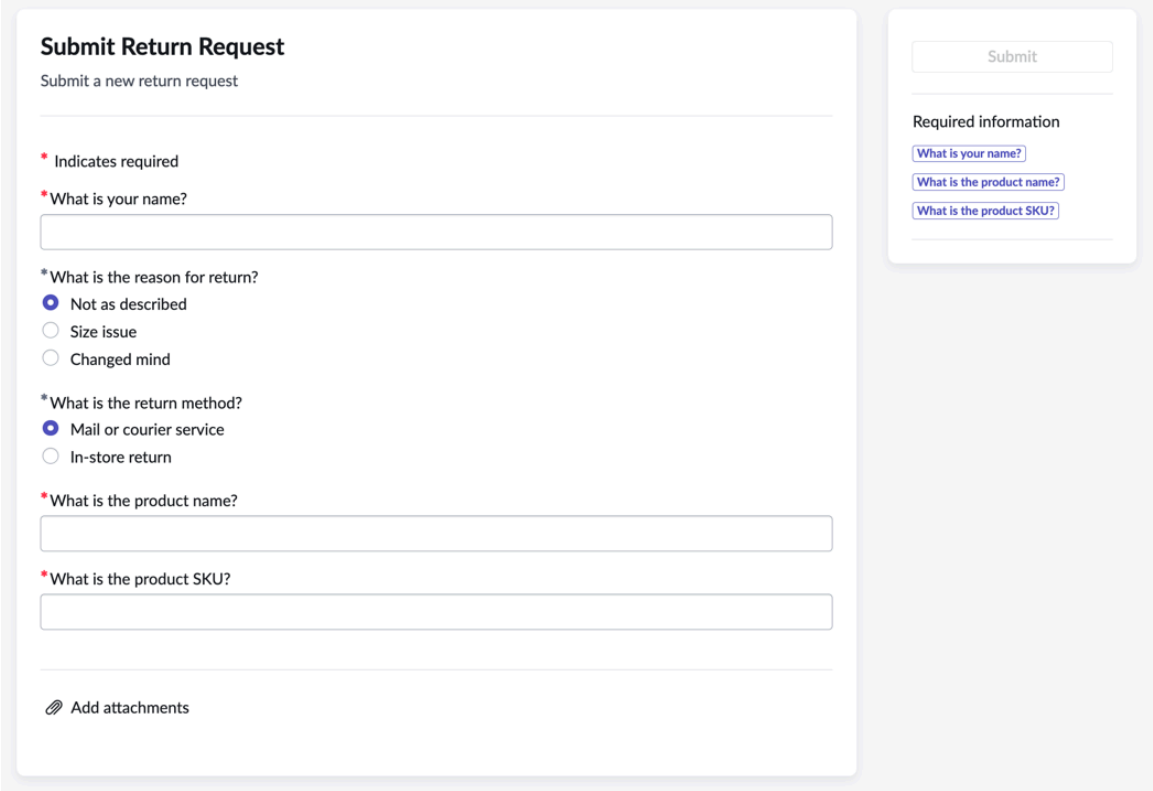
4. Select the **App Home** tab to review the generated forms and record producers.

   a. Under the **Experience** heading, select the Additional actions icon (•••) for a form or record producer in your application, and then select **Edit**.

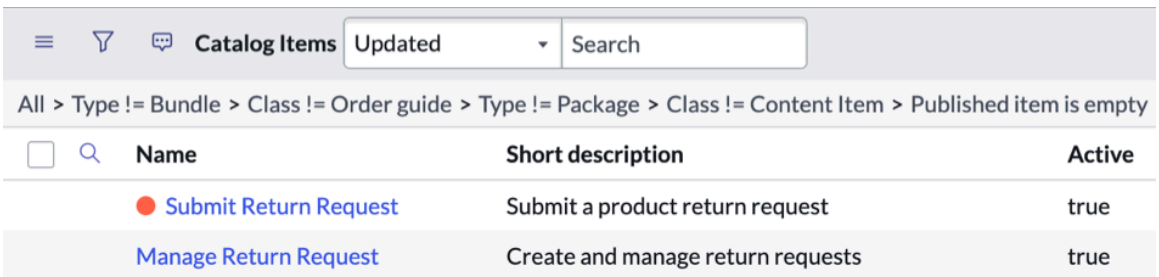
      You can also select the bar that lists the experience you want to view.

   b. Modify the item if necessary.

      You can preview the item. Confirm that the question field types match their intended purpose.



Verify that the catalog item for your record producer appears in the Service Catalog. To learn more, see Service Catalog.



If you want to make changes, you can fine-tune the experience by using App Engine Studio. For more information, see Add a record producer.

What to do next

Set up the application to perform tasks according to predefined rules or logic. You can add flows, create decision tables, build playbooks, write script includes and business rules, or configure an email notification. For more information, see Add logic and automation.

App generation in AES reference

Reference topics provide additional information about configuration properties, roles, and more.

Note:
If you using a Xanadu Store Release 2 instance, Now Assist for app generation works with ServiceNow Studio (not App Engine Studio). For detailed information, see Now Assist for app generation in ServiceNow Studio.

App generation roles for AES

The following roles are installed for use with the Now Assist for Creator app generation skill.

Note:
If you using a Xanadu Store Release 2 instance, Now Assist for app generation works with ServiceNow Studio (not App Engine Studio). For detailed information, see Now Assist for app generation in ServiceNow Studio.

Administrator [admin] role for Now Assist with AES

Access all system features, functions, and data, regardless of security constraints.

Now Assist Creator [now.assist.creator] role for Now Assist with AES

Create applications through a conversation with generative AI.

Contains Roles

List of roles contained within the role.

None.

Groups

List of groups this role is assigned to by default.

None.

Special considerations

None.

Create your app using an application template

Build an application in App Engine Studio (AES) that uses predefined data, experience, logic and automation, and security. Add to the template contents to customize the app for your organization.

Before you begin

Note:
To create apps in AES, you must be an admin or in the App Engine Studio Users group. If you are in the App Engine Studio User Limited group, you can only edit existing apps, not create new ones.

Role required: admin, sn_app_eng_studio.user

About this task

Using templates in App Engine Studio helps you save time and effort creating applications. You can download several templates from the ServiceNow Store, or you can create templates from your existing applications. To install a template on your PDI, see Activating a plugin from your PDI.

This video shows you how to perform the following procedure.

Short video of creating an app from a template

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the **Templates** tab, browse the available templates.
3. Hover over the template you want to use and select **Use template**.
4. Enter a **Name** and **Description** for your application.

5. **Optional:** Add a logo by dragging an image into the AES page or browsing your computer for the file.
6. Select **Continue**.
7. On the summary screen, select **Go to app home**.
8. From the app home, review the data, experience, logic and automation, and security that was created with the template.
   To tailor the application to your business needs, you can edit these predefined items or add your own. For more information, see Enhance your app.

What to do next

After you've finished building your application, submit the application for approval to get it reviewed and deployed by an administrator. For more information, see Submit your app for approval and publishing.

Available templates

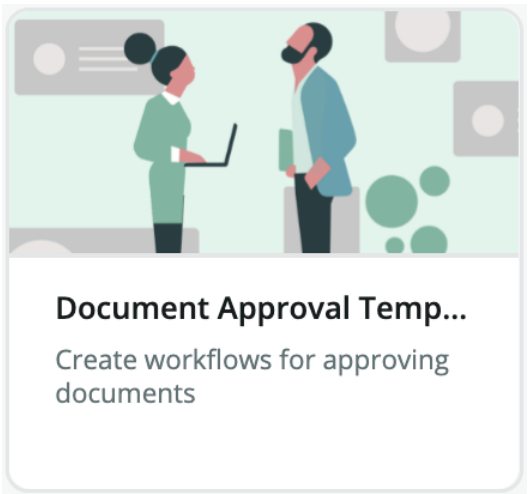
Use an application template to create an application in App Engine Studio (AES) with preconfigured data, experience, logic and automation, and security.

Templates are a good starting place for many applications you might want to build. You can also begin your application using a template and then customize the application. Use any of the templates below to build a fully-fledged application, or use them as a starting point to create an app customized to your needs.

Document Approval template

Build an application in App Engine Studio (AES) using the Document Approval template to manage the approval of documents in your organization.

To use the Document Approval template, install this application from the ServiceNow Store.



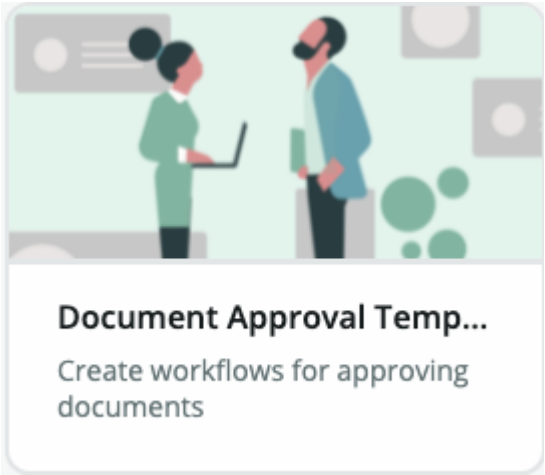
A document approval app enables your users to quickly create and use document approval workflows. Document owners and submitters can upload documents needing approvals and check the progress on those approvals. Designated approvers can approve, reject, or return documents with comments.

Admins can use the document approval application to create different workflows for different document types.

This app template is an ideal way to quickly and efficiently make your internal workflows for processing documents. Document owners and submitters can upload a document needing

approval to a portal using simple upload functionality that automatically sends submissions through a unique workflow. Once a document has been received, submitters follow the progress of their documents while designated users approve, reject, or return their document for edits. Both submitters and approvers use comments to add notes and feedback to the submission, which is stored in the system for future reference. To give this app even more variety, admins can create unlimited workflows that are customized for each document type and user that allows for multiple stages and approvers depending on what the document category requires.

Document Approval template



For more information on creating an application using a template, see Create your app using an application template.

Document Approval template contents

The following tables, experiences, automated workflows, and roles are included in the Document Approval template for App Engine Studio (AES).

Tables in the Document Approval app template

Table label [name]	Description
Document Approver Table [x_ <company-code>_ <document_approver>]	Tracks document approvers, their comments, and state of a document.
Document Approval Category Table [x_ <company-code>_ <document_approval_category>]	Tracks the category of document approval requests.
Document Approval Table [x_ <company-code>_ <document_approval>]	Tracks the document approval requests.

Experiences in the Document Approval app template

Experience	Experience type	Description
Document Approval | Default view	Form	Form for creating a document approval request.

Experience	Experience type	Description
Document Approval | Portal	Form	Simplified form for creating a document approval request.
Document Approval Category | Default view	Form	Form to attach a flow to a document approval record.
Document Approval Category | Workspace	Form	Form to attach a flow to a document approval record.
App name Workspace	Workspace	Main document approval app workspace, where users can upload documents, check the status of an approval, edit documents, and view their submissions in the workspace. Approvers can view assigned approvals, view previous submissions, and approve or reject submissions.
Document Approver | Workspace	Form	Form for assigning document approvers to records.
Document Approval | Workspace	Form	Form for creating a document approval request.
Document Approver | Default view	Form	Approvers can view assigned approvals and approve or reject submissions.
App name Portal	Portal	Main portal for people to submit documents for approvals and manage previous submissions.
Document Approval Mobile Experience	Mobile experience	Mobile app that employees and managers can use to check the status of an approval, approve or reject documents, and view submissions using your organization's iOS or Android app.

Automated workflows in the Document Approval app template

Workflow name	Workflow type	Description
Document Approval - Master Flow	Flow - record	Executed when a new document approval record is submitted. It then retrieves the flow to be executed from the document category and executes it.
Document Approval Notification	Email	Notifies users of changes to their document approvals.

Workflow name	Workflow type	Description
Single Stage Approval	Flow	Allows for the user's manager to be the approver, with basic email notifications being generated on approval, return, or rejection.

Roles in the Document Approval app template

Role title [name]	Description	Contains roles
Approvers [x_<company-name>_<document_approval_app>.approver]	Approvers can create, read, delete all documents, and read document approval category data.	Create, Read (all), Edit (all)
Admins [x_<company-name>_<document_approval_app>.admin]	Admins can create, read, write, and delete in the Document Approval [x_ <company-code>_<document_approval>] and Document Approval Category [x_ <company-code>_<document_approval_category>] tables.	Create, Read (all), Edit (all), Delete (all)
Submitters [x_<company-name>_<document_approval_app>.submitter]	Submitters can create documents, read their own documents, edit their own documents when status is not submitted and delete their own documents. They can also read data from document approval category.	Create, Read (own), Edit (own when status is not submitted), Delete (own)

Emergency Alert template

Build an application in App Engine Studio (AES) using the Emergency Alert template to manage communications and track essential resources during an emergency.

To use the Emergency Alert template, install this application from the ServiceNow Store.

An emergency alert app allows your organization to keep employees informed and connected during natural disasters and infectious diseases, such as COVID-19.

You can use the Emergency Alert template to create command center dashboards and a mobile app to monitor and notify employees about the latest news. Employees can self-report their status, request work-from-home or time off, and receive instructions during an emergency. Managers can assess the impact on their respective teams, approve or reject requests, request for status, and automatically receive instruction on actions to take during any emergency.

For more information on creating an application using a template, see Create your app using an application template.

Emergency Alert template contents

The following tables, experiences, automated workflows, and roles are included in the Emergency Alert template for App Engine Studio (AES).

Tables in the Emergency Alert app template

Table label [name]	Description
Status Report Request Table [x_<company>_<my emergency_alert_app>_status_report_requests]	Track the severity of an emergency reported by employees. For example, the status, the color associated with danger levels, and conditions.
Self Report Table [x_<company>_<my emergency_alert_app>_status_report]	Track reported emergencies created by employees or managers. For example, the emergency details, status, duration, and affected individuals.
Emergency Alert Table [x_<company>_<my emergency_alert_app>_emergency_alert]	Track emergency alerts for your company. For example, the emergency type, who created the alert, and who has read the alerts.
Work Status Request Task Table [x_<company>_<my emergency_alert_app>_work_status_request_task]	Track the work status request of an emergency. For example, the requester, the assignee, and the state of the work status.

Experiences in the Emergency Alert app template

Experience	Experience type	Description
Emergency alert | SendEmergency AlertWS	Form	Form to create a new emergency alert record.
Self Report | Employee View	Form	Form for employees to report an emergency.
Self Report | Default View	Form	Form for employees to report an emergency.
Emergency Alert - Portal	Portal	Portal where employees can view reported emergencies, the status of their team, and submit a self report.
Emergency Alert | Default View	Form	Form to create a new emergency alert and connect knowledge resources to it.
Self Report | Update Report	Form	Form for employees to self report their health status.
Self Report | Activity View	Form	Form to add comments about a self report.
Self Report | RequestReportWS	Form	Form to self report an emergency within a specific time frame.
Status Report Request | Default View	Form	Form to send an update about an emergency with time and location details.
Self Report | Update status report	Form	Form to add an update about an emergency.
App name Workspace	Workspace	Workspace where admins can set up and send notifications, view dashboards, and configure lists and forms.
Work Status Request Task | Default view	Form	Form to update an employee about a work status request.

Automated workflows in the Emergency Alert app template

Workflow name	Workflow type	Description
Emergency report/update state for managers	Email	Notifies managers to update an emergency report if the source is empty.
Emergency Alert App - Please report or update status	Flow - record	Requests a status update on work.

Workflow name	Workflow type	Description
Emergency Alert App - Status request for employees from manager	Flow - record	Notifies employees that a status request is made by their manager.
Self report submitted without manager	Email	Notifies that an employee submitted an expense request without listing their manager.
Emergency Alert App - Create Work Status Task When Self Report is created	Flow - record	Starts the work status task when a self report emergency is submitted.
Emergency Alert App - Create Status Report Requests for Affected	Flow	Creates or updates status report requests.
Emergency alert app - Create App Self Reports	Flow - record	Sends a notification when a report request is created or updated.
Emergency Alert App - Request for WFH/Time off submitted	Flow - record	Starts the work from home or time off request approval flow.
Emergency instructions for employees	Email	Notifies users if a status update is requested.
Emergency Alert App - Employee emergency report changes to submitted	Flow - record	Sends a notification when an employee self report has been submitted.
Emergency Alert App - Request for WFH/Time off approved/denied	Flow - record	Sends a notification to an employee when a WFH or Time off request is approved or denied.
Emergency Alert App - Employee emergency report request response submitted	Flow - record	Sends a notification when an employee emergency report has been submitted.
Request for WFH approved	Email	Notifies an employee that their request to work from home was approved.
Emergency alert app - Emergency instructions for employees	Flow - record	Sends employees instructions to follow during an emergency.
Please report/update status of employee	Email	Requests information about the status of an employee.
Self-report submitted for manager	Email	Notifies a manager that an emergency alert has been created and needs review.
Emergency Instructions for managers	Email	Notifies managers that their request was created and that their team will receive information about it.

Workflow name	Workflow type	Description
Please report/update status	Email	Sends a notification when a status change has been committed.
Request for WFH/Time off submitted	Email	Notifies an employee that their work from home or time off request was successfully submitted.
Employee Emergency report submitted	Email	Notifies an employee that their emergency report was successfully submitted.
Emergency Alert App - Manager emergency request submitted	Flow - record	Sends a notification when a manager emergency report has been submitted.
Emergency Alert App - Employee emergency report submitted	Flow - record	Sends a notification when an employee submits changes to an emergency report.
Create KB article	Flow - record	Creates a knowledge base article when a new emergency alert is created.
Employee emergency report request submit	Email	Notifies an employee that their request for status was submitted and that their manager will be notified.
Emergency Alert App - Cancel work status task if self report closed	Flow - record	Cancels a work status request if the self report is closed.
Manager emergency request submitted	Email	Notifies managers that the emergency report was sent.
Please report/update status of employee	Email	Notifies employees of action needed on their self report.
Request for WFH denied	Email	Notifies the employee that their work from home request was denied.
Emergency Alert App - Emergency report/update state for managers	Flow - record	Sends a notification based on changes in an emergency alert report.

Roles in the Emergency Alert app template

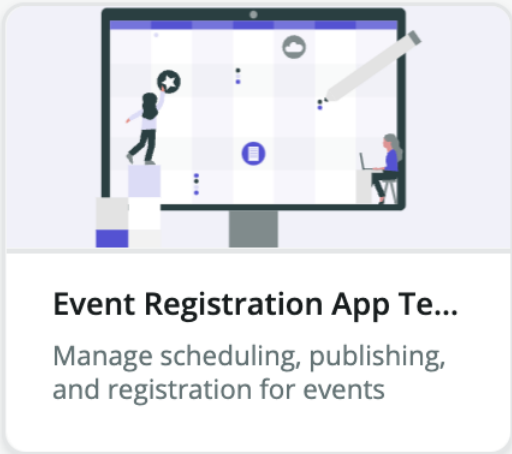
Role title [name]	Description	Contains roles
Emergency alert admin [x_<company-code>_ <my emergency_alert_app>.emergency_alert_admin]	Admins manage the app and configure the dashboard	Create, Read (all), Edit (all), Delete
Emergency alert manager	Managers can approve or reject requests, access	Create, Read (all), Edit (all)

Role title [name]	Description	Contains roles
[x_<company-code>_ <my emergency_alert_app>.manager]	dashboards, and send notifications	
Emergency alert employee [x_<company-name>_ <my emergency_alert_app>.emergency_alert_employee]	Self-report emergency alerts	Create, Read (own), Edit (own)
Admin	The System Administrator role. This role has access to all system features, functions, and data regardless of security constraints.	Create, Read (all), Edit (all), Delete
Public	No login is required to access features or functions with the public role.	No access by default.

Event Registration template

Build an application in App Engine Studio (AES) using the Event Registration template to manage scheduling, booking, and registration for company events.

To use the Event Registration template, install this application from the ServiceNow Store.



An internal event registration app allows your event organizers to publish a list of events to employees and collect attendees in a centralized database. The app helps streamline the event discovery and registration process.

Event organizers can manage events and wait-lists, publish new events, and view registered users' details. Employees and attendees can view a list of events, view event info, and register to attend in a portal. Organizers can set the size of the event, and requesters will automatically be added to a wait-list.

For more information on creating an application using a template, see Use an application template.

Event Registration template contents

The following tables, experiences, automated workflows, and roles are included in the Event Registration template for App Engine Studio (AES).

Tables in the Event Registration app template

Table label [name]	Description
Attendee Table [x_<company-code>_<attendee>]	Track event attendee information such as name, status, and date of submission.
Event Table [x_<company-code>_<event>]	Track event information such as event name, date of the event, and capacity.
Event Location Table [x_<company-code>_<event_location>]	Track event location details such as capacity and location.

Experiences in the Event Registration app template

Experience	Experience type	Description
Event Location | Workspace	Form	Workspace where organizers and admins can configure events, create event applications, and view lists.
App name Portal	Portal	Main portal for organizers to create and view details about the event.
Attendee | Default view	Form	Form that attendees can use to view the event name, change their status, and take notes about the event.
Event | Default view	Form	Form that organizers can use to create events.
App name Workspace	Workspace	Workspace where users can register for an event and where admins/organizers can manage registration information.
Event Location | Default view	Form	Form for organizers to add event location name, location address, status, and capacity.
Event | Workspace	Form	Form for organizers to add event name, location details, timelines, and comments.
Attendee | Workspace	Form	Form for attendees to register for events and add notes about the event.
Mobile app for organizers	Mobile experience	Organizers can view or edit events and view attendees using your organization's iOS or Android app.

Automated workflows in the Event Registration app template

Workflow name	Workflow type	Description
Event Details Have Changed	Email	Notifies users/attendees of changes to the event.
Event Reminder - Confirmed attendees	Email	Sends a reminder about the event to confirmed attendees.
Attendee registered confirmed	Email	Notifies attendees of confirmed registration.
Send Event Reminder Email	Flow	Sends a reminder email to event attendees.
Send Event Details Changed Email	Flow - record	Sends an email to all confirmed and waitlisted attendees when details of an event have changed or the event was cancelled.
Send Waitlist Confirmation Email	Flow - record	Sends an email to attendees when they are added to the waitlist of an event.
Event Registration cancelled events	Flow - record	Updates attendee records when an event is cancelled.
Update Event Registration attendee records	Flow	Updates attendee records when an event is cancelled or is past the end date.
Attendee waitlisted confirmed	Email	Confirms attendees on event waitlist.
Event Registration expired events	Flow	Sets the **Event state** field to Expired and the **Attendees** field to Event ended after the event ends.
Send On Hold Confirmation	Flow - record	Allows for organizers to send an email to attendees when they are on-hold to attend an event.
Event Registration Confirmation Email	Flow - record	Sends an email to attendees when they are confirmed to attend an event.
Attendee registration on hold	Email	Informs attendees that their registration is on hold.
Event Has Been Cancelled	Email	Informs users of event cancellation.
Event Registration Process Flow	Flow	Processes all pending requests.

Roles in the Event Registration app template

Role title [name]	Description	Contains roles
Organizer [x_<company-name>_<my events app>.organizer]	Organizers create and manage events.	Create, Read (all), Update (all).
Attendee [x_<company-name>_<my events app>.attendee]	Attendees can search, sign up, and RSVP to events.	Create, Read (all), Update (own).
Admin [x_<company-name>_<my events app>.admin]	Admins configure event locations and applications.	Create, Read (all), Update (all), Delete.
Public [x_<company-name>_<my events app>.public]	No login is required to access features or functions with the public role.	No access by default.

Expense Pre-Approval template

Build an application in App Engine Studio (AES) using the Expense Pre-Approval template to create, track, and manage planned expenses in your organization.

To use the Expense Pre-Approval template, install this application from the ServiceNow Store.

Expense Pre-Approval Te...

Create, track, and manage planned expenses in your organization

Employees can easily submit their planned expenses for multiple levels of approval using the Expense Pre-Approval application. Managers can then review every submitted request. Approved requests are assigned to the budget review group, which can approve or reject the request. The employee receives a notification with the final status of the request.

To ensure compliance and improved financial planning, admins can customize the existing flow and levels of approval using decision tables. For more information, see Exploring decision tables.

For more information on creating an application using a template, see Create your app using an application template.

Expense Pre-Approval template contents

The following tables, experiences, automated workflows, and roles are included in the Expense Pre-Approval template for App Engine Studio (AES).

Tables in the Expense Pre-Approval app template

Table label [name]	Description
Budget definition [x_<company-code>_<expense_approval_app>_budget_definition]	The description of all budgets.
Expense request [x_<company-code>_<expense_approval_app>expense_request]	The expense requests submitted by employees.
Expense line item [x_<company-code>_<expense_approval_app>_expense_line_item]	Each expense line item in an expense request.

Experiences in the Expense Pre-Approval app template

Experience	Experience type	Description
Expense request | Default view	Form	Form to create a new expense request.
Budget definition | Default view	Form	Form to define budgets for a specified time frame.
Budget Reviewer	Mobile experience	Way to define what the mobile experience for budget reviewers looks like.
Expense request | Workspace	Form	Form to create a new expense request.
Expense request | Cost Approval	Form	Form to create a new expense request from an admin.
Manager	Mobile experience	Way to define what the mobile experience for managers looks like.
Budget definition | Workspace	Form	Form in the workspace view to define budgets for a specified time frame.
App name - Budget workspace	Workspace	Way to configure a high-level view of all budget requests.
Expense request	Record producer	Way for employees to submit an expense request.
Expense line item | Default view	Form	Form to add an expense line item to an expense request.

Experience	Experience type	Description
Expense Request SLA | Default view	Form	Form to create an expense request based on a breach of service level agreement (SLA).
Budget definition | Cost Approval	Form	Form in the cost approval view to define budgets for a specified time frame.
App name - Manager workspace	Workspace	Workspace where managers can approve or reject the expense requests assigned to them.
Cost Approval Mobile App	Mobile experience	iOS or Android app used for the following actions: • Approving or rejecting expense requests. • Viewing dashboards that display historical data to compare budget and expense requests. • Creating or modifying budgets by budget reviewers.
Cost Approval Workspace - Budget Reviewer	Workspace	Budget reviewers can use the workspace for the following actions: • Approving or rejecting expense requests. • Creating or modifying budgets. • Viewing the data on the budget funds allocated, spent, available, and allocated for the future. • Viewing dashboards that display historical data to compare the budget and expense requests.

Automated workflows in the Expense Pre-Approval app template

Workflow name	Workflow type	Description
Expense request approved - reviewer	Email	Notifies users that an expense request has been approved.
Expense Request Assigned	Email	Notifies users that an expense request has been assigned.

Workflow name	Workflow type	Description
Expense request - SLA breached	Email	Notifies users that a service level agreement (SLA) has been breached.
Budget Creation	Flow - record	Executed when a budget reviewer creates a budget. It checks the start date of an expense request and associates the request to a budget with corresponding time frame.
Expense request - SLA- close to due	Email	Notifies users that a service level agreement (SLA) is close to due.
Expense Request Reassigned	Email	Notifies users that their request was reassigned.
Expense request comment added - Assigned	Email	Notifies users about a comment that has been added to their request.
New Expense Request or manager approval	Email	Notifies users about a new expense request.
Expense request reject	Email	Notifies users that their request was rejected.
Budget state update	Flow	Updates each budget record by priority.
Expense Request Inserted - Requestor	Email	Notifies users that their request was successfully created.
Expense Request Approved	Email	Notifies users that their request was approved.
Expense request work note added	Email	Notifies users a work note was added to their request.
Notify Team - Work note added	Email	Notifies a team that a work note was added to a request.
Expense request comment added - Employee	Email	Notifies users a comment was added to a request.
Expense Request Rejected	Email	Notifies users that their request was rejected.
Budget Updation	Flow - record	Executed when a budget reviewer modifies the budget. After a budget is modified, this flow updates the **Budget** field in the corresponding expense requests.
Notify Requestor - Comments added	Email	Notifies users that a comment was added to a request.

Workflow name	Workflow type	Description
SLA Notification - Expense Request - warning	Flow - record	Warns admins about a potential service level agreement (SLA) issue.
Budget Approval Process	Flow - record	Executed to obtain expense approvals in the following scenarios: • When an employee submits an expense request for the manager's approval. • Upon the manager's approval, the expense request is sent to the budget reviewer for the final level of approval.
Budget Calculations	Flow	Recalculates the budget spent and the available budget when a budget reviewer performs any of the following actions: • Creates a budget • Modifies an existing budget • Approves an expense request

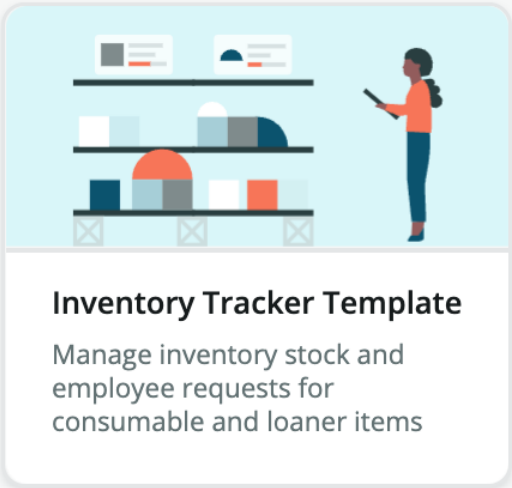
Roles in the Expense Pre-Approval app template

Role label [name]	Description
Budget Reviewer [x_<company-code>_<expense_approval_app>.Budget reviewer]	Budget reviewers can create or modify budgets and provide the final level of approval on expense requests.
Manager [x_<company-code>_<expense_approval_app>.Manager]	Managers can submit, view, and approve or reject expense requests at the first level.
Employee [x_<company-code>_<expense_approval_app>.Employee]	Employees can submit and track expense requests.
Admin	The System Administrator role. This role has access to all system features, functions, and data, regardless of security constraints.

Inventory Tracker template

Build an application in App Engine Studio (AES) using the Inventory Tracker template to track the status of inventory and assets.

To use the Inventory Tracker template, install this application from the ServiceNow Store.



An inventory tracker app enables your business and teams to easily track what they have, where it is, and who has it. Anytime, anywhere, and on any device. If your company needs to track the stock of items, this app can be used as a stock management tool or inventory tracker at home or in a warehouse.

This app covers three inventory use cases: consumables, loaners, and discontinued. Employees can request items via mobile app or web portal from any warehouse. Inventory managers can set reorder levels and will be automatically notified if an item is low on stock. Discontinued items with outstanding stock can still be requested but will not prompt inventory managers for reorder. Inventory managers can create items manually or by uploading a spreadsheet.

For more information on creating an application using a template, see Create your app using an application template.

Inventory Tracker template contents

The following tables, experiences, automated workflows, and roles are included in the Inventory Tracker template for App Engine Studio (AES).

Tables in the Inventory Tracker app template

Table label [name]	Description
Inventory Control Table [inventory_ x_<company-code>_<my_inventory_tracker_app>_inventory_control]	Track the stock of items used by your business. For example, item name, manufacturer, image, category, and location.
Inventory Request Table [x_<company-code>_<my_inventory_tracker_app>_inventory_request]	Track inventory requests submitted by employees. For example, the name and department of the requester, stock location, count, requested date, and return date.
Stock Location Table	Track the location of stocked items. For example, name, address, and inventory manager.

Table label [name]	Description
[x_<company-code>_<my_inventory_tracker_app>_stock_location]	

Experiences in the Inventory Tracker app template

Experience	Experience type	Description
Inventory Control | Portal	Form	Form for admins and managers to add inventory items, view stock quantity, and add stock locations.
Inventory request | Workspace	Form	Form for employees to submit and manage inventory requests.
Inventory Control | Default view	Form	Form for admins and managers to manage inventory items, view stock quantity, and manage stock locations.
App name | Portal	Portal	Portal where all users can manage inventory requests and tracking.
Inventory request | Default view	Form	Form for admins and managers to inventory items, view stock quantity, stock locations, and add notes.
Inventory request | Portal	Form	Form for admins and managers to request inventory items.
App name | Workspace	Workspace	Workspace to manage all Inventory tracker app configuration.
Stock Location | Workspace	Form	Form for inventory managers to add stock names, addresses, and descriptions.
Stock Location | Default view	Form	Form for inventory managers to add stock name, address, description, and department.
Inventory Control | Workspace	Form	Form for admins and managers to manage inventory items, view stock quantity, and stock locations.
Inventory Tracker App	Form	Form for employees to view inventory list, see stock quantities, request items, and check the status of their requests using their

Experience	Experience type	Description
		organization's iOS or Android app.

Automated workflows in the Inventory Tracker app template

Workflow name	Workflow type	Description
Inventory Tracker Request Completed - Admin/Manager	Flow - record	Sends a notification when an inventory request has been completed.
Request Update as Submitted - Manager	Email	Notifies a manager that they have a new update on a requested item.
Item Out of Stock- Admin	Email	Informs an admin of items that are out of stock.
Return Date Updated - Manager	Email	Notifies manager of an updated return date.
Inventory Tracker Request Submitted - Manager	Flow - record	Notifies managers when an inventory request is submitted.
Inventory Tracker Inventory track Created - Admin	Flow - record	Notifies admins when an inventory track is created.
Stock Location Updated - Admin	Email	Informs an admin of updated stock location.
Inventory Tracker Checked Out - Employee	Flow - record	Notifies an employee when an item is checked out.
Request Overdue - Manager	Email	Informs a manager of an overdue request.
Item Returned - Manager	Email	Informs a manager of returned items.
Request Completed - Manager	Email	Informs a manager of a completed request.
Inventory Tracker Request Approved - Employee/Admin/ Manager	Flow - record	Notifies employees, admins, and managers when an inventory request is approved.
Inventory Tracker Request Submitted - Employee	Flow - record	Notifies employees when their inventory request is submitted.
Checked Out - Employee	Email	Notifies employees of items that are checked out.
Inventory Tracker Out of Stock - employee	Flow - record	Notifies employees if an inventory item is out of stock.
Return Date Updated - Admin	Email	Informs an admin of an updated return date.

Workflow name	Workflow type	Description
Request Canceled - Manager	Email	Notifies a manager when a request is cancelled.
Inventory Tracker Stock Location Updated - Admin	Flow - record	Notifies admins when a stock location is updated.
Item Out of Stock - Manager	Email	Notifies a manager when an item is out of stock.
Inventory Tracker Item Returned - Employee/ Manager	Flow - record	Notifies an employee or manager when an item is returned.
Inventory Tracker Inventory track updated- Admin	Flow - record	Notifies an admin when an inventory track is updated.
Item returned - Employee	Email	Notifies employees of returned items.
Request Denied - Admin	Email	Informs admins of denied requests.
Inventory Tracker Stock Location Created - Admin	Flow - record	Notifies an admin when a stock location is created.
Inventory Item Discontinued - Admin	Email	Notifies admins of discontinued items.
Request Approved - Manager	Email	Notifies managers of an approved request.
Inventory Item Updated - Admin	Email	Notifies admin of updated inventory item.
Inventory Tracker Inventory track Created - Admin	Flow - record	Notifies an admin when an inventory track is created.
Out of stock - Employee	Email	Notifies employees of items that are out of stock.
Inventory Tracker Request Denied - Admin	Flow - record	Notifies admins when an inventory request is denied.
Request Approved - Employee	Email	Notifies employees of approved requests.
Inventory Tracker Item Out of stock - Admin/Manager	Flow - record	Notifies admins and managers when an inventory item is out of stock.
Request Approved - Admin	Email	Notifies admins of an approved request.
Request submitted - Manager	Email	Notifies admins of a submitted request.
Request Overdue - Admin	Email	Notifies admins of an overdue request.
Request Denied - Manager	Email	Notifies managers of a denied request.

Workflow name	Workflow type	Description
Stock Location Created - Admin	Email	Notifies admin when a stock location is created.
Overdue - Employee	Email	Notifies employee of overdue items.
Inventory Tracker Request Updated - Manager	Flow - record	Notifies managers when an inventory request is updated.
Request Denied - Employee	Email	Notifies employee of a denied request.
Inventory Tracker Inventory Discontinued - Admin	Flow - record	Sends a notification when an inventory item is discontinued.
Inventory tracker Return Date Changes - Admin/ Manager	Flow - record	Sends a notification to admin and manager when a return date changes.
Comments Added - Manager	Email	Notifies a manager about added comments.
Comments Added - Admin	Email	Notifies admins about added comments.
Inventory tracker Request Denied - Employee	Flow - record	Sends a notification to employees when their inventory request is denied.
Request Submitted - Employee	Email	Notifies employees when they submit a request.
Request created - Admin	Email	Notifies admins when a request is created.
Inventory Tracker Comments Added - Admin/Manager	Flow - record	Sends a notification to admins and managers when comments are added to inventory requests.
Request Canceled - Admin	Email	Notifies admins of a canceled request.
Inventory Tracker Request Overdue - Employee/ Manager/Admin	Flow	Sends a notification to employees, managers, and admins if a request is overdue.
Inventory Item created - admin	Email	Notifies an admin when an inventory item is created.
Request Completed - Admin	Email	Notifies an admin when a request is completed.
Inventory tracker Request Cancelled - Admin/Manager	Flow - record	Sends a notification to admins and managers when an employee cancels an inventory request.
Inventory tracker Request Created - Admin	Flow - record	Sends a notification to admins when an inventory request is created.

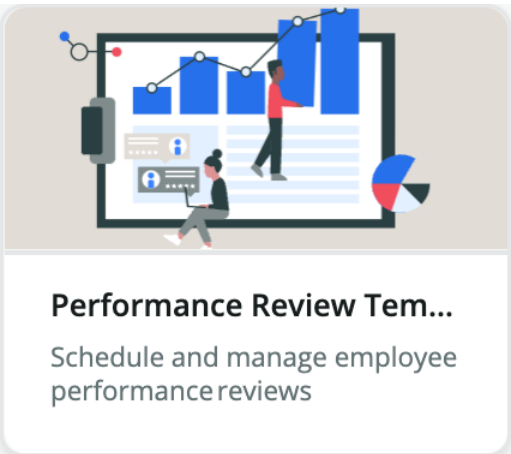
Roles in the Inventory Tracker app template

Role title [name]	Description	Contains roles
Employee [x_<company-name>_<my_inventory_tracker_app>.employee]	Employees can view inventory lists, check stock, request items, and check the status of their requests.	Create, Read (own), Update (own)
Admin [x_<company-name>_<my_inventory_tracker_app>.admin]	Admins can set up multiple inventory managers and stock locations.	Create, Read (all), Update (all), Delete
Inventory Manager [x_<company-name>_<my_inventory_tracker_app>.inventory_manager]	Inventory managers can create inventory items, define the type of item, monitor inventory levels, and update vendor information.	Create, Read (all), Update (all)
Public [x_<company-name>_<my_inventory_tracker_app>.public]	Public users require no login to access features or functions with the public role.	Contains no roles.

Performance Review template

Build an application in App Engine Studio (AES) using the Performance Review template to automate scheduling, notifications, and feedback for performance reviews.

To use the Performance Review template, install this application from the ServiceNow Store.



A performance review app automates the process of scheduling annual or quarterly reviews, reviewing employee performance, and providing manager feedback. The app displays report information by default and lets managers view, update, and request employee performance reviews.

Employees can complete their self-evaluation, monitor the status, and confirm evaluations via a web portal. The app will automatically notify employees and managers to submit their evaluations.

Managers can request an update to the self-evaluations, submit manager-evaluations, confirm evaluations, and monitor the status of their team's submissions via a portal.

The HR/admin role can select an annual, bi-annual, or quarterly review process schedule and edit both the self-evaluation and manager-evaluation templates. HR/admins can monitor the status of all evaluations by department, organization, or manager.

You can improve this template by incorporating the following features:

- Add continuous option as a review process
- Use Playbook to manage review schedules
- Include HR review/escalation process for both self-reviews and manager reviews
- Add peer reviews process
- Add goal setting or objectives and key results (OKRs)
- Add integration with HR software
- Add skill set assessment metrics
- Add performance improvement plan (PIP) process/individual development plans
- Include compensation info/annual performance review (APR)

For more information on creating an application using a template, see Create your app using an application template.

Performance Review template contents

The following tables, experiences, automated workflows, and roles are included in the Performance Review template for App Engine Studio (AES).

Tables in the Performance Review app template

Table label [name]	Description
Performance Review Table [x_<company-code>_<my performance_review_app>_performance_review]	Track the performance evaluations created by employees and managers. For example, the employee name, state of the review, manager feedback, and state of the review.

Experiences in the Performance Review app template

Experience	Experience type	Description
Performance Review | Edit Settings	Form	Form where managers can configure performance review settings and add notifications.
App name | Workspace	Workspace	Workspace where admins and managers can configure review schedules, update evaluation templates, and monitor the status of all evaluations.
Performance Review Test | Emp Mgr feedback	Form	Form for managers to create employee evaluation questions, monitor the status, and leave feedback.

Experience	Experience type	Description
Performance Review | Manager Review	Form	Form managers can use to include questions, review employees responses, and leave feedback.
Performance Review | Self Review WS	Form	Form for employees to complete self review questions.
Performance Review | Workspace	Form	Form for mangers to monitor the status of their team's submissions.
App name | Portal	Portal	Portal where employees can submit, update, review, and confirm evaluations.
Performance Review | Default View	Form	Form that managers and employees can use to view performance reviews.
Performance Review | Self Review	Form	Form employees and managers can use to submit a self-assessment of their performance.
Mobile app	Mobile experience	Employees can access their performance reviews using your organization's iOS or Android app.

Automated workflows in the Performance Review app template

Workflow name	Workflow type	Description
Perf Review - Send PR Employee Update Self-Eval email	Flow - record	Triggers the PR Employee Update Self-Eval email.
PR Employee Manager - Eval submitted	Email	Notifies managers that an evaluation has been submitted.
Perf Review - Send PR Employee Complete Evaluation email	Flow - record	Triggers the PR Employee Complete Evaluation email.
PR Employee Manager - Eval submitted DJ	Email	Notifies managers when self-evaluations are submitted.
PR Admin Eval Deadline Passed	Email	Notifies admins when an evaluation deadline has passed.
PR Reminder: Manager need to submit eval	Email	Reminds managers to submit their evaluations.

Workflow name	Workflow type	Description
PR Manager Self Evaluation Submitted	Email	Notifies admins that managers have submitted self-evaluations.
Perf Review - Send PR Employee Manager - Eval submitted	Flow - record	Triggers the PR Employee Manager - Eval submitted email.
PR Reminder - Employee Self Evaluation	Email	Notifies employees to submit self-evaluation.
Perf Review - Send deadline email	Flow	Triggers a deadline reminder email on the date indicated in the system properties.
PR Employee Complete Evaluation	Email	Notifies employees their evaluation is complete.
PR Employee Update Self Evaluation	Email	Notifies employees they need to update their self-evaluation.
Perf Review - Main Flow	Flow	Controls when reviews are triggered, email reminders, and end of review period is reached. This is the main flow that triggers daily.
Perf Review - Send reminder emails	Flow	Triggers reminder emails on the date indicated by sys_properties.
Perf Review - Create records	Flow	Triggers review records on the date indicated by sys_properties.
Perf Review - Set review inactive	Flow	Inactivates reviews on the end date of the review period, specified in sys_properites.
PR Admin Self - Eval Submitted w/o Manager	Email	Notifies admins if a self-evaluation is submitted without a manager or title.
PR Reminder Employee Update Self Eval	Email	Notifies employees to update self-evaluation.
Perf Review - Send Manager/ Admin Self-Evaluation Submitted email	Flow - record	Triggers the PR Admin Self-Evaluation Submitted without manager or the PR Manager Self Evaluation Submitted email.

Roles in the Performance Review app template

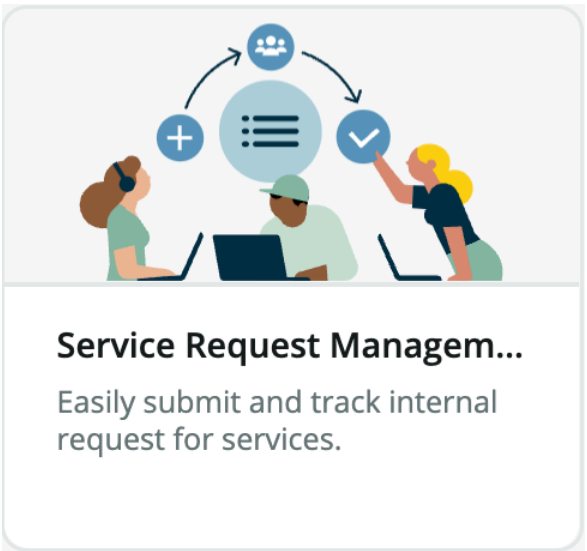
Role title [name]	Description	Contains roles
Employee	Employees can complete/ update self-evaluations and confirm evaluations.	Read (own) and Update (own)

Role title [name]	Description	Contains roles
[x_<company-name>_ <my performance_review_app>.employee		
HR/Admin [x_<company-name>_ <my performance_review_app>.admin	HR/Admins can schedule reviews (annual, bi-annual, custom, or manual), edit self and manager evaluation templates, and monitor the status of all reviews.	Create, Read (own), Update (own), Delete (all)
Manager [x_<company-name>_ <my performance_review_app>.manager	Managers can request updates to a self-evaluation, submit manager-evaluations, and monitor issues related to evaluation submissions.	Read (my team) and Update (my team)

Service Request Management template

Build an application in App Engine Studio (AES) using the Service Request Management template to create, track, and manage service requests in your organization.

To use the Service Request Management template, install the application from the ServiceNow Store.



Personas in the Service Request Management template

This template contains several important roles that correspond to individuals or groups who may use the app in some way. The following information describes how each persona may interact with the application you build. For more information on the permissions each role contains, see the Security section.

Requester

You are a person who has a request.

Problem: You need a service or a physical item to complete your responsibilities. You want to know that your request is in progress and reassurance that it is making progress. If you have questions or your request has changed, you want the ability to communicate that to those working on the request.

App objective: Empower employees by guiding them through the process, including what information is required. Make the approval process transparent, and keep them updated on the status and outcome of their request.

Fulfiller

You are a person or team who must complete a task in a timely manner.

Problem: You only want to be assigned a task when it is ready to be fulfilled, meaning after required approvals or dependent tasks (if relevant) are closed. You need to ensure that requests are closed in a timely fashion, especially in adherence with SLAs.

App objective: Direct the fulfiller's attention to the tasks assigned to them and their team. Empower them by providing the information they need such as request information and SLA.

Fulfiller's manager

You are the manager of the fulfillment team, and you can fulfill tasks.

Problem: You need to ensure tasks are being efficiently closed by your team. You want to spot bottlenecks such as popular service requests creating a backlog of tasks, a fulfiller who is under-performing, or late approvals blocking fulfillers from completing their tasks on time.

App objective: Direct the fullfiller's manager to tasks that need immediate attention so SLAs are met. Clearly show what their team is working on so they can re-prioritize work as needed.

Service owner

You are the owner of the services within the app.

Problem: You need to analyze how services are performing and identify which services need attention. You need information that will help you recommend improvements to the process, if necessary.

App objective: Provide aggregate information that indicates each service's performance, measured by resolution and response time.

Approver

You are the gatekeeper and can respond to or reject requests, saving time for fulfillers.

Problem: You must review requests before they are assigned to a fulfillment team to ensure the ticket is appropriate.

App objective: Direct the approver's attention to the requests that require their approval. Empower them by providing the information they need such as request and SLA information.

Related concepts

For information about how to set up a service for your Service Request Management application, see Service catalog setup.

Service Request Management template contents

The following tables, experiences, automated workflows, and roles are included in the Service Request Management template for App Engine Studio (AES).

Tables in the Service Request Management app template

Table label [name]	Description
Primary [x_ <company-code>_<primary>]	Table that extends the Task table and stores the details of requests raised for the Service Request Management: Primary catalog item. To create a new service and table, extend this Primary table.
Example Request [x_ <company-code>_<example_request>]	Example Request table created by extending the Primary table that stores the details submitted from the Travel Request catalog item.
Fulfiller Task [x_ <company-code>_<fulfiller_task>]	Stores fulfiller tasks and helps admins keep service requests organized.

Experiences in the Service Request Management app template

Experience	Experience type	Description
Travel Request | Fulfiller view	Form	Form for fulfillers to create a travel request record.
Primary | Fulfiller view	Form	Form for fulfillers to create a service request record.
Fulfiller task | Default view	Form	Form for users to create a fulfiller task.
Fulfiller Workspace	Workspace	Workspace for fulfillers to see requests that need their attention, create requests, view request assignments, and see data and analytics related to their requests.
Approver Workspace	Workspace	Workspace for approvers to see requests pending their approval, approve or deny requests, and see data and analytics related to their requests.
Fulfiller Task | Fulfiller task view	Form	Form for users to create a fulfiller task.
Service Request Management: Primary	Record producer	Contains questions to add information and attach documents relevant to a service request.
Travel Request	Record producer	Contains questions to add information and attach documents relevant to a travel request.

Experience	Experience type	Description
Primary | Default view	Form	Form for users to create a service request record.
Fulfiller's Manager	Workspace	Workspace for the fulfiller's manager to see requests pending their approval, create requests, approve or deny requests, and see data and analytics related to their and their team's requests.
Travel Request | Default view	Form	Form for users to create a travel request record.

Automated workflows in the Service Request Management app template

Workflow name	Workflow type	Description
Fulfiller task work notes added	Email	Notifies users in the Assigned to field, Work notes list, and Watch list when work notes are added to a fulfiller task.
Fulfillment Process Trigger	Email	Notifies a service owner when a fulfiller task has been created but has no default Fulfiller or Fulfillment Assignment Group.
On Insert empty Assigned to	Email	Notifies an assignment group when a request record has been created and has no assignee.
On Insert/Change Assigned to	Email	Notifies a user that a request record has been assigned to them.
Assignment Process	Flow - record	Creates and updates a new record as it moves through the assignment, approval, and fulfillment processes.
Fulfiller Decision	Decision table	Defines the default fulfiller user and fulfiller assignment group for each service.
Approval Process	Flow	Moves a record through the approval process.
Approver Decision	Decision table	Defines the default approver and approver assignment group for each service.
Fulfillment Process	Flow	Moves a record through the fulfillment process.
Child worknotes added	Email	Notifies users named in the **Assigned to** field, Work notes

Workflow name	Workflow type	Description
		list, and Watch list when work notes are added to a travel request task.
Primary worknotes added	Email	Notifies users in the **Assigned to** field, Work notes list, and Watch list when work notes are added to a request record on the Primary table.

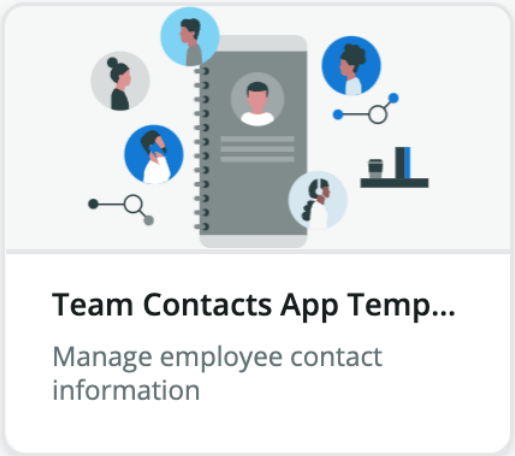
Roles in the Service Request Management app template

Role title [name]	Description
Approver [x_<company-name>_ <app_name>.approver]	Provides access to approver workspace and provides read and write access to the Primary [x_ <company-code>_<primary>] and Travel Request [x_ <company-code>_<travel_request>] tables.
App admin [x_<company-name>_ <app_name>.app_admin]	Provides complete access to the Service Request Management data and workspaces.
Fulfiller's manager [x_<company-name>_ <app_name>.fulfillers_manager]	Provides access to fulfiller's Manager workspace and provides read and write access to the Primary [x_ <company-code>_<primary>], Travel Request [x_ <company-code>_<travel_request>], and Fulfiller Task [x_ <company-code>_<fulfiller_task>] tables.
Requestor [x_<company-name>_ <app_name>.requestor]	Provides access to submit a request for default record producers from a service portal and provides read and write access to the Primary [x_ <company-code>_<primary>] and Travel Request [x_ <company-code>_<travel_request>] tables.
Fulfiller [x_<company-name>_ <app_name>.fulfiller]	Provides access to the fulfiller workspace and provides read and write access to the Primary [x_ <company-code>_<primary>], Travel Request [x_ <company-code>_<travel_request>], and Fulfiller Task [x_ <company-code>_<fulfiller_task>] tables.
admin [x_<company-name>_ <app_name>.admin]	The System Administrator role. This role has access to all system features, functions, and data, regardless of security constraints.

Team Contacts template

Build an application in App Engine Studio (AES) using the Team Contacts template to manage contact information for the people in your organization.

To use the Team Contacts template, install this application from the ServiceNow Store.



A team contacts app uses a mobile experience to view employee contact information, tag favorites, and add privacy notes. The app displays report information by default and lets an employee search for employees, view employee profiles, and create a list of favorite employee contacts.

For more information on creating an application using a template, see Create your app using an application template.

Team Contacts template contents

The following tables, experiences, automated workflows, and roles are included in the Team Contacts template for App Engine Studio (AES).

Tables in the Team Contacts app template

Table label [name]	Description
Relationship Contact Table [x_<company-code>_<team_contact>_relationship_contact]	Tracks employee contact information. For example, contact names, date created, favorites, and updates.
Reminder Table [x_<company-code>_<team_contact>_relationship_contact_reminders]	Tracks reminders created by your team. For example, when to send a reminder and the subject of a reminder.
Relationship Notes Table [x_<company-code>_<team_contact>_relationship_notes]	Tracks the notes that employees create for each other. For example, the employee name, reason for contact, HTML notes, and option to mark the contact as a favorite.

Experiences in the Team Contacts app template

Experience	Experience type	Description
Employees Mobile experience	Mobile experience	Provides employees access to an employee directory and relationship notes using your organization's iOS or Android app.

Experience	Experience type	Description
Team contact	Portal	Portal where employees and managers can view employee profiles, mark favorite contacts, and add relationship notes using a website. The team contacts portal includes a sign-in page, a home page, and a page to add relationship notes.
Note | Default view	Form	Form to add notes to contact information.
Relationship contact	Form	Form to view contact information.
Reminder | Portal	Form	Form for employees and managers to set reminders.
Reminder | Default view	Form	Form for employees and managers to set reminders.
Note | Portal	Form	Form for managers and employees to add notes to contacts.

Automated workflows in the Team Contacts app template

Workflow name	Workflow type	Description
Team Contact Reminder Push	Email	Sends a reminder push to employees.
Notify Contact Reminder	Flow - record	Sends a notification about a reminder that an employee sets up in a relationship note. The notification contains details from the relationship note.
Team Contact Reminder Notification	Email	Sends a reminder to employees.
Team Contact Reminder Ackn Notification	Email	Sends employees a reminder to acknowledge a notification.

Roles in the Team Contacts app template

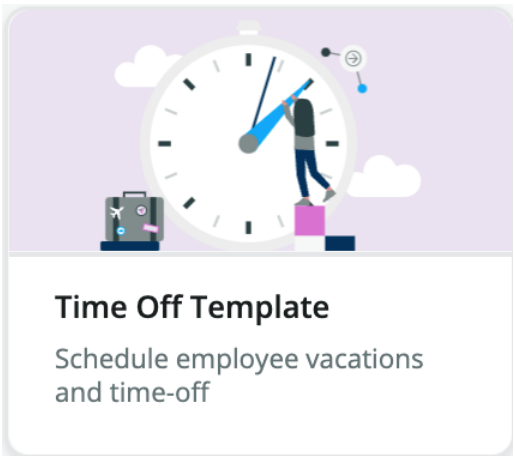
Role title [name]	Description	Contains roles
Manager [x_<company-name>_ <app-name>.manager]	Managers can view profile info, add relationship notes, and see their direct reports.	Create, Read (own), Update (own), Delete (own)

Role title [name]	Description	Contains roles
Admin [x_<company-name>_ <app-name>.admin]	Managers can configure the team contacts app and grant access to the app.	Read (all), Update (all), Delete
Employee [x_<company-name>_ <app-name>.employee]	Employees can view information on their team, search for employees, view employee profiles, and mark favorite contacts.	Create, Read (own), Update (own), Delete (own)
Canvas user	User role for Canvas Core application	

Time Off template

Build an application in App Engine Studio (AES) using the Time Off Requests template to schedule employee vacations and time-off.

To use the Time Off template, install this application from the ServiceNow Store.



Create a time off application to streamline the process for employees to request time off through a single mobile or web-based form. Managers are notified via email when a request is submitted. They can use a mobile app or a web-based portal to view historical and pending requests by month or year for an employee or their entire team. This allows the manager to make informed decisions, improve planning for upcoming work, or identify where shifts must be covered. Employees are automatically notified when their requests are approved or rejected and why. If plans change and updates to an approved request are required, the employee can edit and resubmit their requests for approval through the portal.

For more information on creating an application using a template, see Create your app using an application template.

Time Off template contents

The following tables, experiences, automated workflows, and roles are included in the Time Off template for App Engine Studio (AES).

Tables in the Time Off app template

Table label [name]	Description
Time Off Request Table [x_<company-code>_<my_time_off_request_app>_request]	Track time off requests that employees create using the Time Off Request app. For example, the employee name, creation date, manager, requested days off, and state of the time off request.

Experiences in the Time Off app template

Experience	Experience type	Description
Time Off Request | Default view	Form	Form for managers and admins to use as a template for time off requests.
Time Off Request | Workspace	Form	Workspace where managers can view, submit, and approve or reject time off requests. The workspace also contains historical records which can be viewed by team, employee, and time.
App name | Portal	Portal	Portal for managers and admins to view, submit, and approve or reject time off requests and for employees to view, edit, review, or cancel their time off requests.
App name | Workspace	Workspace	Workspace where managers and admins can use view, submit, and approve or reject time off requests.
Time Off Request | Record	Form	Form that managers can use to view, submit, and approve or reject time off requests.
Time Off Request | New	Form	Users can create new (save, submit, or cancel), update (save, submit, delete, or cancel), and review (approve or reject) time off requests.
Mobile app	Mobile experience	Employees can request time off and access their requests. Managers can submit, view, update, approve or reject time off requests using your organization's iOS or Android app.
Dashboard	Mobile experience	Managers and admins can view time off requests using

Experience	Experience type	Description
		your organization's iOS or Android app.

Automated workflows in the Time Off app template

Workflow name	Workflow type	Description
Time off request - Send comments	Flow - record	Sends a notification to an employee or manager when a comment has been added to a time off request.
Daily Time Off inactivating records	Flow	Updates the remaining time off on Time off request records until the remaining days is set to 0.
Time Off Req Approved	Email	Notifies an employee of an approved time off request.
Time Off Req Updated/ resubmitted	Email	Notifies managers of updated/ resubmitted time off requests.
Time Off Comments	Email	Notifies an employee or manager of a comment added to a time off request.
Time Off requests - Send notifications	Flow - record	Sends a notification to an employee or manager when a time off request state changes. The notification contains details from the time off request.
Time Off Request Cancelled	Email	Notifies an employee or manager of a cancelled time off request.
Time Off Request Rejected	Email	Notifies an employee of a rejected time off request.
Time Off Request - updated/ resubmitted	Flow - record	Sends a notification to an employee or manager that a time off request is updated/ resubmitted.
Time Off Req Submitted	Email	Notifies a manager when a time off request is submitted.
Time Off Request	Email	Employees can draft, request, approve, reject, and cancel time off requests.

Roles in the Time Off app template

Role title [name]	Description	Contains roles
Manager [x_<company-name>_ <my time_off_request_app>.manager]	Managers can submit, view, approve, and reject time off requests	Create, Read (all), Edit (all)
Employee [x_<company-name>_ <my time_off_request_app>.employee]	Employees can submit, view, edit, and cancel time off requests	Create, Read (own), Edit (own), Delete (own)
Admin [x_<company-name>_ <my time_off_request_app>.admin]	Admins can manage the time off app and view all records	Create, Read (all), Edit (all), Delete (all)
Public	Public users require no login to access features or functions within the application.	Role contains no access by default.

Build a custom template

App Engine Studio (AES) provides various predefined application templates. However, creating custom templates can speed up time to production if you have types of applications that you want to create more than once.

You can create a custom template from scratch or use existing content as a starting point. If one of the predefined AES templates is close to what you need, consider extending that template and then customizing it for your needs. Complete your template by adding and customizing data, experiences, logic and automation, and security roles for access.

Build a custom template from an existing application

Use an existing application as a starting point to build a custom template in App Engine Studio (AES). Existing data, experience, logic and automation, and security roles from the application are automatically added to the custom template.

Before you begin

Role required: admin, app_template_author

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the Template page, select **Create new template**.
3. Select **Existing app**, and then select **Continue**.
4. On the app selection screen, select an existing application for the custom template creation, and then select **Continue**.
   The following outcomes are possible.

- All items are supported: The custom template is created. All the objects are available in the custom template and in apps created from the custom template.
- Some items aren't supported: You can continue to create the template but a few items are not available in the custom template. There's no impact on the custom template or the apps created from the template.
- Some items are denied: You encounter an error message and cannot continue further until the errors are fixed. The author must change the app, if it was created from an existing app or the template, if it was created from scratch.

Note:
For more information, see Supported features and metadata in custom templates.

5. Enter a name and description for the custom template.
6. Accept or change the application's logo.
   - If the existing application has a logo in SVG format, the same logo is applied to the custom template.
   - If there's no logo or the logo is in a format other than SVG, upload an SVG logo for the custom template by either dragging the image to the **Browse or drag an SVG image** field or selecting the field, selecting the image from your file directory, and selecting **Open**.
7. Select **Continue**.
8. Specify which users or groups should have access to this template.

Option	Description
Share with specific users and groups	**a.** From the **Shared with** list, select **Specific users and groups**. **b.** Add users individually or in groups. **c.** Select **Give access**. **d.** Select **Continue**.
Share with all users and groups	**a.** From the **Shared with** list, select **All users and groups**. **b.** Select **Continue**.

For more information about template sharing and permissions, see App template sharing.

9. Access the custom template in one of the following ways.

Result
Your custom template is created and ready for your use. The custom template is activated and is available to App Engine Studio users who have access to it.

Build a custom template from scratch

Build your custom template from scratch if none of the predefined App Engine Studio (AES) templates fit your business needs or you don't have an existing application as a starting point.

Before you begin
Role required: admin, app_template_author

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the Template page, select **Create new template**.
3. Select **Start from scratch**, and then select **Continue**.
4. Enter a name and description for the custom template.
5. **Optional:** Add a logo to the custom template by either dragging the image to the **Browse or drag an SVG image** field or selecting the field, selecting the image from your file directory, and selecting **Open**.
6. Select **Continue**.
   A progress message indicates when the empty custom template has been created.
7. Specify which users or groups should have access to this template.

Option	Description
Share with specific users and groups	**a.** From the **Shared with** list, select **Specific users and groups**. **b.** Add users individually or in groups. **c.** Select **Give access**. **d.** Select **Continue**.
Share with all users and groups	**a.** From the **Shared with** list, select **All users and groups**. **b.** Select **Continue**.

   For more information about template sharing and permissions, see App template sharing.
8. On the template details screen, select **Go to template dashboard**.
9. Add data, experience, logic and automation, and security measures to the custom template. For more information, see the following topics:
   - Create a data model for your application
   - Add an application experience
   - Add logic and automation
   - Add application security
10. Select **Save**.

What to do next

Provide an overview of the custom template and activate the template to make it available to App Engine Studio users through the template library. For more information, see Update custom template contents and properties.

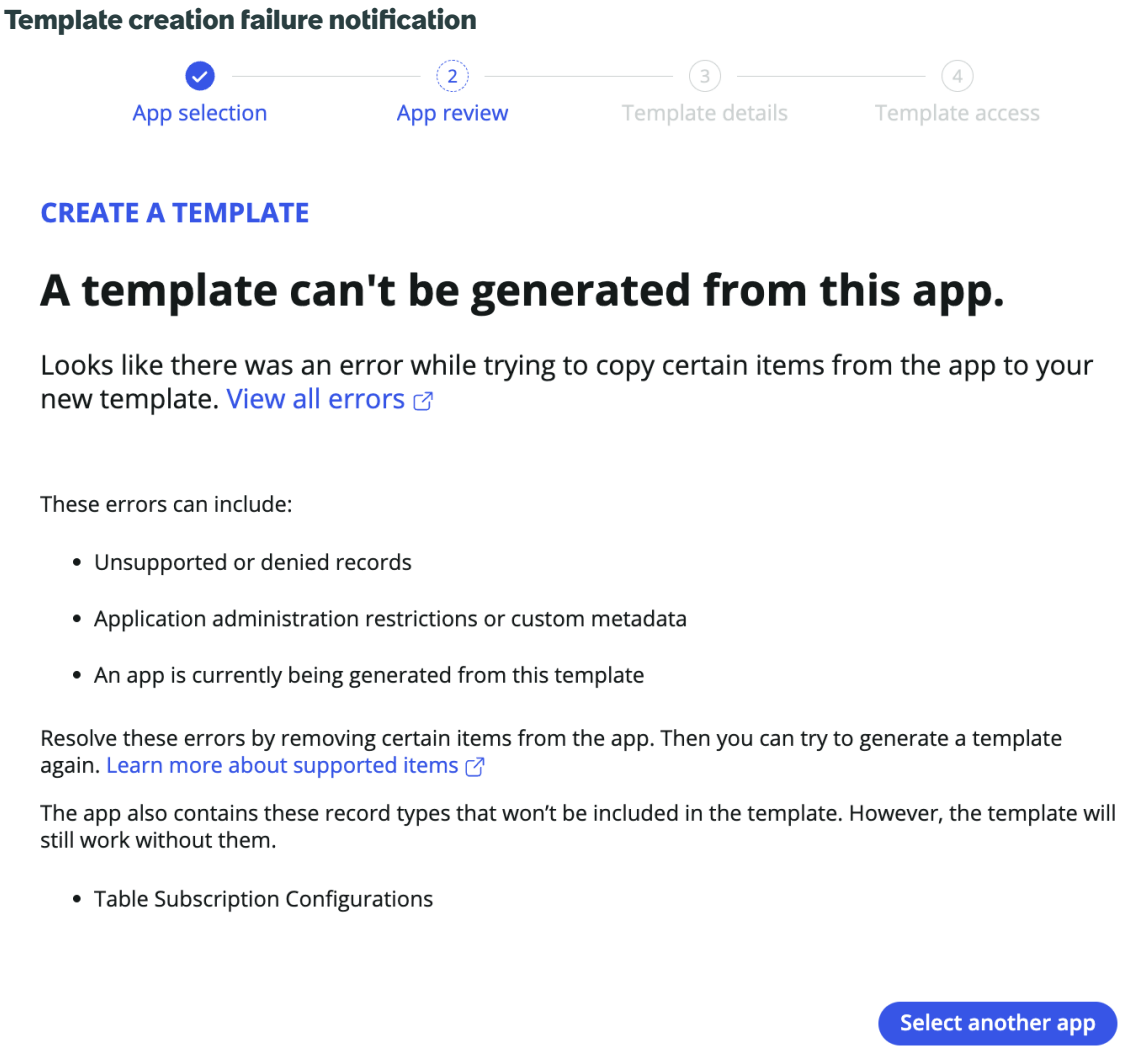
Identifying and fixing app template creation errors

Learn how to identify and fix your app when unsupported features or metadata stop the template creation process in App Engine Studio (AES).

Unsupported features

Sometimes items in a source application from which you created a custom template aren't supported in the template. You must remove unsupported items before you can create a template from the application.

When an error happens during the template creation process, an error message is displayed.



You can identify which elements of your application are causing it to fail the template creation process by selecting **View all errors**, which displays the Scan Failure [sys_app_scan_failure] table. For more information about the errors in the table and their cause, see Template creation errors in the Scan Failure table.

Template creation errors in the Scan Failure table

Errors that occurred during the creation of a custom template that are displayed in the Scan Failure [sys_app_scan_failure] table and their cause.

Error messages

For more information about what types of records are allowed and skipped, see Supported features and metadata in custom templates.

Error message	Definition	More details
Unsupported_table	Record class for which there is no allow rule, and there's an unknown level of support.	One or more unsupported records are unrecognized by App Engine Studio and must

Error message	Definition	More details
		be removed for the template creation process to continue.
Denied_table	Record class for which there is a deny rule, and template creation isn't supported.	The record class is explicitly denied and must be removed for the template creation process to continue.
Scripted_record	Scripted rule that runs on each record in the app.	Current scripted deny rules include, but aren't limited to: • Custom metadata: The app has tables that extend the sys_metadata table. • Scoped administration: The app has scoped administration turned on. • Template execution in progress: The template associated with the source app is currently being executed.
Scripted_app	Scripted rule that runs once for the app.	

Assign permissions to publish custom templates

Assign permissions for developers to publish a custom template to the application repository or the ServiceNow Store. You can grant permissions to publish either a specific custom template or publish all existing custom templates in App Engine Studio (AES).

Assign permissions to publish an individual custom template

Assign permissions to developers so they can publish an individual App Engine Studio (AES) custom template to the application repository or ServiceNow Store.

Before you begin
Role required: admin

Procedure

1. Navigate to **All** > **App Engine** > **Templates** > **App Templates**.
2. Select the personalize List icon (⚙).
3. Add the Package column by moving the **Package** value from the **Available** list to the **Selected** list and select **OK**.
4. From the Package column, select the package link for your custom template.
5. Under the **Related links** heading, select **Manage Collaborators**.
6. Assign permissions to new or existing collaborators.

Result
The specified user or group now has permission to publish an individual custom template.

Assign permissions to publish custom templates and apps

Assign permissions for developers to publish custom App Engine Studio (AES) templates and applications to the app repository or ServiceNow Store.

Before you begin
Role required: admin

Procedure

1. Navigate to **All** > **App Engine** > **Collaboration** > **Descriptors**.
2. In the **Name** field of the All Collaboration Descriptors page, search for and select **Owner**.
3. From the **Development Permission Sets** tab, select **Edit**.
4. Move the **Publish To App Repo** and **Publish To App Store** values from the **Collection** list to the **Development Permission sets** list.
5. Select **Save**.

Result

Developers who own applications now have permissions at a platform level to publish all custom templates to their application repository or ServiceNow Store.

Publish a custom template

You can publish a custom template that is built from scratch or from an existing application using App Engine Studio (AES).

Before you begin

You must have permissions to publish a custom template. For details, see Assign permissions to publish custom templates.

Role required: admin, app_template_author

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio** > **Templates**.
2. Point to the custom template, select the More actions icon (•••), and select **Properties**.
3. On the **General** tab, select the **Activate this template** check box.
4. Select **Publish template**.
5. **Optional:** For a custom template built from scratch, select **Publish anyway**.

   (Optional) App Engine Studio displays the publish template fields.
6. Fill in the fields.

Provide a version number and release notes

Field	Description
Version	Provide a new version number for the custom template.
Release notes	Provide a brief description of the changes made to the custom template.
Publish options	Select where you want the custom template published. ○ To publish the template to your application repository, select **My application repository (Make the app available to my instances)**. ○ To publish the template to the ServiceNow Store, select **ServiceNow Store (Submit this app to the ServiceNow Store for review)**.

7. Select **Continue**.
8. On the confirmation screen, select **Close**.

Result
The custom template is published and available to users the app template admin shares it with.

Update custom template contents and properties

Improve the effectiveness of your custom template by updating its contents or properties in App Engine Studio (AES).

Before you begin
Role required: admin, app_template_author

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. Select the **Template** tab.
3. Modify different aspects of the template to make them more useful to your needs.

Manage template activation

Manage template activation and deactivation to ensure that users creating apps in App Engine Studio (AES) have access to the best application templates for their needs.

Before you begin
Role required: app_template_admin

About this task
App template admins manage the templates that are active in non-production instances and determine which users can see and use those templates. Admins may want to deactivate custom templates that deviate from company standards or don't suit the needs of their users.

Procedure

1. Navigate to **All** > **App Engine** > **Templates** > **App Templates**.
2. Select the template that you want to activate or deactivate.
3. To edit the template, select **here** in the message that appears.
4. Activate or deactivate the template.
   - To activate the template, select the **Active** check box.
   - To deactivate the template, clear the **Active** check box.
5. Select **Update**.

App template sharing

Specify which users and groups have access to predefined and custom templates in App Engine Studio (AES).

Implicit sharing

Templates are implicitly shared with some users in your organization, meaning that no sharing record exists that determines sharing. The three rules governing implicit sharing are:

- Instance administrators can see and use all templates.
- App template admins (users with the app_template_admin role) can also see and use all templates.
- App template authors (users with the app_template_author role) can see templates that are not shared with them if they have permission to edit the scope in which the template resides.

These sharing rules apply regardless of whether a template is active.

Explicit sharing

You can share app templates with anyone in your organization by creating a sharing record for that user or group. However, regardless of sharing permissions, the user or group must have the sn_app_eng_studio.user role to see and use the template. The three ways to share templates are:

- Share globally with every user in your organization
- Share with a specified user group
- Share with a specified single user

Note:
If you stop sharing a template with an individual user but that user is a part of any group the template is shared with, that user is still able to see and use the template.

Manage template sharing permissions

Share App Engine Studio (AES) templates globally, with groups, or with individual users to ensure users have access to the right templates.

Before you begin
Role required: app_template_admin

Procedure

1. Navigate to **All** > **App Engine** > **Templates** > **App Templates**.
2. Select the template you want to modify sharing permissions for.
3. In the Template Sharing Permissions related list, select **New**.
4. Select a sharing option.
5. Select **Submit**.

Create your application from scratch

If the available application templates in App Engine Studio (AES) don't fit your application goal, create an empty application to which you can add data, experience, logic and automation, and security.

Before you begin

Note:
To create apps in AES, you must be an admin or in the App Engine Studio Users group. If you are in the App Engine Studio User Limited group, you can only edit existing apps, not create new ones.

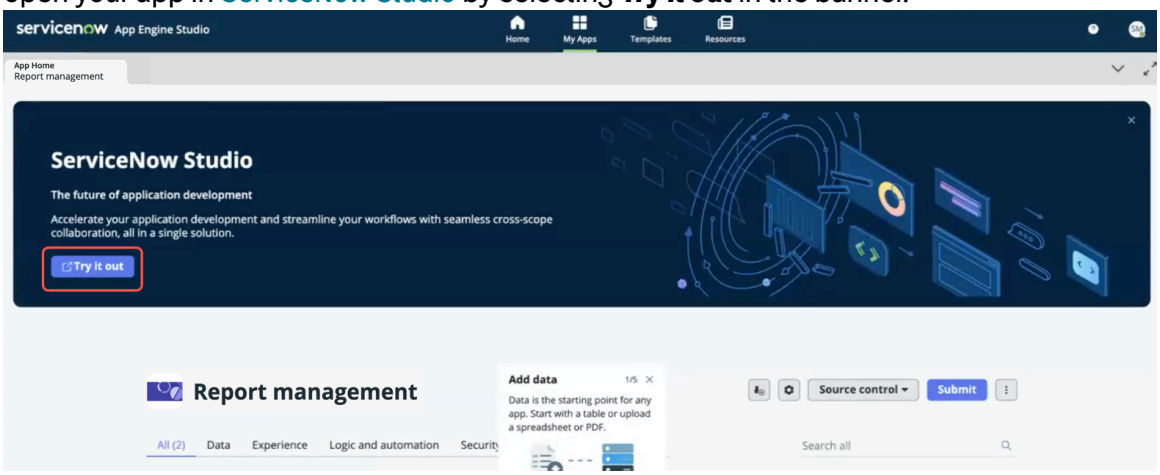
Role required: admin, sn_app_eng_studio.user

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. Select **Create app**.
3. Enter a **Name** and **Description** for your application.
4. **Optional:** Add a logo by either dragging the image to the **Drag app logo or browse to upload** field or selecting the field, selecting the image from your file directory, and selecting **Open**.
5. Select **Continue**.
6. Control who has access to create, read, write, and delete content in your application by adding or customizing roles.
   To add security roles to your app, select **Add a role**, and enter a name and description for the role.

   Note:
   At least one role must have **Read** access to be able to update application security.

7. Select **Continue**.
8. On the summary screen, select **Go to app dashboard**.
9. **Optional:** If you want a more advanced app development environment to work in, you can open your app in ServiceNow Studio by selecting **Try it out** in the banner.



10. **Optional:** Review the guiding information for each section of your application by enabling the **Guidance** toggle ( ) in the bottom left corner of the window.
11. Continue building your application by adding data, experiences, logic and automation, and security to your app.
    For more information, see the following topics:
    - Create a data model for your application
    - Add an application experience
    - Add logic and automation
    - Add application security

What to do next

After you've finished building your application, submit the application for approval to get it reviewed and deployed by an administrator. For more information, see Submit your app for approval and publishing.

Prepare your app for approval

Prepare your app for the approval process by making any needed changes, such as renaming your app, editing the app description, or replacing the app image. You can also delete the app from App Engine Studio (AES) if it has taken a wrong turn.

Before you begin

Role required: admin, sn_app_eng_studio.user

About this task

You can change certain features of your application, based on your permissions. To request delete permissions, contact your administrator.

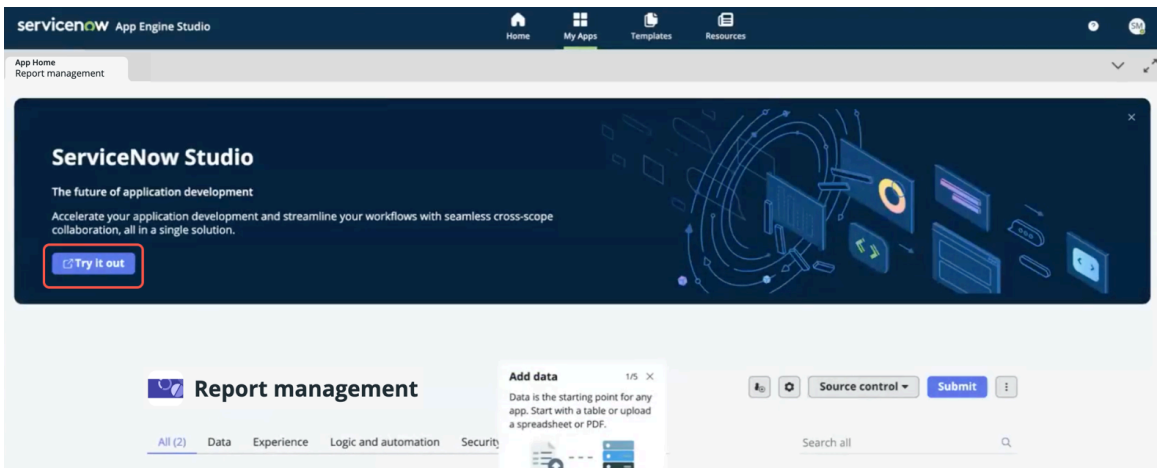
Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. From the app home, select the Edit application properties icon ( ).
4. Update the application properties.
   - To rename the application, update the **Name** field.
   - To rewrite the application description, update the **Description** field.
   - To replace an image, select **Remove image** and then add a new image.
   - To delete your application, select **Delete application**, type `Delete` to confirm your choice, and select **Delete**.
5. Select **Save**.

Enhance your app

Whether you use a template or create your app from scratch in App Engine Studio (AES), you must enhance it by creating a data model, user experiences, logic and automation, and security.

If you're creating apps in App Engine Studio (AES) and want a more advanced app development environment to work in, you can open your app in ServiceNow Studio. When you open your app in AES, a banner appears with information about ServiceNow Studio. Select **Try it out** to open your app in ServiceNow Studio.



Use the following topics as guides as you build out your application.

Resources for enhancing your app

As you add data, experience, logic and automation, and security to your application, you might interact with several different parts of the ServiceNow AI Platform. The links in the following table point you to resources for adding content in App Engine Studio and also Table Builder, UI Builder, Decision Builder, App Engine Management Center, and the ServiceNow AI Platform.

Resources for adding content to your application

Add data	Add an experience	Add logic and automation	Add security
Add data to your application in App Engine Studio. • Create a data model for your application • Create a blank table • Use a spreadsheet to add data • Use a PDF to create data tables • Create a data integration • Modify application data tables	Work in App Engine Studio to add experiences to your application. • Add an application experience • Add a record producer • Add a standard catalog item • Add a workspace • Add a portal • Add a mobile experience • Editing an experience in App Engine Studio	Build logic and automation into your application in App Engine Studio. • Add logic and automation • Add a pre-built flow using a template • Add a flow from scratch • Add a decision • Add an email notification • Edit a process	Configure roles to manage your application's security in App Engine Studio. • Add application security • Build a new role for your application • Use an existing role for your application • Collaborate with other developers • Change access settings for a role • Delete a role
Seamlessly transition to using Table Builder to enhance your application's data. • Exploring Table Builder • Data in Table Builder • Forms in Table Builder • Policies and rules in Table Builder • Flows in Table Builder	Use Workspace Builder and UI Builder to further enhance experiences in your application. • Add a workspace • Configure a workspace in Workspace Builder • UI Builder and configurable workspaces • Configure how users interact with your applications in UI Builder • Customize UI Builder pages using components	Use Decision Builder to easily manage business logic for your application. • Create decision tables in Workflow Studio • Manage decision tables in Excel • Modify decision table structure in Workflow Studio • Modify decision table rules in Workflow Studio	Configure and manage roles for your organization and application using the ServiceNow AI Platform and App Engine Management Center. • Configure AES personas and roles • Configure users and groups • Add users to the App Engine Admin group • Delegate developers using AES

Resources for adding content to your application (continued)

Add data	Add an experience	Add logic and automation	Add security
			• Managing developers using AEMC • Manage collaboration requests

Create a data model for your application

Create a data model in App Engine Studio (AES) for your application by creating the tables that will house your application data. Create data tables that will be used in your application by creating the table from scratch, uploading a spreadsheet or PDF, or by using an existing table as a template. You can also create and schedule data imports by mapping a spreadsheet to an existing table in your application.

Creating a data model and populating data into user interfaces are crucial starting points for any new application. Users may provide data, such as their name and phone number, when they fill in a form, and other users may refer to data as they fulfill a request.

In App Engine Studio, application data is stored in table format. When users update application data, they create a row or change an existing row in the table (each row is also known as a data record). You can create tables from existing ServiceNow AI Platform tables, from a spreadsheet or PDF, or customize a completely new table for your application to store records in.

The data records stored in your application may come from several sources. One table might store data records entered by users who are using your application, another might be populated via the import and mapping of data from a spreadsheet or through a script that updates data records with data from another ServiceNow AI Platform table.

Application templates and data

Application templates automatically add data to your application. If you use a template to create your application, you can edit the tables that were added or add different tables.

Forms

When you add data to your app, any associated form views display in the **Experience** section of your app. To edit the form, select it in the **Experience** section. Form views can be edited in the **Forms** tab of Table Builder. For more information, see Forms in Table Builder.

Table extension

There are several tables that your organization may already be using in other applications. For example, your organization may be using the Configuration item [cmdb_ci] table for an application that houses configuration data. You can create a table for your application by extending any of these existing tables.

Table extension allows a table to share fields and records with a parent table. You might extend a table if you expect users to fill in similar fields across different form views in an application. For example, to create a ticketing type of table, you might extend the Task [task] table. The Task table includes fields that are standard for most work tickets, such as **Number** and **Assigned to**.

Many organizations create extensible standardized tables for their citizen developers to use when designing their apps. This enables the developers to standardize the way data is utilized across multiple applications within the organization.

To make a table in your application extensible, select **Make extensible** during table creation. After extending an existing table, you can further customize the new table by adding more columns.

The following table provides a list of commonly extended tables in the ServiceNow AI Platform. For more information on commonly extended tables and models for extension, see Table extension and classes.

Commonly extended tables

Table	Short Description	Description
task	Task	Stores fields for the core applications such as Incident, Problem, and Change Management. It provides a series of standard fields used on each of the tables that extend it. Any table which extends the task table can take advantage of task-specific functionality for driving tasks.
cmdb_ci	Configuration Item	Stores configuration items. This table can be extended for configuration items such as hardware, services, etc.
sn_customerservice_case	Customer Service Case	Stores customer service case records.
sm_order	Service Order	Stores data that defines and manages work that needs to be performed.
sm_task	Service Task	Stores units of work performed by one person in one session (one location, one time).
planned_task	Planned Task	Stores additional fields for tasks pertaining to time and effort as part of a planned, multi-stage process.
cmn_location	Location	Stores location information.
cmdb_serviceorder_product_model	Service Order Model	Stores service order templates.
cmdb_servicetask	Service Task Model	Stores service task templates.

Commonly extended tables (continued)

Table	Short Description	Description
_product_mode		
cmdb_qb_result _base	Query Results	Stores query results created by the CMDB Query Builder.
alm_asset	Asset	Stores general, financial, and contractual information about assets.
cmdb_ci_service	Business Service	Stores IT Services data that directly supports a business process.
sys_user	User	Stores user data. The User table provides a list of all system users and their related department.
sys_user_group	User Group	Stores and groups a set of users who share a common purpose. Groups may perform tasks such as approving change requests, resolving incidents, receiving email notifications, or performing work order tasks.
core_company	Company	Stores company information.
cmn_schedule	Schedule	Stores records that specify a time zone and a type of schedule and use one or more schedule entries.
cmn_department	Department	Stores a list of all departments and their related business unit.
cmn_cost_center	Cost Center	Stores cost center records. This is a reference between financial systems and IT. Cost center records represent business entities, and have a related list of CI Cost Center Relationships that measure the cost center's consumption of business services.
cmdb_model	Product Model	Specific versions or various configurations of an asset. Models are used for managing and tracking assets through various ServiceNow platform asset applications, including Product Catalog,

Commonly extended tables (continued)

Table	Short Description	Description
		Asset Management, and Procurement.
life_cycle_stage	Life Cycle Stage	Stores standard fields and values for tracking life cycle stages for CIs. Using these standard values consistently across applications helps to effectively track assets through their life cycle transitions.
life_cycle_stage _status	Life Cycle Status	Stores standard fields and values for tracking life cycle stage status for CIs. Using these standard values consistently across applications helps to effectively track assets through their life cycle transitions.
incident	Incident	Stores deviations from an expected standard of operation.
sysapproval	Approval	Stores data that allows you to require authorization on tasks before the work is done. You can define approvals for all tasks and associate users or groups to a task to approve or reject them.

Use the methods listed below to create a data model and add data to your application.

Create a blank table

Add a data table to your application from an existing table that you or someone else previously created or from scratch.

Before you begin

Role required: admin, sn_app_eng_studio.user, or delegated_developer

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the Data heading, select the Add icon (⊕).
4. Select **Create a blank table**, and then select **Continue**.
5. Choose from the following choices.

Option	Description
Create new table	Select this option to create a blank table from scratch. **a.** Choose **Create new table**. **b.** Click **Continue**.
Create from an extensible table	Select this option to create a new blank table from an existing extensible table in the Servi ceNow AI Platform. **a.** Choose **Create from an extensible table**. **b.** Search for and select an existing table. **Note:** Tables already in the app display first in the list and recommended Service Now AI Platform tables display next, along with a short description of the type of information the table stores. **c.** Select **Continue**.

6. On the form, fill in the properties of your new table.

Field	Description
Table label	Unique label to identify the table.
Table name	Database name for the table. A table name is created automatically after you enter a table label. You can edit the name if needed.
Make extensible	Option to allow other tables to share data from this table. For more information on table extension, see Table extension.
Auto number	Option to track table records with a unique number. If you select this option, define the **Prefix**, **Starting number**, and **Number of digits**.
Prefix	Abbreviated name of the table to append to the beginning of the record number. For example, if you are creating a "Laptop" table, then your prefix may be "LPTP" or "LT."
Starting number	Number to identify the first record created for your table.
Number of digits	Maximum number of digits to allow in the record number. This value determines the highest possible record number. For example, if you enter 7, then the highest possible number is 9999999.

7. Select **Continue**.
8. Control who has access to create, read, write, and delete content from this table by adding new roles and/or defining the desired Create, Read, Write, Delete permissions for existing roles.

Roles that you create in the table persist throughout the application and can be further defined in other tables you add. See Add application security for more information on how security permissions work.

Note:
At least one role must have read access for you to be able to preview the table.

9. Select **Continue** to add the new table.
10. Return to your application home page or continue editing your table.
    - Select **Done** to return to your application home page.
    - Select **Edit table** to review the columns that were added to your table and continue editing it in Table Builder.
11. If you continue editing, in the new tab that opens, review the table columns.

    Each table column appears as a row in Table Builder.

    A lock icon ( ) appears next to columns from the extended table. You can't modify these columns.
12. Customize your table by adding table columns.

    a. In Table Builder, select **+ Add new field**.

    b. On the empty row, fill in the fields.

Table column properties

Field	Description
Column label	Unique label for the column.
Column name	Database name for the column.
Type	Type of information that the column contains. For example, to contain plain text in the column, select **String**. Depending on the type that you select, fill in the additional fields to further define the table column. For example, if you select **String**, define the character limit of the string input. Or, if you select **Choice**, define the choices that users can choose from. See Field types.
Reference	Table that is associated with the column. This field applies only if the column type is **Reference**.
Max length	Maximum number of characters that users can enter in the field.
Default value	Value that populates the field automatically after a new record is created.
Display	Option to set the column as the display value for the table. A reference field shows the display value of the table to which it is referring. For example, the **Opened by** column of the task table refers to the user table. Because the display value of the user table

Field	Description
	is the user name, the **Opened by** field shows something like **Beth Anglin** or **Joe Employee**. When you select a display value, choose the table column that would act as an appropriate title for individual records. Only one column can act as the display value for a table.

13. Select **Save**.

What to do next
For more information on Table Builder, see Table Builder.

Use a spreadsheet to add data

Add data tables and records to your application by uploading a Microsoft Excel spreadsheet.

Use the following procedures to upload a spreadsheet and use it to create data tables and import data records. Use the Import a spreadsheet procedure first to upload your spreadsheet, and then create data tables from your spreadsheet import as needed.

Import a spreadsheet

Upload a Microsoft Excel spreadsheet and use it to create your application data model.

Before you begin
Check that the spreadsheet meets the following requirements:

- Formatted with horizontal columns and a header label for each
- Saved as an XLSX file type

Note:
The Excel spreadsheet you upload can have multiple worksheet tabs, each of which you might use to create a data table. If multiple worksheets are present, you will be prompted to select which worksheet tab to create the table from after you upload the file.

Role required: admin, sn_app_eng_studio.user or delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the Data heading, select the Add icon ( ).
4. Select **Import a spreadsheet**, and then select **Continue**.
5. Upload your spreadsheet and then select **Continue**.

   You can drag the spreadsheet file onto App Engine Studio or browse your computer.
6. You will be presented with one of the following paths:
   - If your spreadsheet only has a single worksheet tab, proceed to the next step.
   - If the spreadsheet has multiple worksheet tabs, you will have a list of sheets to choose from. From this list of sheets, you can select which worksheet tab to add as a table to your application first.

a. In the **Enter a row number for the table header** field, enter the row number in which the sheet column headers are located.
For example, if the headers are in the second row, enter 2.

b. **Optional:** To upload the information below the column headers in the sheet (e.g., the data in your spreadsheet), select the **Import spreadsheet data** option.
To upload only the column headers, leave the check box cleared. You may want to upload only the headers if you intend to use this table as a template and don't want any of the other data added to your AES table.

7. Make one of the following selections.
   - If you have uploaded an Excel file with multiple worksheet tabs, select **Convert to table** to proceed with converting the selected worksheet tab to a table, and then proceed to next step.
   - If you have uploaded an Excel file with a single worksheet tab, select **Continue**.

   Options display for where to import your file into.
8. Based on what you would like to do with your spreadsheet data, navigate to one of the following procedures to continue.

Option	Description
Create a new table directly from the spread sheet import	Refer to Create new table from spreadsheet import for next steps.
Create a new table by extending an existing table, and then using the spreadsheet im port to update it	Refer to Create new table from extensible ta ble and spreadsheet import for next steps.
Use spreadsheet import to update an exist ing application table	Refer to Modify existing table using spread sheet import for next steps.

Create new table from spreadsheet import

Create new tables directly from a Microsoft Excel spreadsheet import.

Before you begin

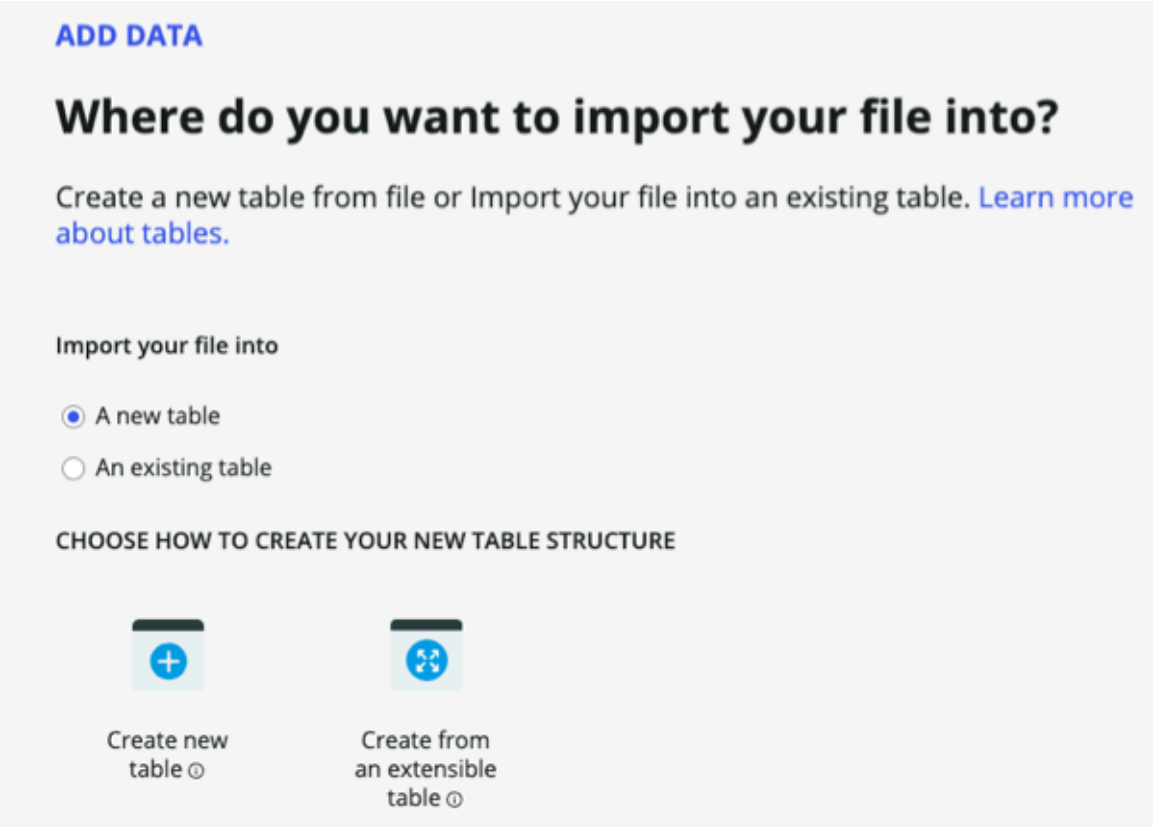
- Import a spreadsheet into App Engine Studio. See Import a spreadsheet.

Role required: admin, sn_app_eng_studio.user or delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. After you have uploaded your spreadsheet as described in Import a spreadsheet, choose what you want to do with the imported data in the displayed wizard.

Create new table from spreadsheet



   a. Select **A new table**.

   b. Choose **Create new table**.

2. Select **Continue**.

3. Review the properties for each column header, using the **Expand options** drop-down to display more properties.

Column header property	Description
Field Label	Unique label for the column.
Field name	Database name for the column.
Type	Type of information that the column contains. For example, to contain plain text in the column, select **String**. Depending on the type that you select, fill in the additional fields to further define the table column. For example, if you select **String**, define the character limit of the string input. Or, if you select **Choice**, define the choices that users can choose from. See Field types.

Column header property	Description
Character limit	Max length of the strings that can be stored.
Display	Option to set the column as the display value for the table. A reference field shows the display value of the table to which it is referring. For example, the **Opened by** column of the task table refers to the user table. Because the display value of the user table is the user name, the **Opened by** field shows something like **Beth Anglin** or **Joe Employee**. When you select a display value, choose the table column that would act as an appropriate title for individual records. Only one column can act as the display value for a table.
Mandatory	Option to require that the column must contain a value before a new record can be saved.

To add another column to your table, select **Add new field**.

To delete a column, select the trash icon (🗑).

4. Select **Continue** to define table properties.
5. On the form, fill in the fields.

Field	Description
Table label	Unique label to identify the table.
Table name	Database name for the table. A table name is created automatically after you enter a table label. You can edit the name if needed.
Make extensible	Option to allow other tables to share data from this table. For more information on table extension, see Table extension.
Auto number	Option to track table records with a unique number. If you select this option, define the **Prefix**, **Starting number**, and **Number of digits**.
Prefix	Abbreviated name of the table to append to the beginning of the record number. For example, if you are creating a "Laptop" table, then your prefix may be "LPTP" or "LT."
Starting number	Number to identify the first record created for your table.
Number of digits	Maximum number of digits to allow in the record number. This value determines the highest possible record number. For example, if you enter 7, then the highest possible number is 9999999.

6. Select **Continue**.
7. Control who has access to create, read, write, and delete content from this table by adding new roles and/or defining the desired Create, Read, Write, Delete permissions for existing roles.

Roles that you create in the table persist throughout the application and can be further defined in other tables you add. See Add application security for more information on how security permissions work.

Note:
At least one role must have read access for you to be able to preview the table.

8. Select **Continue** to add the new table created from your spreadsheet to your app.
9. **Optional:** Convert any remaining worksheets if your file contained multiple worksheet tabs.
   - **Note:**
     For each sheet that you want to convert, you will need to select **Convert to table**, and then repeat steps in this procedure or choose a different pathway for the additional worksheet (see Create new table from extensible table and spreadsheet import or Modify existing table using spreadsheet import for steps).
   - If you do not have multiple worksheets, proceed to the next step.
10. Select **Done** on the summary screen.
    Alternatively, to view your table in Table Builder, you can select **Edit table(s)** on the summary screen.

What to do next

For more information on editing your data tables in Table Builder, see Table Builder.

Create new table from extensible table and spreadsheet import

Create a new table by extending an existing table, and then using a Microsoft Excel spreadsheet import to update or customize it.

Before you begin

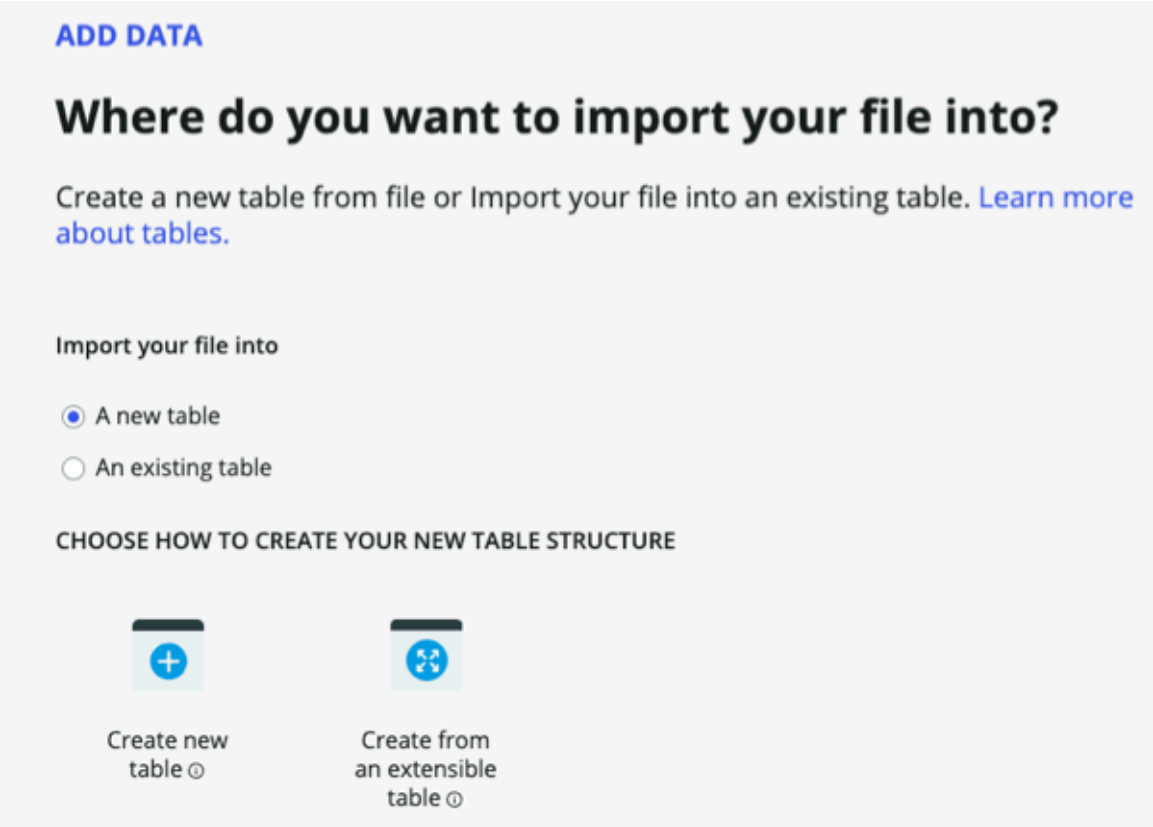
- Import a spreadsheet into App Engine Studio. See Import a spreadsheet.

Role required: admin, sn_app_eng_studio.user, delegated developer permissions for "Integration" must be present to view data import and mapping options. For more information, see Delegate developers using AES.

Procedure

1. After you have uploaded your spreadsheet as described in Import a spreadsheet, choose where to import your file in the displayed wizard.

Import into a new data table



   a. Select **A new table**.

   b. Choose **Create from an extensible table**.

2. Select **Continue**.
3. Search for an select an existing **Table** to add that table's data to your app and create an extension with branching logic.

> **Note:**
> Tables already in the app display first in the list and recommended ServiceNow AI Platform tables display next, along with a short description of the type of information the table stores.

4. Select **Continue**.
5. On the form, fill in the fields.

Field	Description
Table label	Unique label to identify the table.
Table name	Database name for the table. A table name is created automatically after you enter a table label. You can edit the name if needed.
Make extensible	Option to allow other tables to share data from this table. For more information on table extension, see Table extension.

Field	Description
Auto number	Option to track table records with a unique number. If you select this option, define the **Prefix**, **Starting number**, and **Number of digits**.
Prefix	Abbreviated name of the table to append to the beginning of the record number. For example, if you are creating a "Laptop" table, then your prefix may be "LPTP" or "LT."
Starting number	Number to identify the first record created for your table.
Number of digits	Maximum number of digits to allow in the record number. This value determines the highest possible record number. For example, if you enter 7, then the highest possible number is 9999999.

6. Select **Continue**.
7. Control who has access to create, read, write, and delete content from this table by adding new roles and/or defining the desired Create, Read, Write, Delete permissions for existing roles.

   Roles that you create in the table persist throughout the application and can be further defined in other tables you add. See Add application security for more information on how security permissions work.

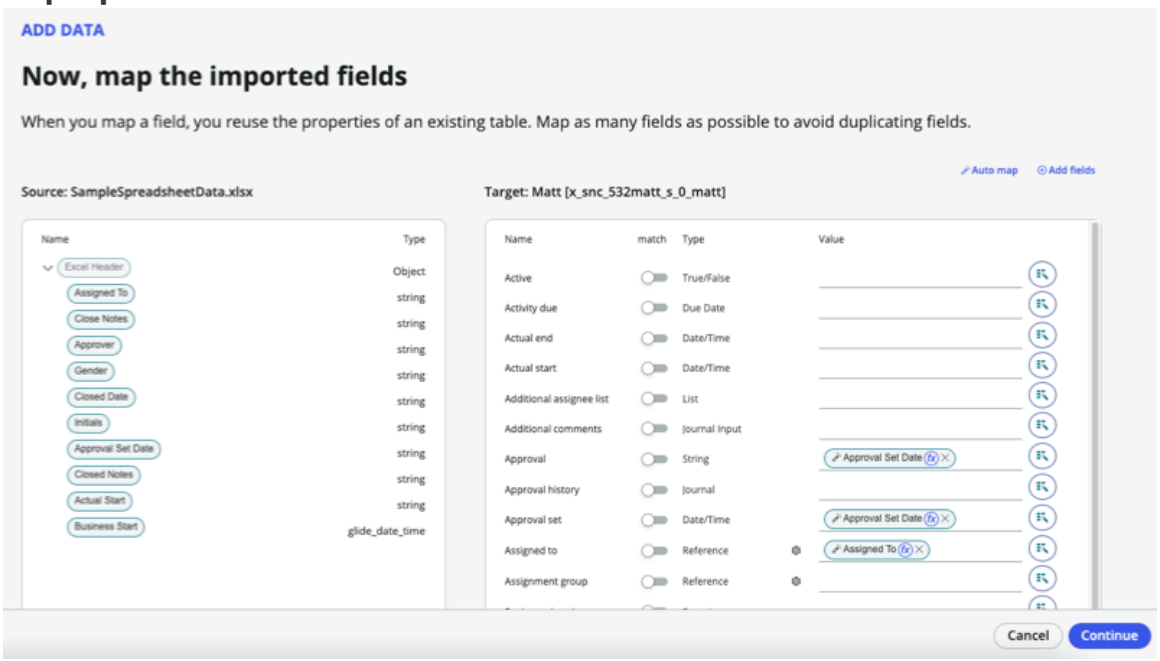
   Note:
   At least one role must have read access for you to be able to preview the table.

8. Select **Continue**.

   Note:
   Now your table is ready for data. In the next step, fields from your import are matched with fields on your newly extended table. You will have an opportunity to adjust mapping or create new fields from your import in it.

9. Select **Continue** to proceed to mapping fields.
   A screen displays where you can map field columns from your source spreadsheet on the left to field columns in your newly extended target table that you just created on the right.

Map imported fields



10. You have the following choices available to you during this step.

Option	Description
Batch map fields using Automap	Select **Automap** to automatically map your spreadsheet fields to displayed fields in the selected target table. Automap maps source entities to similarly named target entities. For a target entity that has already been mapped, selecting **Automap** doesn't change its value. An automapped field has a wand icon on its data pill.
Map fields by dragging and dropping	Map fields from your import to your target table by dragging data pills representing columns in the imported spreadsheet to the **Value** column in the target table. You can also map fields by using the pill picker to the right of each target table field.
Map fields using the data picker	Map data by using the pill picker ( ) on the right side of the target table. You can enter more than one value in the **Value** column, you can also enter text.
Add transform options	Add transform options for each of the individ ual mapped spreadsheet fields by clicking the FX icon ( ) on the right side pane. For example, you can use the Uppercase trans form function to change an input string to all uppercase characters. See Transform func or information on how to use these

Option	Description
	to trim data for your imported spreadsheet fields. **Note:** Not all the transform functions available in Flow Designer are available.
Activate data record matching	Specify fields where the system should check for matching data and, if found, update existing records instead of creating new ones by enabling the **Match** toggle in your target table. For example, if a target table has an **Order number** field and a record with an order number of 743, there are two options for handling potential matches. ○ If the **Match** toggle switch for the **Order number** field is on, the system checks for matching data. If the source data has a record with the same order number, the system updates the existing record in the target table. ○ If the **Match** toggle switch for the **Order number** field is off, the system doesn't check for matching data. If the source data has a record with the same order number, the system creates a new record in the target table. In this case, the target table will have two records, both with an order number of 743.
Configure reference and choice field behavior	Configure behavior for **Choice** and **Reference** field types. **a.** Click the gear icon (⚙). **b.** In **Field**, select the field column to match the incoming record value against in the target table. **c.** In **If no matching record exists then do the following**, specify what to do if that record isn't found in the target table. ▪ **Create a new choice/record:** Creates the choice or record in the target table's matching field column from the data imported for the field. ▪ **Ignore this field:** Ignores the field in the target table and leaves it null. ▪ **Skip this record:** Skips adding the entire record (row) in the target table. **d.** Click **Done**.
Add new field columns to target table	Add unmapped field columns from your import to your target table or create new fields in your target table.

Option	Description
	a. Select **Add fields** above the target table on the right. **b.** Choose which fields to add to your target ta ble by selecting the checkbox on the left or select **All** to choose all of them. **c.** You can also perform the following func tions: ▪ Edit the **Field Label** for the imported field. ▪ Edit field properties for the imported field by selecting the **Edit icon** ( ). ▪ Create a new field by clicking **Create a new field** in the top right corner of the window. **d.** Select **Add fields** to add the selected fields to your target table.

11. Select **Continue** to create the data import mapping and update your target table.
12. **Optional:** Convert any remaining worksheets if your file contained multiple worksheet tabs.
    - **Note:**
      For each sheet that you want to convert, you will need to select **Convert to table**, and then repeat steps in this procedure or choose a different pathway for the additional worksheet (see Create new table from spreadsheet import or Modify existing table using spreadsheet import for steps).
    - If you do not have multiple worksheets, proceed to the next step.
13. Select **Done** on the summary screen.
    Alternatively, to view your table in Table Builder, you can select **Edit table(s)** on the summary screen.

What to do next

For more information on editing your data tables in Table Builder, see Table Builder.

Modify existing table using spreadsheet import

Use a Microsoft Excel spreadsheet import to update an existing application table.

Before you begin

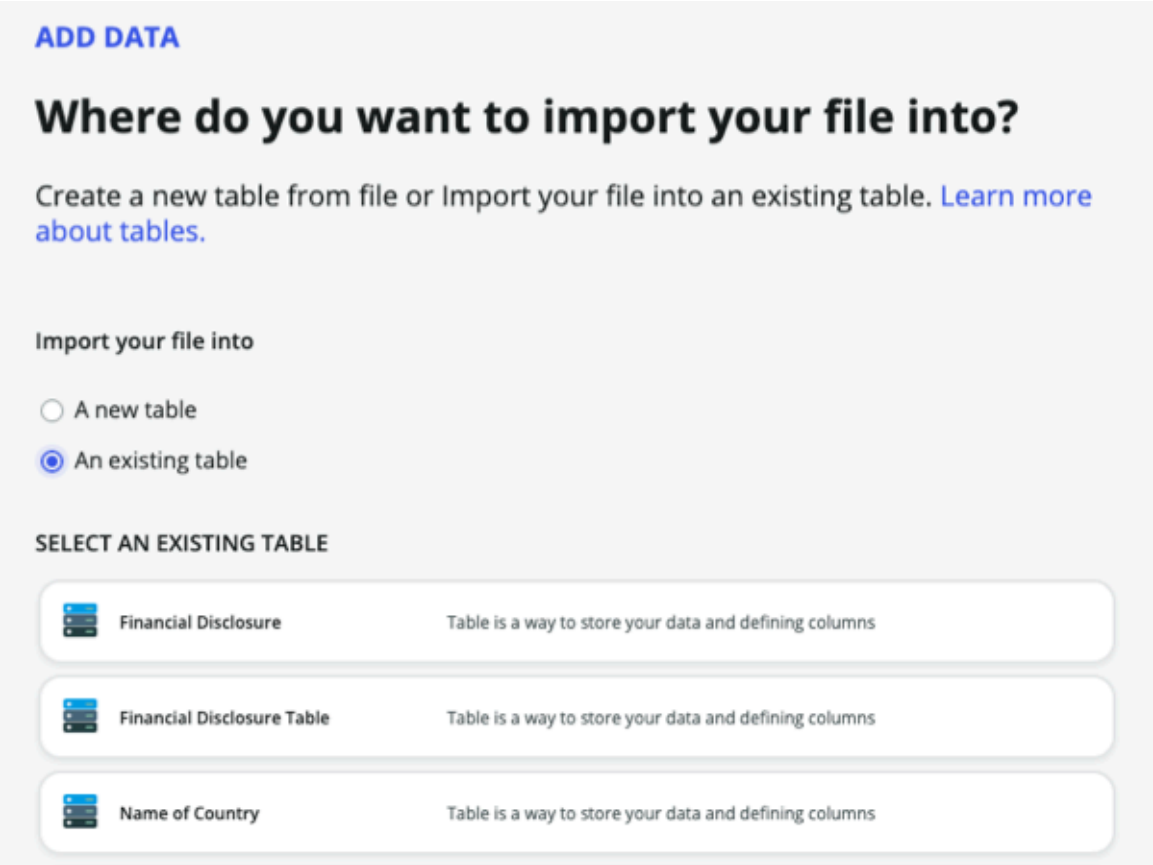
- Import a spreadsheet into App Engine Studio. See Import a spreadsheet.

Role required: admin, sn_app_eng_studio.user or delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. After you have uploaded your spreadsheet as described in Import a spreadsheet, choose where to import your file in the displayed wizard.

Import into an existing application table



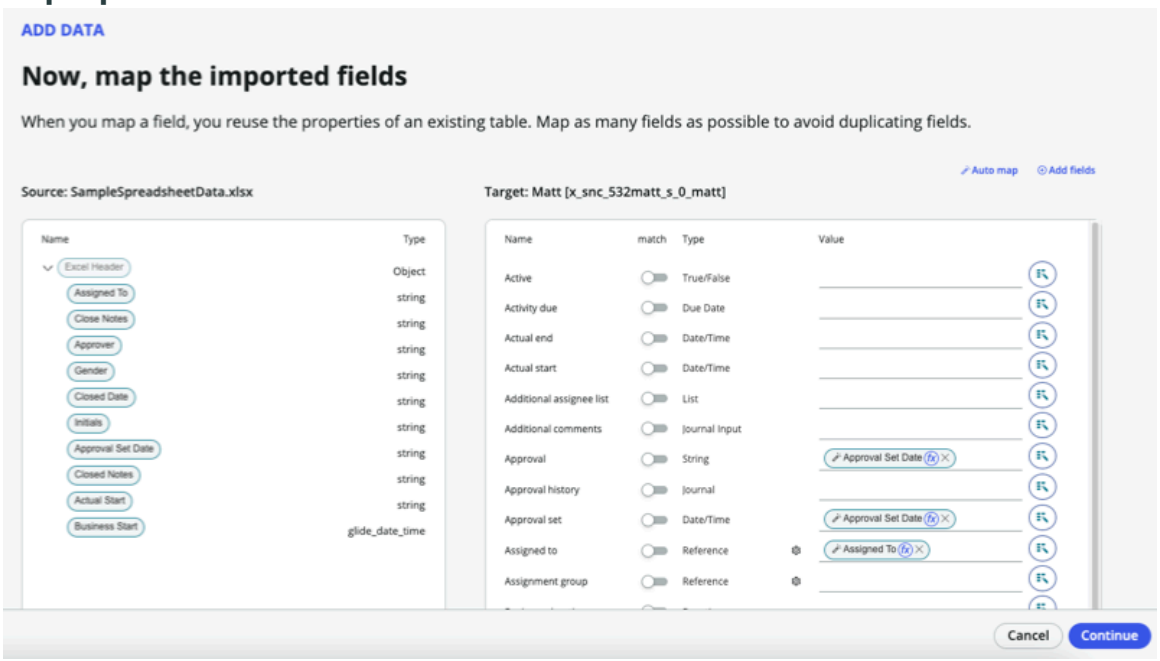
   a. Select **An existing table**.

   b. Select an existing application table from the list that displays.

2. Select **Continue**.
   A screen displays where you can map field columns from your source spreadsheet on the left to field columns in your existing target application table on the right.

> **Note:**
> You will have the opportunity to adjust mapping or create new fields from your import in it.

Map imported fields



3. You have the following actions available to you during this step.

Option	Description
Batch map fields using Automap	Select **Automap** to automatically map your spreadsheet fields to displayed fields in the selected target table. Automap maps source entities to similarly named target entities. For a target entity that has already been mapped, selecting **Automap** doesn't change its value. An automapped field has a wand icon on its data pill.
Map fields by dragging and dropping	Map fields from your import to your target table by dragging data pills representing columns in the imported spreadsheet to the **Value** column in the target table. You can also map fields by using the pill picker to the right of each target table field.
Map fields using the data picker	Map data by using the pill picker ( ) on the right side of the target table. You can enter more than one value in the **Value** column, you can also enter text.
Add transform options	Add transform options for each of the individ ual mapped spreadsheet fields by clicking the FX icon ( ) on the right side pane. For example, you can use the Uppercase trans form function to change an input string to all uppercase characters. See Transform func or information on how to use these

Option	Description
	to trim data for your imported spreadsheet fields. **Note:** Not all the transform functions available in Flow Designer are available.
Activate data record matching	Specify fields where the system should check for matching data and, if found, update existing records instead of creating new ones by enabling the **Match** toggle in your target table. For example, if a target table has an **Order number** field and a record with an order number of 743, there are two options for handling potential matches. ○ If the **Match** toggle switch for the **Order number** field is on, the system checks for matching data. If the source data has a record with the same order number, the system updates the existing record in the target table. ○ If the **Match** toggle switch for the **Order number** field is off, the system doesn't check for matching data. If the source data has a record with the same order number, the system creates a new record in the target table. In this case, the target table will have two records, both with an order number of 743.
Configure reference and choice field behavior	Configure behavior for **Choice** and **Reference** field types. **a.** Click the gear icon (⚙). **b.** In **Field**, select the field column to match the incoming record value against in the target table. **c.** In **If no matching record exists then do the following**, specify what to do if that record isn't found in the target table. ▪ **Create a new choice/record:** Creates the choice or record in the target table's matching field column from the data imported for the field. ▪ **Ignore this field:** Ignores the field in the target table and leaves it null. ▪ **Skip this record:** Skips adding the entire record (row) in the target table. **d.** Click **Done**.
Add new field columns to target table	Add unmapped field columns from your import to your target table or create new fields in your target table.

Option	Description
	a. Select **Add fields** above the target table on the right. **b.** Choose which fields to add to your target ta ble by selecting the checkbox on the left or select **All** to choose all of them. **c.** You can also perform the following func tions: ▪ Edit the **Field Label** for the imported field. ▪ Edit field properties for the imported field by selecting the **Edit icon** ( ). ▪ Create a new field by clicking **Create a new field** in the top right corner of the window. **d.** Select **Add fields** to add the selected fields to your target table.

4. Select **Continue** to create the data import mapping and update your target table.

   Note:
   You will be prompted if there are any unmapped fields remaining in your spreadsheet. You can map those fields or continue without mapping them.

5. **Optional:** Convert any remaining worksheets if your file contained multiple worksheet tabs.

   - **Note:**
     For each sheet that you want to convert, you will need to select **Convert to table**, and then repeat steps in this procedure or choose a different pathway for the additional worksheet (see Create new table from spreadsheet import or Create new table from extensible table and spreadsheet import for steps).
   - If you do not have multiple worksheets, proceed to the next step.

6. Select **Done** on the summary screen.
   Alternatively, to view your table in Table Builder, you can select **Edit table(s)** on the summary screen.

What to do next

For more information on editing your data tables in Table Builder, see Table Builder.

Use a PDF to create data tables

Create tables from a PDF form using the PDF extractor tool in App Engine Studio.

Note:
This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.

Overview

The PDF extractor in App Engine Studio allows you to upload a PDF and then extract table fields for use in a custom app. This streamlines the process of building application data tables from pre-existing PDF forms because you no longer need to manually type in field labels.

Note:
Ensure that you are mindful about recreating any tables that may already exist on the ServiceNow AI Platform. For example, with a PDF containing user demographic data, determine whether you want to reference existing user data tables rather than creating new tables from the PDF.

General navigation

The following table contains a list of some of the other basic navigational elements within the PDF extractor tool.

Navigational elements in PDF extractor

Navigational element	Description
Collapse / expand icons ()	Select these icons to toggle between collapsing or expanding the PDF and table panes.
Page controls ()	Use the arrows at the bottom of the PDF to page through the document one page at a time or navigate directly to the beginning or end of the document.
Zoom controls ()	Use the zoom controls to zoom in or out in the PDF.
Add new field icon ()	Add a field column to your table manually by selecting **+Add new field**, and then enter details directly. This is similar to how you add fields on the **Data** tab of Table Builder.
Add reference table icon ()	Select **Add reference table** in the right side panel to add a reference field in your table that points to a new table.
Open data table and form icon ()	Select **Open data table and form** to open the selected table within Table Builder.
Table additional actions list ()	Use the table additional actions menu to edit basic properties for your table, or if you have added a reference table, select **Delete** to delete the table.

Navigational elements in PDF extractor (continued)

Navigational element	Description
Trash icon (🗑)	Select the **Trash icon** (🗑) to delete fields from a table.

PDF parsing limitations

The following limitations currently exist when uploading a PDF using this tool:

- PDF cannot be converted from an image.
- PDF cannot be longer than 25 pages.
- PDF file size cannot exceed 5 MB.
- Languages other than English and French are not currently supported.

Create a table from a PDF form

Add data tables to your application by uploading a PDF in App Engine Studio.

Before you begin

Note:
This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.

Verify that the PDF meets the following requirements:

- PDF cannot be converted from an image.
- PDF cannot be longer than 25 pages.
- PDF file size cannot exceed 5 MB.
- Languages other than English and French are not currently supported.

Role required: admin, sn_app_eng_studio.user, or delegated_developer

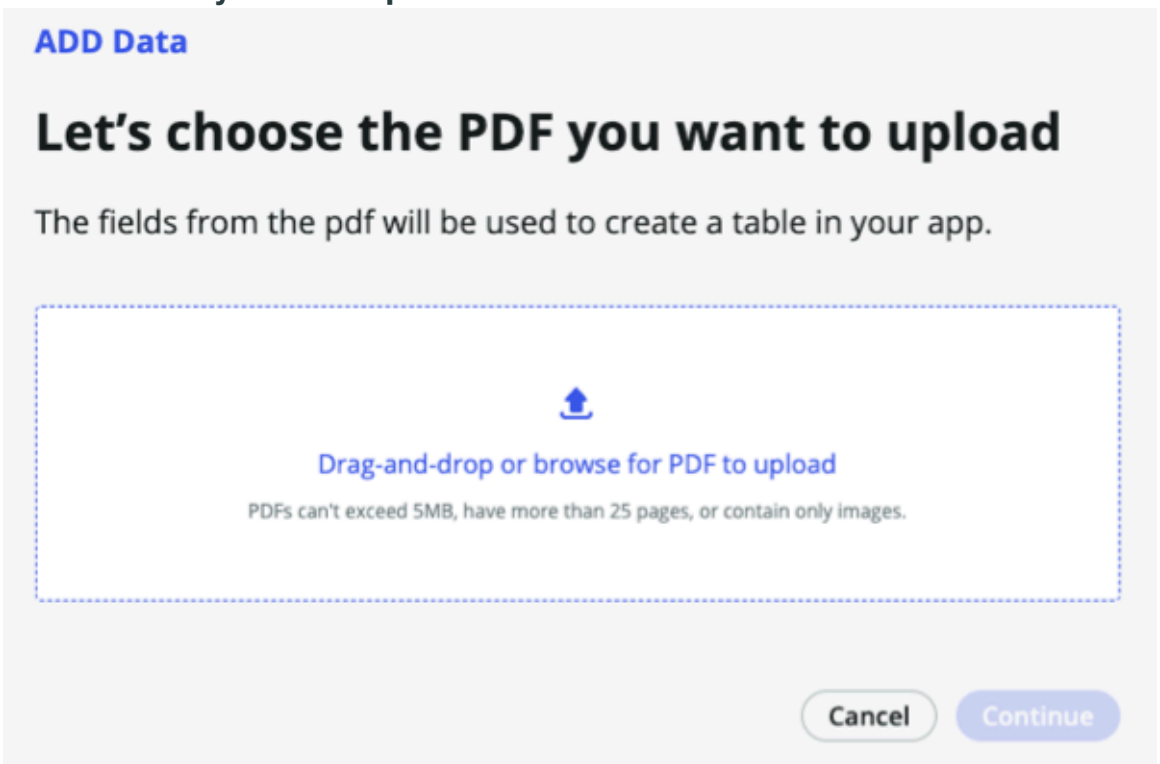
Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the Data heading, select the Add icon (⊕).
4. Select **Create a blank table**, and then select **Continue**.
5. Select **Upload a PDF** and then select **Continue**.
6. Select and upload the PDF.

   You can drag the PDF file onto App Engine Studio or browse your computer.

   Note:
   It may take a moment for the PDF to load. Select the trash icon (🗑) to clear the PDF selection.

Choose the PDF you want to upload



7. Select **Continue**.

> **Note:**
> It may take up to a minute for the PDF to be parsed.

8. Define the properties for your new table.

Property	Description
Table label	Unique label to identify the table.
Table name	Database name for the table. A table name is created automatically after you enter a table label. Once this is created, this field is read only.
Make extensible	Option to allow other tables to share data from this table. For more information on table extension, see Table extension.
Auto number	Option to track table records with a unique number. If you select this option, define the **Prefix**, **Starting number**, and **Number of digits**.
Prefix	Abbreviated name of the table to append to the beginning of the record number. For example, if you are creating a "Laptop" table, then your prefix may be "LPTP" or "LT."

9. Select **Continue**.
   You may now configure permissions for your new data table.
10. Control who has access to create, read, write, and delete content from this table by adding new roles and/or defining the desired Create, Read, Write, Delete permissions for existing roles.

    Roles that you create in the table persist throughout the application and can be further defined in other tables you add. See Add application security for more information on how security permissions work.

    Note:
    At least one role must have read access for you to be able to preview the table.

11. Select **Continue** after you have configured table permissions.
12. Select **Go to PDF extractor** to open up the PDF in the PDF extractor.

What to do next

Next, you will add fields from the uploaded PDF. Follow the steps in Add new fields from a PDF form to populate your new data table with field columns.

Add new fields from a PDF form

Add field columns to your data table using the PDF extractor in App Engine Studio.

Before you begin

Note:
This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.

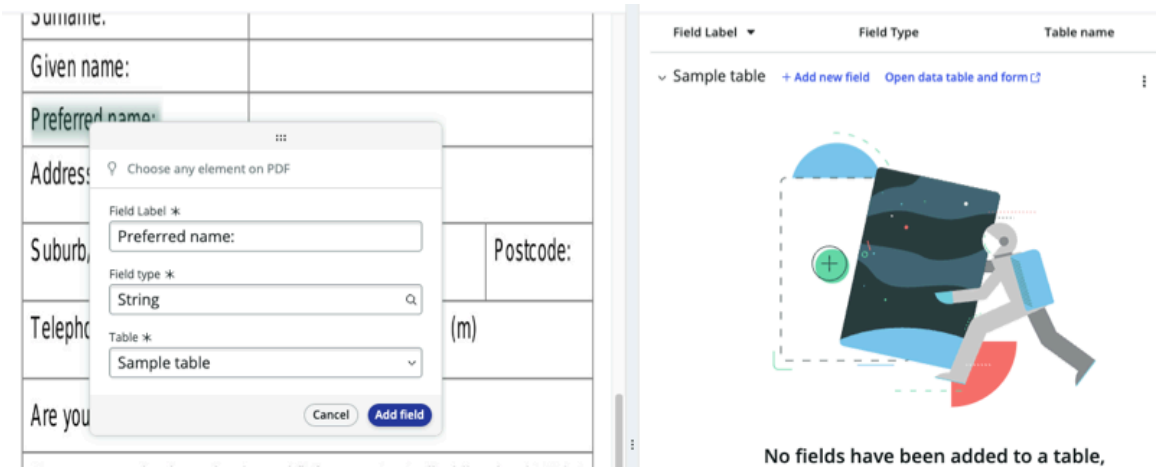
Role required: admin, sn_app_eng_studio.user, or delegated_developer

Procedure

1. Choose one of the following options:
   - Follow the steps outlined in the previous procedure to upload a PDF (see Create a table from a PDF form).
   - If you have already uploaded the PDF, select the **Open pdf extractor** element on the **Forms** or **Table** tab in Table Builder to launch the PDF extractor tool.

   The PDF form displays.
2. Review the PDF.
   Use any of the navigational controls in the PDF pane to navigate to where you'd like to begin adding fields. See General navigation.
3. Click **Select** to enter text selection mode.
4. In the PDF, click on a field or text phrase to highlight it to use it as the basis for a new field.

Add fields from PDF



5. Edit the extracted text in the **Field Label** as needed for the new field you are going to add.
6. Choose a **Field type**.

   Field type refers to the type of data that the new column will store. For example, to store plain text in the column, select **String**. To understand the basic field properties, see Add and customize a field in a table.

   Depending on the type that you select, you may need to fill in additional fields to further define the table column. For example, with Reference field types, search for and select the appropriate data table to reference.

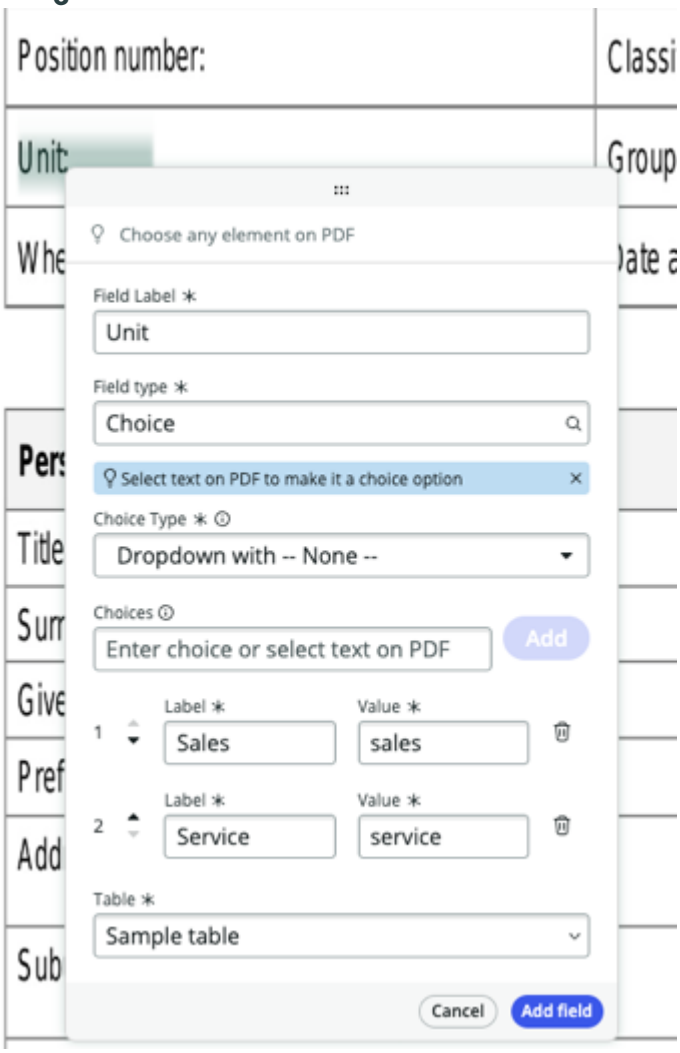
   a. **Optional:** If you have chosen to create a choice field, perform the following actions to populate options for the choice field.

      i. Choose a **Choice Type** from the menu. This controls whether or not the list will have None as an option.

      ii. Click in the **Choices** field, and then enter text manually or select text on the PDF to automatically extract it.

      iii. Click **Add** to add your choice to the list.

Adding a choice field



b. Optional: Repeat the previous steps until all choices have been added for your choice field.

7. Select the **Table** where you want to add the new field.
   Multiple options may appear if you have added reference tables. The original table is selected by default.
8. Select **Add field** to add the field to the list of fields in the table on the right.

> **Note:**
> Any fields you add to the table on the right remain highlighted in green in the PDF to remind you that you have created a related field from them. If you delete a field from the tables on the right, the extracted text will no longer be highlighted.

9. Repeat the steps above until you have added all the fields you want to add from the PDF.

> **Note:**
> You can also add fields manually by clicking **+Add new field** in the table on the right and entering details directly similar to how you add fields on the **Table** tab of Table Builder.

10. Review the fields in your table and edit as necessary.

Table pane actions

Action	Description
Move fields between tables	To move a field between tables within the PDF extractor, change the selected **Table Name** to the desired table location.
Delete reference table	Delete any additional reference tables that you add by selecting **Delete** from the Additional actions list in the top right corner of the table (⋮)
Add a new field	Add a field column to your table manually by selecting **+Add new field**, and then enter details directly. This is similar to how you add fields on the **Data** tab of Table Builder.
Edit a field	To edit field properties (e.g., Field type), click on the field property to select it, and then make your edits.
Delete a field	Select the **Trash icon** (🗑) to delete fields from a table.
Open table within Table Builder	Select **Open data table and form** to open the selected table within Table Builder.

11. Select **Save**.

What to do next

To open your data table in Table Builder, select **Open data table and form** above the displayed tables. For more information on editing your data tables in Table Builder, see Table Builder.

To add a reference table, see Add a reference table from a PDF form.

Add a reference table from a PDF form

Move fields into a new referenced data table using the PDF extractor in App Engine Studio.

Before you begin

Note:
This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.

Role required: admin, sn_app_eng_studio.user, or delegated_developer

About this task

You may decide that you want to add some of the form information into separate tables that are referenced from the original table and can be referenced by other tables. For example, if you work for an insurance company, you may want to create separate data tables for bills, claims, and policies that can be referenced as separate data tables even though the information is extracted from a single PDF form.

You can add a reference field, and then create the corresponding reference data table directly from the PDF extractor in App Engine Studio.

Note:
Before you create any new reference tables for your app, it's a good practice to verify that you are not accidentally recreating any tables that already exist on the ServiceNow AI Platform.

Procedure

1. Choose one of the following options:
   - Follow the steps outlined in the previous procedure to upload a PDF (see Create a table from a PDF form).
   - If you have already uploaded the PDF, select the **Open pdf extractor** element on the **Forms** or **Table** tab in Table Builder to launch the PDF extractor tool.

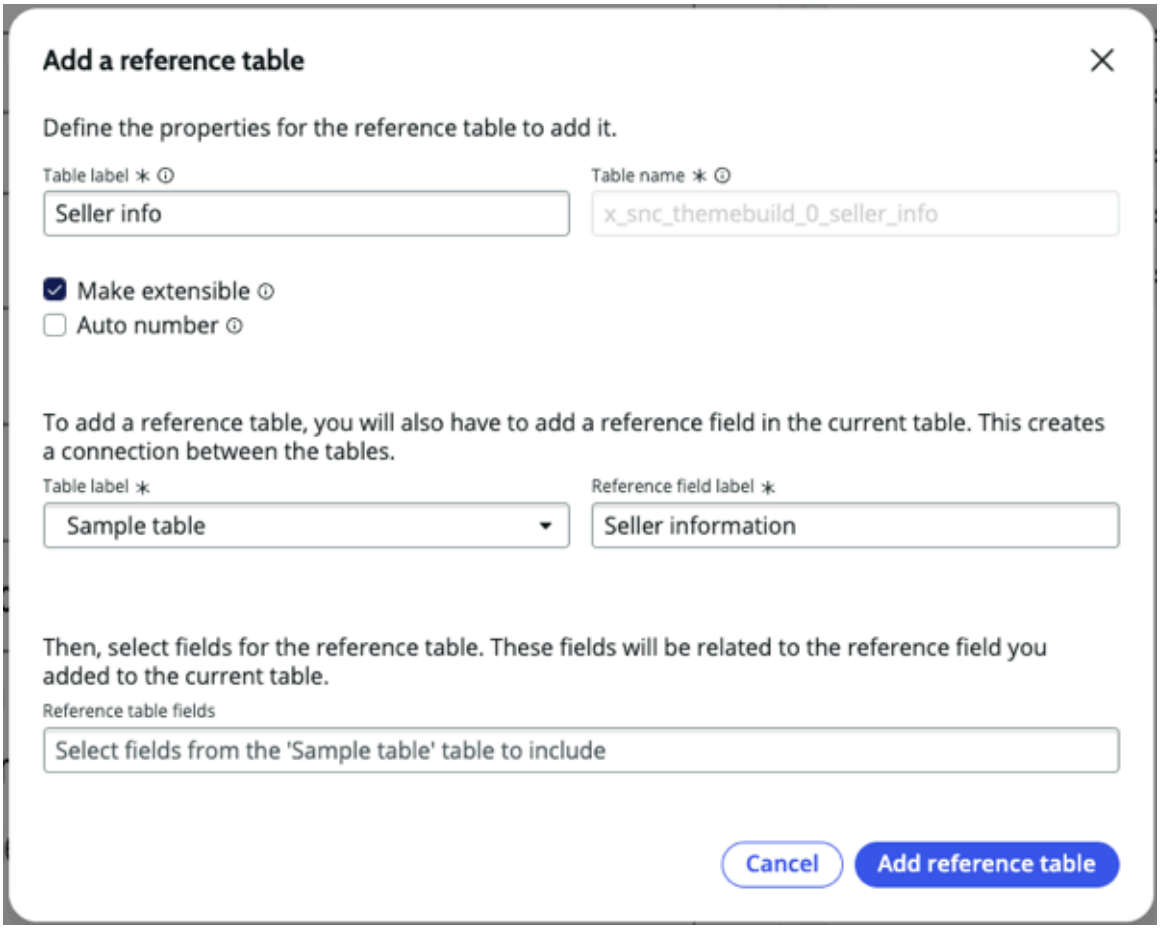
   The PDF form displays.
2. Select **Add reference table** in the right side panel to add a reference table to your table.
3. Enter the following properties for your reference table.

Column header property	Description
Table label	Unique label for the reference table.
Table name	Database name for the reference table.
Make extensible	Option to allow other tables to share data from this table. For more information on table extension, see Table extension.
Auto number	Option to track table records with a unique number. If you select this option, define the **Prefix**, **Starting number**, and **Number of digits**.

Next, you will add a reference field to the original table you are editing. This creates the connection between the tables (e.g., "Seller info").

4. Select the **Table label** of the table where you want to add your reference field in (the original table is selected by default).
   The original table should be selected by default in the **Table label** field.
5. Enter a **Reference field label** for this new reference field (e.g., "Seller information").
6. Select fields from your original table to add them to your new reference table.
   These fields are related to the reference field you added to the current table.

Add a reference table



7. Select **Add reference table**.
   The reference table appears below and contains the fields that you selected that were in the original table.
8. Review and edit the fields and table properties as necessary.

Table pane actions

Action	Description
Move fields between tables	To move a field between tables within the PDF extractor, change the selected **Table Name** to the desired table location.
Delete reference table	Delete any additional reference tables that you add by selecting **Delete** from the Additional actions list in the top right corner of the table (⋮)
Add a new field	Add a field column to your table manually by selecting **+Add new field**, and then enter details directly. This is similar to how you add fields on the **Data** tab of Table Builder.

Action	Description
Edit a field	To edit field properties (e.g., Field type), click on the field property to select it, and then make your edits.
Delete a field	Select the **Trash icon** ( ) to delete fields from a table.
Open table within Table Builder	Select **Open data table and form** to open the selected table within Table Builder.

See General navigation for more information on additional navigational controls in the table pane.

9. Select **Save**.

What to do next

To open your data table in Table Builder, select **Open data table and form** above the displayed tables. For more information on editing your data tables in Table Builder, see Table Builder.

Create a data integration

Add new or updated information to your current data sets by importing it into existing tables in your application through the power of Integration Hub. Schedule future imports to add data at specific times.

This video shows you how to perform the following procedure.

https://player.vimeo.com/video/982217306?badge=0&autopause=0&player_id=0&app_id=58479

Before you begin

- You must have an existing table in your application. For more information, see Create a blank table.

  Note:
  Currently, Excel files are the only accepted data format users can upload.

- Role required: admin, sn_app_eng_studio.user, or delegated_developer. For more information, see Delegate developers using AES.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the Data heading, select the Add icon ( ).
4. Select **Create a data integration**.

Configure your data import source

5. Configure your data import source by performing the following steps.

   a. Fill in the fields for the data integration.

Field	Description
Name	Name of your integration.
Short description	Description of your integration.
Application	Name of the application scope for your integration.

   b. Select **Save & Continue**.

   c. Use the **Source type** menu to select a data source type.

   - To choose an Excel or CSV file, select **AD-HOC** > > **File**, then either **Excel** or **CSV**. In the File upload section, upload an XLSX, XLS, CSV, or ZIP file.
   - To choose a Data Stream action, select **Spokes**. Active Data Streams are organized by spoke and listed alphabetically. In the Connection section, select the connection alias. For Data Stream actions that use a connection alias, you can override the default connection alias with any of its children's aliases. For more information, see Create a Connection and Credential alias.

   d. Configure the data source.

   - To configure an Excel spreadsheet, specify which sheet to use and which row number to use for the header. Only one sheet and one header can be specified.
   - To configure a CSV file, select the delimiter.
   - To configure a data stream action, enter any required inputs. This section only appears if the Data Stream action has inputs to enter. Data Stream actions with inputs have unique inputs, so the fields in this section depend on which Data Stream action is selected and its configuration. Any default values already configured by the spoke inputs are pre-filled. You can view the details of the Data Stream action by clicking the information icon ( ) next to it. Clicking the icon opens the Data Stream action's configuration.

6. Select the **Save** button.

Choose your target table and configure import options

7. Select the **Map to target** tab.
8. Select the target table and options for the data integration you're creating.
   Mapping the source data to the target table allows you to control how the system treats the data you uploaded.

   a. On the Source to target table data mapping page, select **Add a table**.

   b. In the Mapping Properties window, fill in the fields.

Field	Description
Target table	Name of the table to transform your data to.

Field	Description
Run table's business rules when importing	Option to run the target table's business rules when you import the data.
Run the import synchronously	Option to run the import synchronously. When data is transformed in parallel, and there are multiple source records with a specific coalesce value, checking this option ensures that only one record with that coalesce value is inserted at a time. For more information, see the Synchronized inserts section on the Concurrent imports page.

c. Select **Save**.

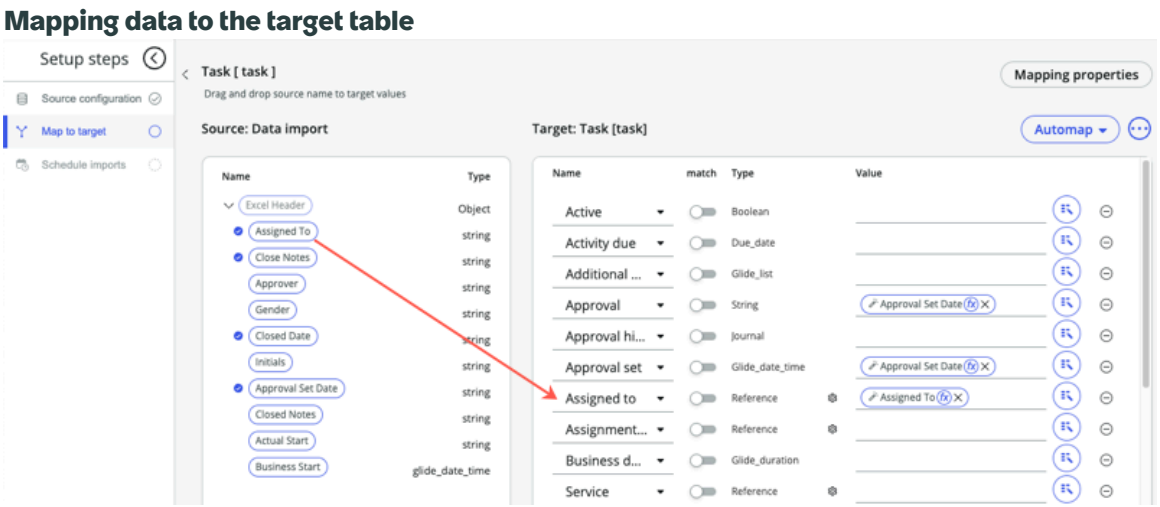
You can add multiple target tables to your integration. Repeat steps 4a through 4c for each target table you want to add.

Tables are listed in the order they're added. You can change the order by using the drag icon ⠿ to the left of each table card to drag the card to a different position. The order of the table cards determines the order in which data is transformed into the target tables at runtime.

Note:
You can't select the same table more than once, and you can't select an extended table or parent table of a table already in the integration.

Map import fields to your target table

9. Map the fields in your import to the target table by performing the following actions.



a. On the Source to target table data mapping page, select the button with the target table you want to map to.
This opens the data-mapping section for the table. The left side of the page shows the source data. The right side is for the target table.

b. Use the **Add a field**, **Add all fields**, or **Automap** button to configure the target side and begin mapping data.

- To add all the fields in the target table, select **Add all fields**. Fields are added alphabetically.
- To add fields individually, select **Add a field** and choose the field name from the list. A field can only be added once.
- To add fields and map them automatically, select **Automap**. Automap maps source entities to similarly named target entities. The **Automap** button has two options.
  - **All matching fields in the target table** adds all the fields and maps the ones that match.
  - **All unmapped fields in the target selection** maps only the fields that you've already added but haven't mapped yet. If you haven't added any fields yet, this option is not available.

- For a target entity that has already been mapped, selecting **Automap** doesn't change its value. An automapped field has a wand icon on its data pill.

  By default, Automap maps a maximum of one source entity to a target entity. You can change this by setting the *`glide.ih.automap.extractone`* property to false.

  Automap matches source to target entities by looking for column names that are at least an 80% match. You can modify this with the *`glide.ih.automap.minimum.score`* property. For example, to make matches based on a 60% similarity, you can set *`glide.ih.automap.minimum.score`* to 60.

c. Map data by dragging data pills from the source to the **Value** column on the target table.

You can also map data by using the pill picker () on the right side of the target table. You can enter more than one value in the Value column, you can also enter text.

After you map a source field to a target field, a small check mark icon () appears next to the source field name. If you move your cursor to the check mark, the tooltip lists the target fields that source field has been mapped to, up to five fields.

d. Optional: Use the **Match** toggle switch to specify fields where the system should check for matching data and, if found, update existing records instead of creating new ones.

(Optional) For example, if a target table has an **Order number** field and a record with an order number of 743, there are two options for handling potential matches.

- If the **Match** toggle switch for the **Order number** field is on, the system checks for matching data. If the source data has a record with the same order number, the system updates the existing record in the target table.
- If the **Match** toggle switch for the **Order number** field is off, the system doesn't check for matching data. If the source data has a record with the same order number, the system creates a new record in the target table. In this case, the target table will have two records, both with an order number of 743.

e. Optional: For choice and reference fields, specify the column to map data to.

i. Select the gear icon (⚙).

ii. In **Field**, select the column to map to.

iii. In **If no matching record exists then do the following**, specify what to do if that column isn't available.

- **Create a new choice/record**: Creates the choice or record in the target table.
- **Ignore this field**: Ignores only the field in the target table.
- **Skip this record**: Skips the entire record in the target table.

iv. Select **Done**.

f. **Optional:** Change or modify your data by applying transform functions.
For example, you can use the Uppercase transform function to change an input string to all uppercase characters. For more information, see Transform functions.

Note:
Not all the transform functions available while creating flows and subflows are available in Integration Hub - Import.

i. Select the transform function icon ( ).

ii. Select a transform function from the menu, then select **OK**.

iii. To apply additional transform functions, select **Add a new transform**. Transform functions are applied in the order they're selected.

iv. After you've added all your transform functions, select **Done**.

10. When you're done with your mapping, click **Save**.

Import the file or schedule the data import

11. Choose what to do with your data import you just created and mapped.
    - To import the data set you uploaded now, select **Run Import**.
    - To set up a future import of the data, select **Schedule an import**.

    See Run or schedule a data import.
12. **Optional:** If you ran the import, preview the table and ensure it appears with the additional content by returning to the target table, refreshing your browser screen, and clicking **Preview**.
13. **Optional:** If you added new columns during the import and want them to display on the target table, select **Personalize list** to edit the columns that display.

Modify application data tables

Use Table Builder to modify data tables used in your application.

Role required: admin, sn_app_eng_studio.user, or delegated_developer

To modify data tables associated with your application:

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the **My Apps** page, open your application.
3. Under the Data heading, select the Additional actions icon (•••) for an existing table in your application, and then select **Edit**.

When you click on a data table or select **Edit** in the Additional actions menu (•••) to edit data for your application, Table Builder is launched. Table Builder is a tool for editing data tables that you've added to your application.

You can perform the following actions in Table Builder:

- Manage your application data tables on the **Data** tab.
  - Manage table fields.
  - Edit table and field column properties.
  - View schema relationships between your data tables.

    Note:
    This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.

- Create and manage form views and display logic that is associated with the table on the **Forms** tab.
- Create and manage flows triggered when the table is updated or created on the **Flows** tab.

Note:
This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.

See Data in Table Builder for more information on editing tables in Table Builder.

Note:

To delete a table from your application, select the Additional actions icon (•••) for an existing table in your application, and then select **Delete** and follow the instructions.

Add an application experience

Use App Engine Studio (AES) to add an experience to your app. Experiences are the interface, or wrapper, for how users interact with your app.

Application templates automatically add experiences to your application.

Types of experiences

Experience	Definition	Example	Reference
Standard catalog item	A standard catalog item displays a form that users can fill in to create requests, such as a form to request an item or service. Standard catalog items make use of the Request table and produce a request record. In addition, generated request items are linked with a workflow, a catalog task, and any related approvals. Using standard catalog items, users can add multiple items to a	Create a catalog item form, such as a form to request time-off. After submitting the form, a request is created for a manager to approve or reject.	• Add a standard catalog item • Sample standard catalog item

Types of experiences (continued)

Experience	Definition	Example	Reference
	cart and generate a request for each requested item. Use a catalog item when you want the service catalog form to generate a request, complete with a workflow, approvals, tasks, etc.		
Record producer	A record producer is a simplified type of catalog item that displays a form for users to fill out. The completed and submitted form inserts a new data record in a selected custom target table that you set up. This type of catalog item enables users to create task-based records, such as an incident record (without any of the more complex functionality that comes with using a standard catalog item (e.g., creating requests). Use a record producer when you want the service catalog form to create a simplified type of record that isn't a request (e.g., a task record such as an incident, change, or enhancement, etc.).	Create a service catalog form that generates a task record, such as an incident, instead of a request.	• Add a record producer • Sample record producer
Workspace	A workspace is a suite of tools that provides agents, case managers, help desk professionals, and managers with tools to help answer customer questions and resolve customer problems.	Create an equipment fulfillment dashboard where employees can request laptops and other items. Workspaces are similar to portals but smaller in scope.	• Add a workspace • Building workspaces in AES • Configure a workspace in Workspace Builder • Configure workspace settings

Types of experiences (continued)

Experience	Definition	Example	Reference
			in Workspace Builder • Configure a workspace home page in Workspace Builder • Configure a record page for a workspace in Workspace Builder • Configure lists for a workspace in Workspace Builder • Configure analytics for a workspace in Workspace Builder • Sample workspaces you can build
Portal	A portal is a site where users inside of your organization can find information, submit requests, and complete business tasks.	Create a self-service portal, or entry point for employees to make requests and browse a knowledgebase, which you can then populate with additional workspaces and catalogs.	• Add a portal • Sample portals you can build
Mobile	A mobile experience enables users to access your application from a ServiceNow native mobile app. You must then populate the app with data.	Create a time-off request application for mobile for managers to approve or reject requests in the Now Mobile app on their mobile device.	• Add a mobile experience • Sample mobile experience

Choose between record producers and catalog items

Both the record producer and standard catalog item in App Engine Studio (AES) create experiences where users can submit requests through a service portal. However, record producers are more extensible, and enable users to create a record in any table.

Consider the level of customization

Record producers allow for more customization, making the most of the App Engine tools to build experiences from scratch.

Catalog items represent services or items a user can request. The catalog of all items is called the Service Catalog, and is part of Service Catalog Management.

Comparison of record producers and standard catalog items

Functional area	Record producer	Catalog item
Purpose	Create custom fulfillment experiences using App Engine automation tools.	Support simple request management applications for which a predefined approval and qualification process exists, such as requesting IT hardware and software.
Extensibility	Customizable.	Pre-defined.
Example	Password reset form that creates a password reset request.	Form to request a parking permit.
Development model	Process is as follows: **1.** Set up a business-focused data model. **2.** Define the kind of experience your users will leverage. **3.** Define the automation or business logic. **4.** Define the user roles that will use the app.	Catalog items heavily rely on the request fulfillment process for fulfilling items defined in Service Catalog, a central, accurate, and consistent source of data.
Reporting	Easier to create dashboards with a purpose-built data model.	Requires creating database views and scripting to gather variables into other tables to configure specific reports.
Scalability	Business-focused app makes it easier to find resources and make app changes in a low-code environment.	Typically part of a Service Catalog Management operation with a dedicated team managing each catalog.
Security	Scoped apps allow flexibility to manage permissions without being an administrator.	Catalog items are typically built in a specific scope, and less flexible.

Help topics on record producers and standard catalog items

For more information, see the following topics:

- Record producers:
  - Add a record producer
  - Sample record producer
- Standard catalog items:

- Add a standard catalog item
- Sample standard catalog item

Add a record producer

A record producer enables users to create task-based records, such as an incident record, in apps you create in App Engine Studio (AES) without creating a request. It enables a better end-user experience by standardizing requests using a simplified form.

Before you begin

Create a data table that the record producer will add a new record to when a user fills out the form. For more information, Create a data model for your application.

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task

A record producer is a simplified type of catalog item that displays a form for users to fill out. The completed and submitted form inserts a new data record in a selected custom target table that you set up. This type of catalog item enables users to create task-based records, such as an incident record (without any of the more complex functionality that comes with using a standard catalog item (e.g., creating requests). Use a record producer when you want the service catalog form to create a simplified type of record that isn't a request (e.g., a task record such as an incident, change, or enhancement, etc.). When you add a record producer, you create a basic version of the record producer. You must then continue editing it in Catalog Builder before it's ready to deploy.

Note:
If you created an application using a template, a record producer may already be added to your application.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, next to Experience, select **Add**.
4. Select **Record Producer** and select **Begin**.
5. On the form, fill in the fields.

Record Producer form

Field	Description
Name	Name for the record producer.
Short description	Brief overview of the record producer.

6. Select **Edit record producer**.
7. In Catalog Builder, specify the required information.
   After completing each step, select **Save**.

Catalog Builder steps

Step	Description
Details	○ Basic info: Record producer name and short description. ○ Item details: Description of the record producer. **Note:** You can attach files such as graphics, images, videos, or a link to an external source of information.
Destination	Table in which the record producer creates records.
Location	○ Catalogs: Name of the catalog the record producer belongs to. ○ Categories: Available categories that can be linked with the record producer. ○ Topics: Available topics that can be connected to the standard catalog item. Find out more about topics in Associate a catalog item with a taxonomy topic in Employee Center.
Questions	Questions for the record producer. From the **Insert new question** list, you can include additional question sets, questions, deactivated questions, single-column container, two-column container, and line break. For information on creating a question and supported question types, see Create a question for a catalog item in Catalog Builder. You can also add dynamic form behavior for a question. For information on adding dynamic form behavior, see Edit a question in Catalog Builder.

Step	Description
	Note: ○ The questions within a question set can't be edited. The question sets can be reordered by dragging and dropping them. ○ Question sets specified in the template can't be removed, while the ones added by the user can be removed. ○ Removal of question sets simply removes the association with the item and doesn't delete the question set. ○ A single-column question set can be added to a single-column or a two-column container. A two-column question set can't be added to a container.
Settings	Options to customize the record producer behavior. ○ Hide 'Add to wishlist' button: Option to hide the **Add to wishlist** button for the record producer. ○ Hide attachment button: Option to hide the **Attachment** button for the record producer ○ Make attachment mandatory: Option to make the **Attachment** field required for the record producer.
Access	Group or individual users for whom the record producer is available or not available.
Review and Submit	Review and submit the record producer.

8. To preview a catalog item in Portal or Now Mobile, select **Preview**.

   Note:
   When you preview an item, you can interact with it but not submit it.

   a. To preview the item in the portal, select **Portal** for the **View within** field.

      Note:
      Portal preview is based on the portal URL configured for the Catalog Builder.

   b. To view an item representation in Now Mobile, select **Now Mobile** for the **View within** field.

c. To open the preview in a new tab, select **Open preview in a new tab**.

> **Note:**
> When you change the item and save it, the preview is dynamically refreshed in the opened tab.

9. Select **Submit**.

What to do next

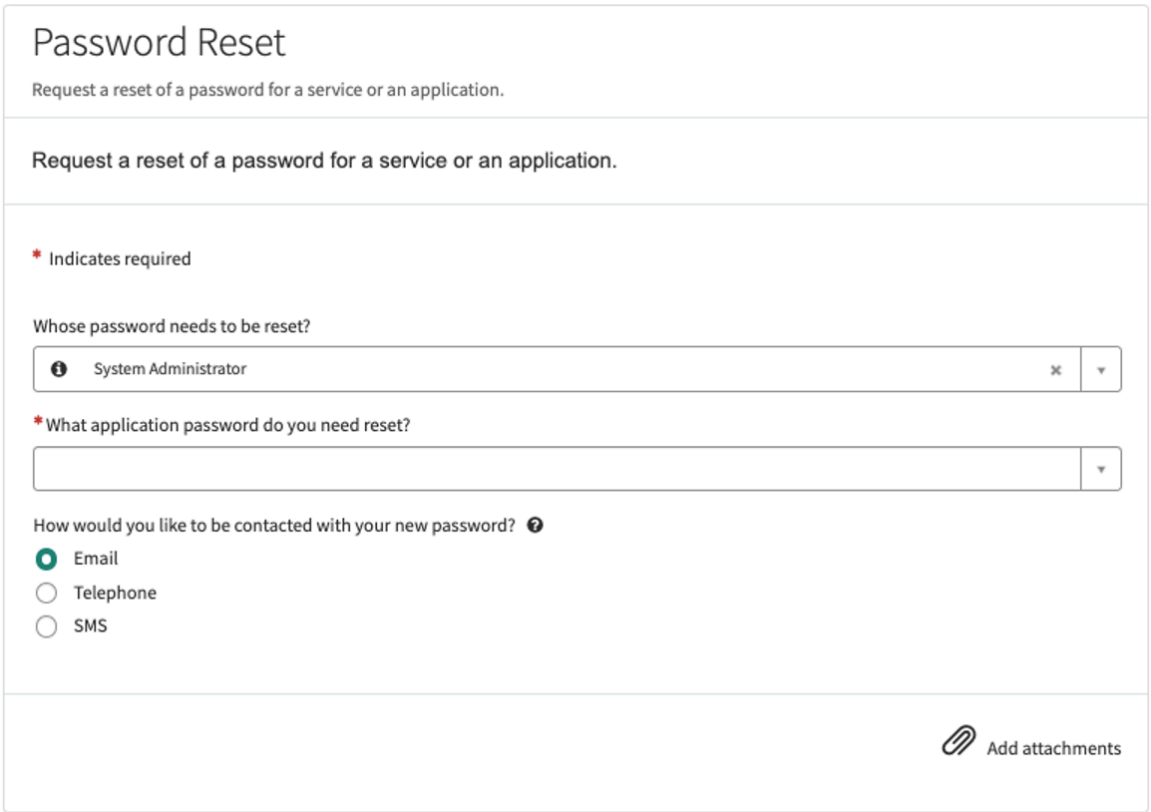
For the record producer you want to edit, select the menu icon (•••) and select **Edit**.

Sample record producer

A record producer experience that you create in App Engine Studio (AES) is a service catalog form that generates a task record, such as an incident, instead of a request.

Example record producer

The following provides an example of a record producer that creates a password reset request.



Add a standard catalog item

Add a standard catalog item to your application in App Engine Studio (AES) so users can submit requests for services and offerings.

Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task

A standard catalog item displays a form that users can fill in to create requests, such as a form to request an item or service. Standard catalog items make use of the Request table and produce a request record. In addition, generated request items are linked with a workflow, a catalog task, and any related approvals. Using standard catalog items, users can add multiple items to a cart and generate a request for each requested item. Use a catalog item when you want the service catalog form to generate a request, complete with a workflow, approvals, tasks, etc. When you add a standard catalog item, you're creating a standard version of the catalog item, which you must then continue editing in Catalog Builder before it's ready to deploy.

Note:
If you created an application using a template, a standard catalog item may already be added to your application.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, next to Experience, select **Add**.
4. Select **Standard catalog item** and select **Begin**.
5. On the form, fill in the fields.

ADD EXPERIENCE form

Field	Description
Name	Name for the standard catalog item.
Short description	Brief overview of the standard catalog item.

6. Select **Edit catalog item**.
7. Specify the required information.

   After completing each step, select **Save**.

Catalog Builder steps

Step	Description
Details	○ Basic info: Standard catalog item name and short description. ○ Item details: Description of the standard catalog item. **Note:** You can attach files such as graphics, images, videos, or a link to an external source of information.
Location	○ Catalogs: Name of the catalog the standard catalog item belongs in. ○ Categories: Available categories that can be linked with the standard catalog item.
Questions	Questions for the standard catalog item.

Step	Description
	From the **Insert new question** list, you can include additional question sets, questions, deactivated questions, single-column container, two-column container, and line break. For information on creating a question and supported question types, see Create a question for a catalog item in Catalog Builder. You can also add dynamic form behavior for a question. For information on adding dynamic form behavior, see Edit a question in Catalog Builder. **Note:** ○ The questions within a question set can't be edited. The question sets can be reordered by dragging and dropping them. ○ Question sets specified in the template can't be removed, while the ones added by the user can be removed. ○ Removal of question sets simply removes the association with the item and doesn't delete the question set. ○ A single-column question set can be added to a single-column or a two-column container. A two-column question set can't be added to a container.
Settings	Configure the standard catalog item based on the request method. Request method: ○ Order: Use this method in scenarios, such as a corporate laptop request. ○ Request: Use this method in scenarios, such as a code access request. ○ Submit: Use this method in scenarios, such as a reset password request. Other options: ○ Hide 'Add to cart' button: Option to hide the **Add to Cart** button for the standard catalog item. This option is automatically selected if the **Request method** is **Request** or **Submit**. ○ Hide 'Add to wishlist' button: Option to hide the **Add to wishlist** button on standard catalog items.

Step	Description
	○ Hide quantity selector: Option to hide the **Quantity** list on standard catalog items. ○ Hide delivery time: Option to hide the **Delivery time** field on the standard catalog items. This option is automatically selected if you set the **Request method** to **Submit**. ○ Hide attachment button: Option to hide the **Attachments** button on standard catalog items. ○ Make attachment mandatory: Option to make the **Attachment** field mandatory on standard catalog items.
Access	Specify the group or individual users for whom the standard catalog item is available or not available.
Fulfillment	Flow with which you can associate the standard catalog item. The flow defines the fulfillment process for the standard catalog item. For more information on flows, see Flows.
Review and Submit	Review and submit the standard catalog item.

8. To preview a catalog item in Portal or Now Mobile, select **Preview**.

   Note:
   When you preview an item, you can interact with it but not submit it.

   a. To preview the item in the portal, select **Portal** for the **View within** field.

      Note:
      Portal preview is based on the portal URL configured for the Catalog Builder.

   b. To view an item representation in Now Mobile, select **Now Mobile** for the **View within** field.

   c. To open the preview in a new tab, select **Open preview in a new tab**.

      Note:
      When you change the item and save it, the preview is dynamically refreshed in the opened tab.

9. Select **Submit**.

What to do next

For the standard catalog item you want to edit, select the menu icon (•••) and select **Edit**.

Sample standard catalog item

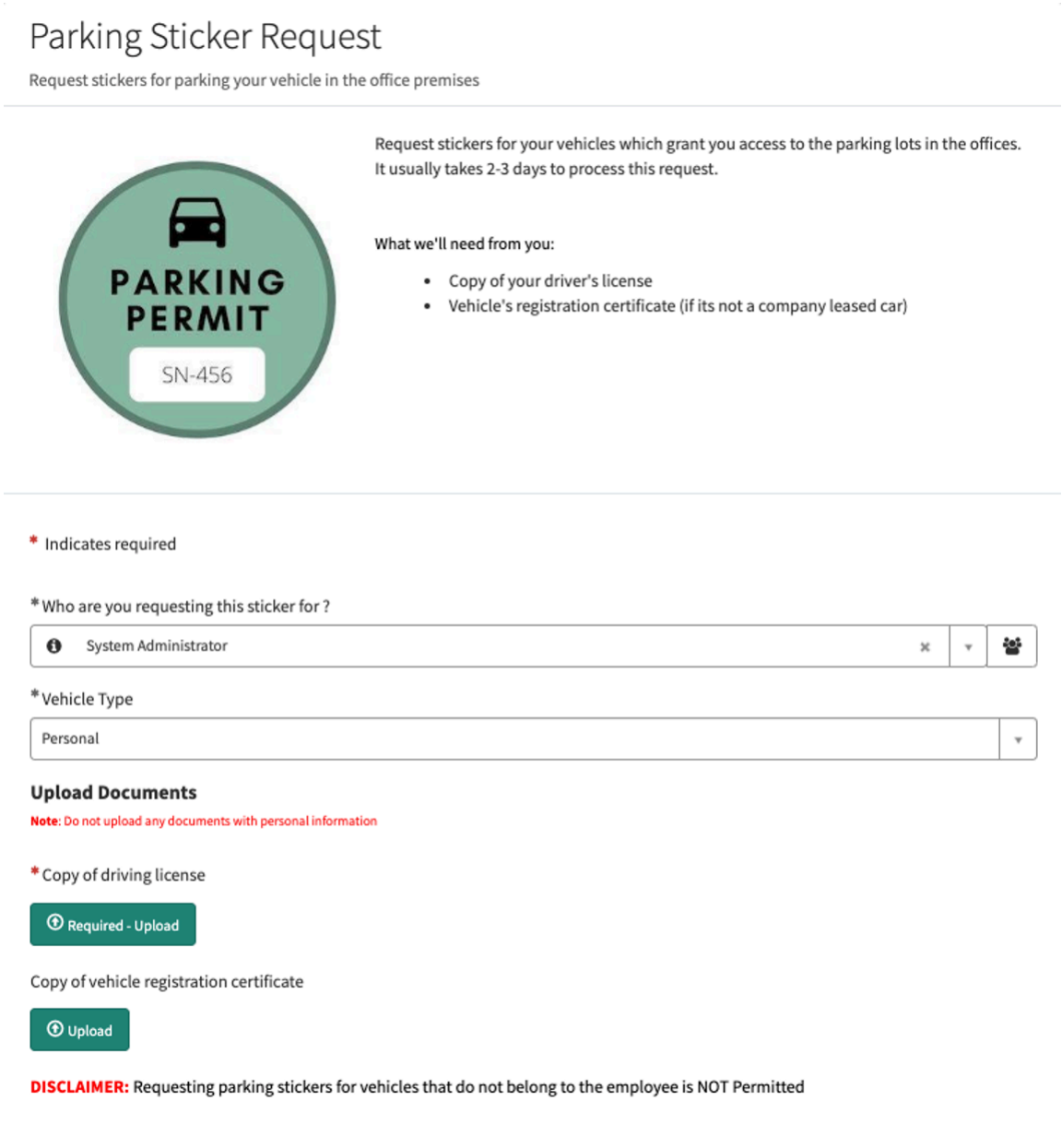
A standard catalog item displays a form that users can fill in to create requests, such as a form to request an item or service. Standard catalog items make use of the Request table and produce a request record. In addition, generated request items are linked with a workflow, a catalog task, and any related approvals. Using standard catalog items, users can add multiple items to a cart

and generate a request for each requested item. Use a catalog item when you want the service catalog form to generate a request, complete with a workflow, approvals, tasks, etc.

For example, an employee can fill in a catalog item form to request time-off for a holiday. After submitting the form, a request is created for a manager to approve or reject.

Example standard catalog item

The following provides an example of a standard catalog item to request a parking permit.



Add a workspace

Create a workspace in App Engine Studio (AES) to build a space that provides agents, case managers, help desk professionals, and managers with tools to help answer customer questions and resolve customer problems. Workspaces are primarily used for request and fulfillment processes, such as a service desk to manage tickets.

This video shows you how to perform the following procedure.

This video shows you how to add a workspace.

Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task

When you add a workspace you're creating a basic version of it, which you must then continue editing in Workspace Builder before it's ready to deploy. For more complex customizations, edit the workspace in UI Builder.

If you create a workspace from an App Engine Studio workspace experience template, you can edit the home page, lists, and record pages in Workspace Builder. However, if you create the workspace using an application template, you must edit some of its pages and objects in UI Builder.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, next to Experience, select **Add**.
4. Select **Workspace**, and then select **Begin**.
5. On the form, fill in the fields.

Field	Description
Name	Name of the workspace that appears to users. By default, the workspace shares the same name as your application. **Tip:** Use a name that uniquely identifies the workspace from other experiences. For example, `Office art requests workspace`.
Description	Description of the workspace.
URL	Unique web address of the workspace. By default, the workspace URL is based on the application name, but you can edit it.
Roles	User roles to limit who can access the workspace. ○ You must specify at least one role to grant access to the workspace. ○ To use a custom role for your workspace, you must create one in **Security** first. For more information, see Add application security.

6. Select **Continue** to define data for your workspace.
7. On the form, fill in the fields.

Field	Description
Primary Table	Data table that workspace users view or update.

Field	Description
Secondary Tables	Additional data tables that workspace users view and update.

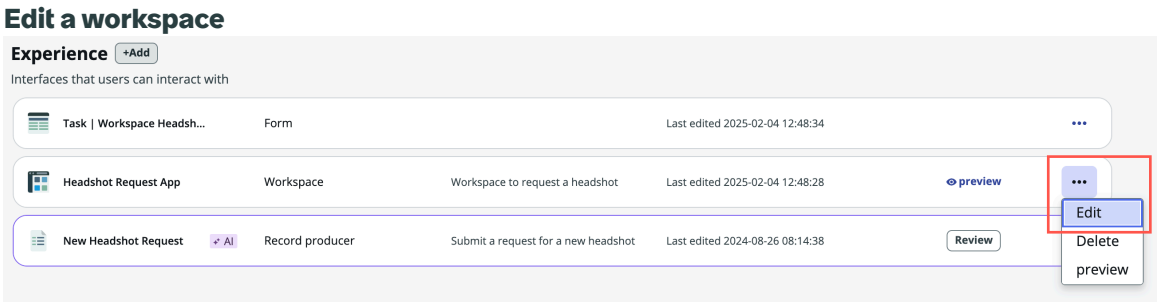
Note:
You can select only tables with default views. You can select from two sections: Tables that are already available in your app, and all tables outside the app scope.

Workspace Builder automatically generates a record page for the table you select.

8. Select **Continue**, and then select *Done*.
   After you create the workspace in AES, you must edit it in Workspace Builder by adding or configuring components on each page.

9. Next to the workspace you created, select the additional actions icon (•••) and then select **Edit**.

   You can also select the bar that lists the workspace in the Experience section of your app.



10. **Optional:** Edit your workspace home page in the In-line editor by selecting **Preview**.

    a. On your workspace home page, select **Edit**.
    If you don't see **Edit**, you may not have the permissions needed to edit the workspace home page.

    b. If the workspace is missing an Access Control List (ACL), enter the roles that should have access in the Add user roles to continue modal and select the **Add roles** button.
    For more information, see Configure workspace settings in Workspace Builder and Access Control List Rules.

    c. Adjust the home page elements and widgets by resizing, reorganizing, or reconfiguring them.

    Editing your workspace home page works similarly to editing a dashboard on your workspace. For more information, see Edit Platform Analytics dashboards.

    Note:
    If the workspace was created before Tokyo, you must edit it in UI Builder. See UI Builder for more information.

    If you created the workspace that contains a technical dashboard, Workspace Builder prompts you to **Open in UI Builder** when you try to edit the workspace.

    d. Select **Add new element** to add items to your workspace home page.

    e. Select **Exit Editing Mode** to stop editing in the In-line editor.

11. Edit the workspace in Workspace Builder. For more information, see Configure a workspace in Workspace Builder.

    You can also add more record pages to the workspace, if needed.

    Note:
    The default workspace record pages are read-only and can't be edited. To change the content of each record page, create a page variant and edit the variant.

Sample workspaces you can build

App Engine Studio (AES) provides a basic workspace that includes a home page, record pages, lists, an Analytics Center, and other functions.

The workspace that you create in App Engine Studio includes several default pages. You can use these pages as-is or edit them to suit your business needs.

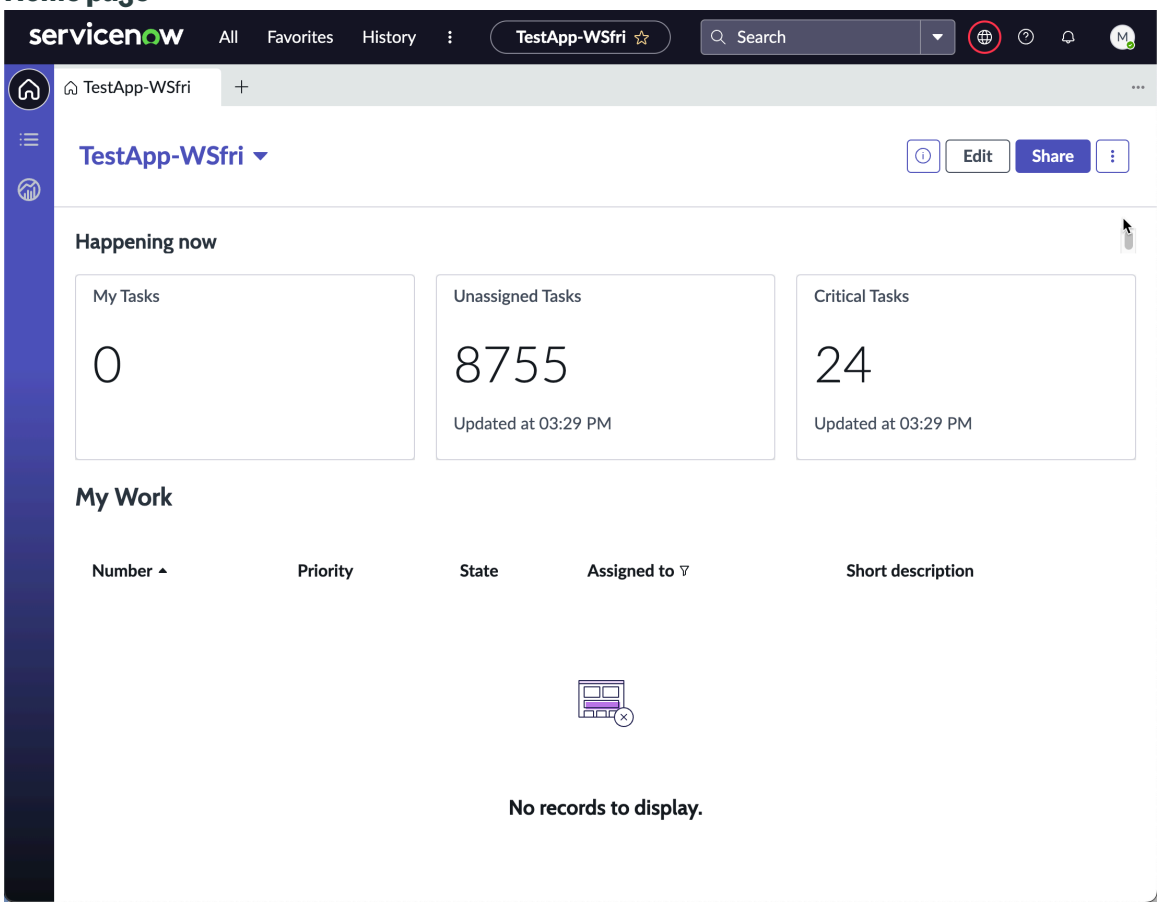
The following are some examples of the page customizations you can apply to the workspace after you add it:

- Home page
- List page
- Record page
- Search page
- Analytics Center

Workspace home page

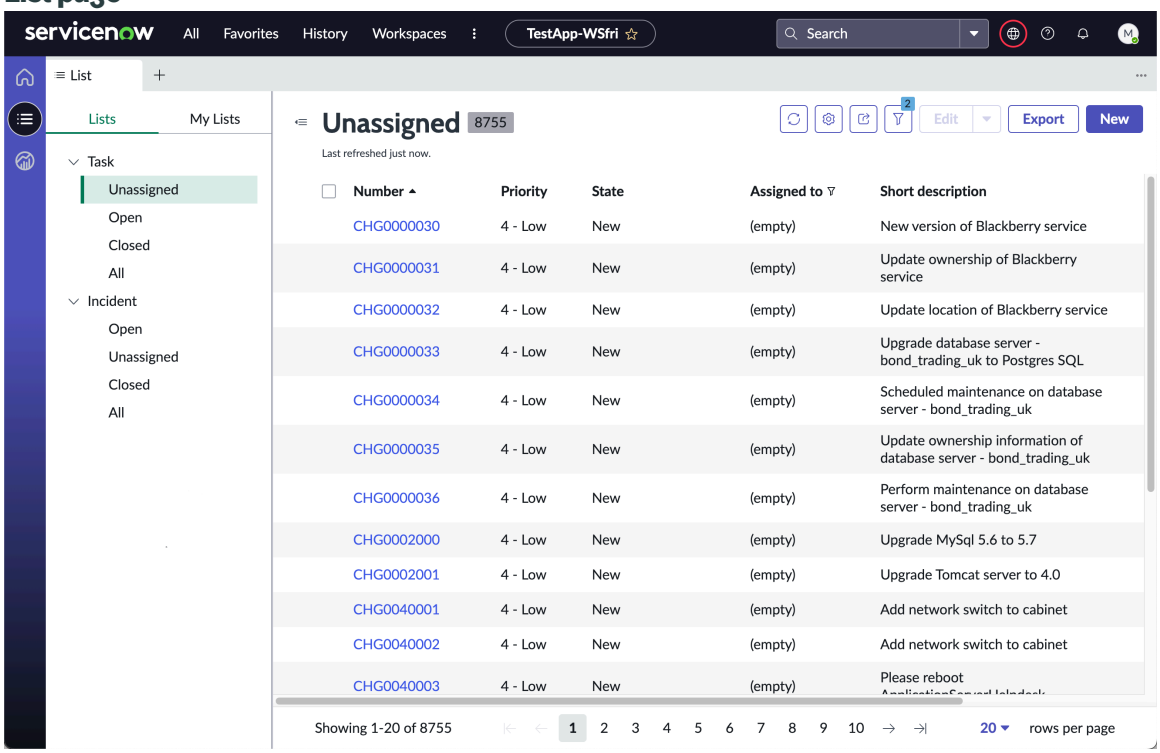
Note:
You can make changes to the home page, but editing all other pages requires creating a variant or copy of the page. If you edit the pages directly, then they won't get the latest code in a system upgrade.

Home page



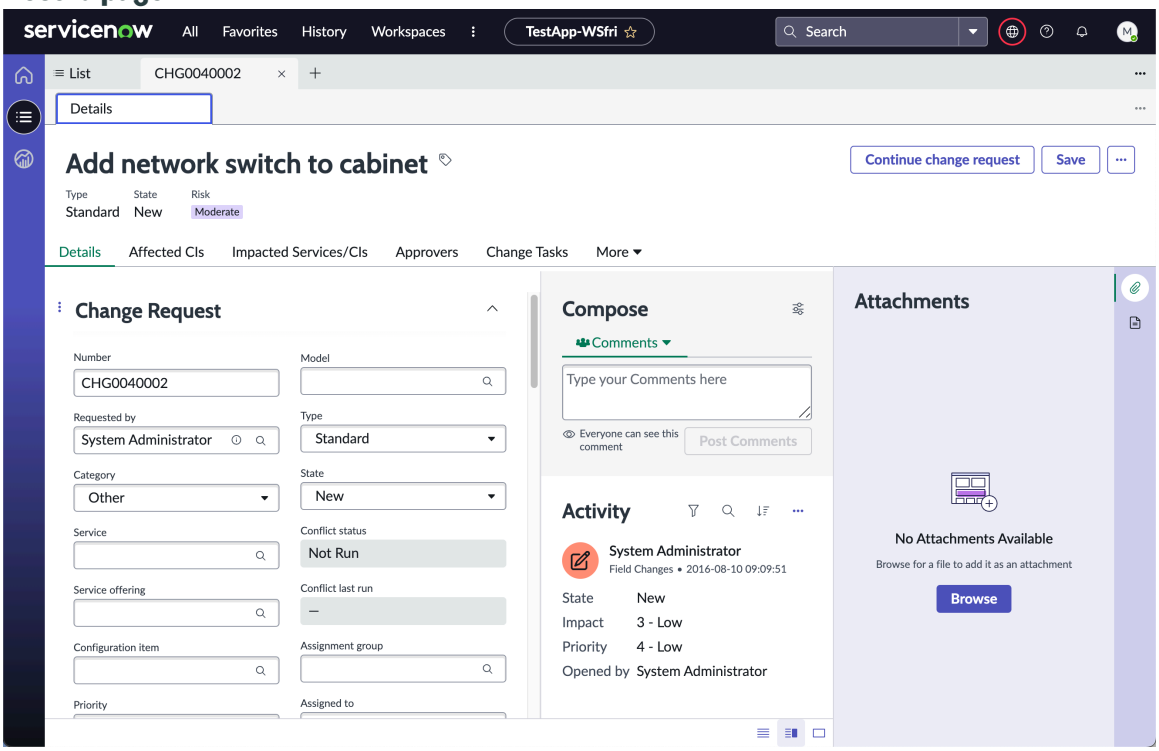
Workspace list page

List page



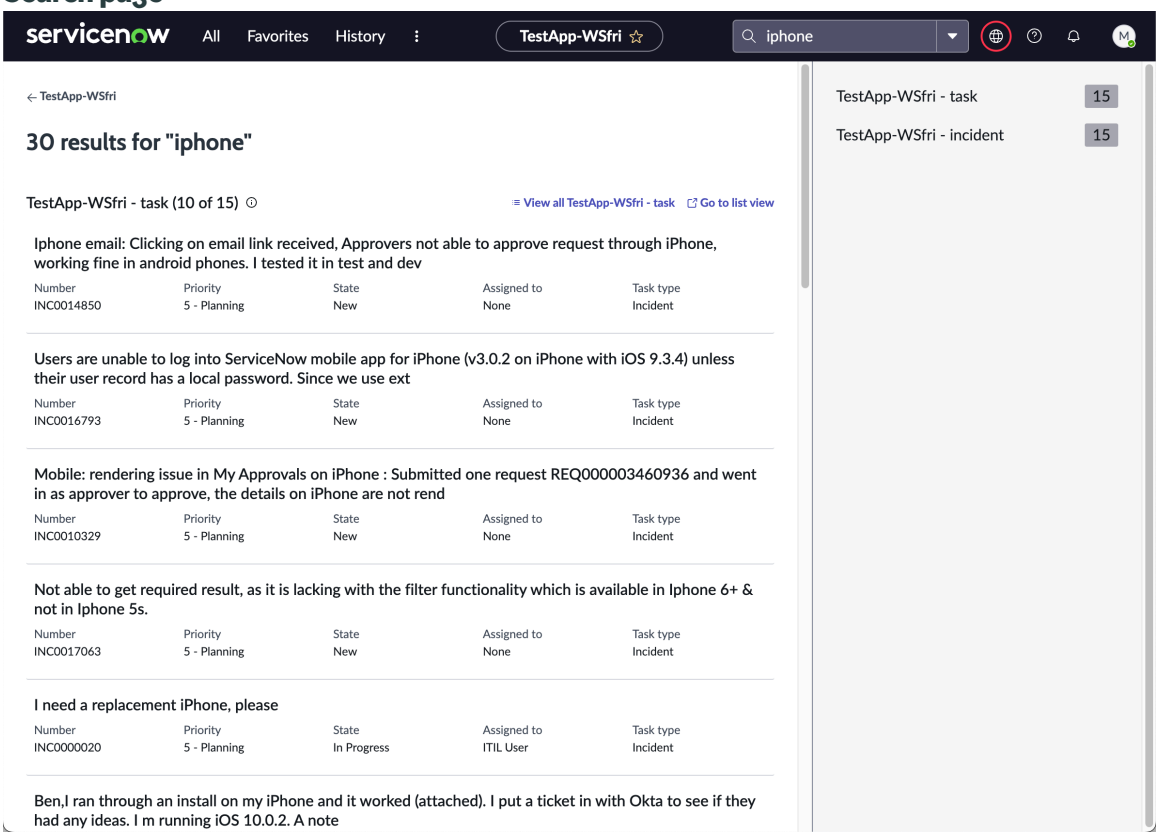
Workspace record page

Record page



Workspace search page

Search page



Workspace Analytics Center

The Analytics Center includes the following components:

Analytics Q&A

A field where you can ask natural language questions about the trends and health of your processes. (The Natural Language Query feature is required.)

Indicator Scorecard

A list of the Performance Analytics indicators that you are allowed to access based on your roles. Depending on the configuration, selecting an indicator can open it in KPI Details.

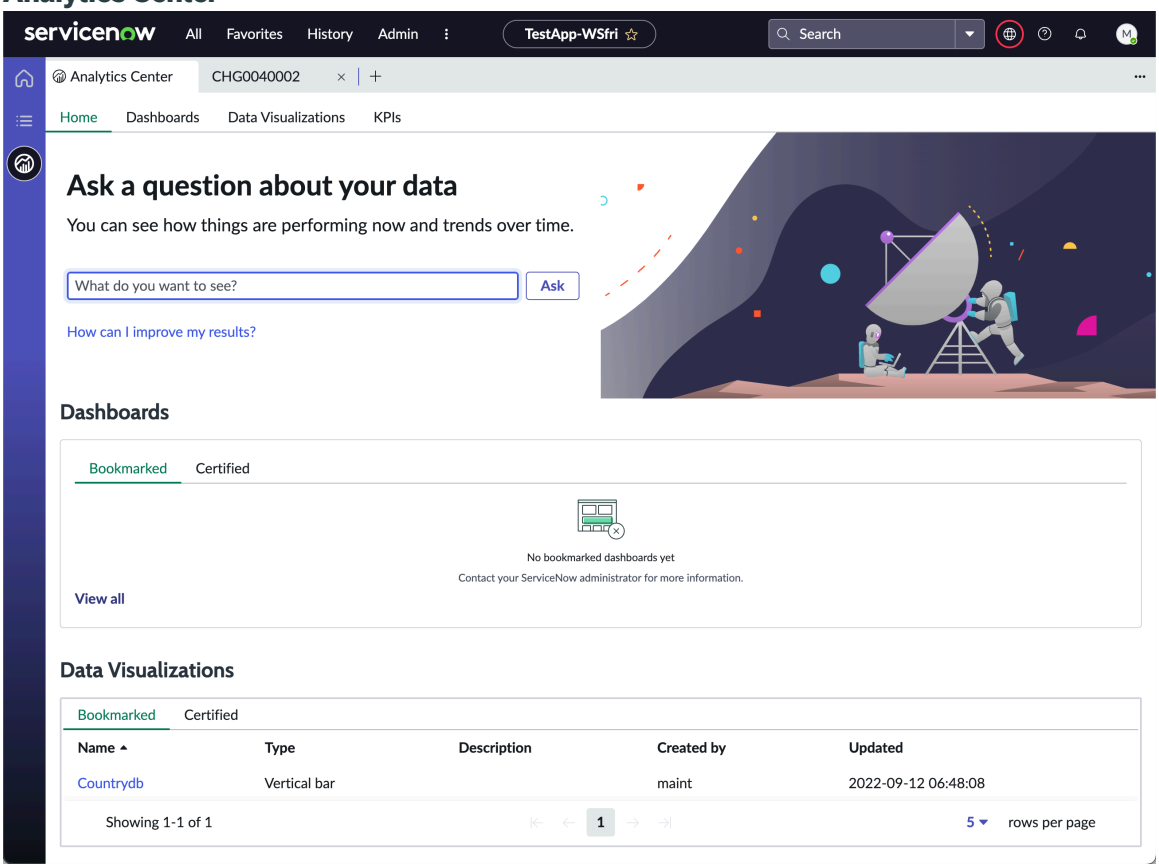
Dashboards Overview

A set of tiles for each dashboard on the instance that you are allowed to access. Selecting a tile opens that dashboard.

> **Note:**
> You can create and edit dashboards in-line. For more information, see Configure Platform Analytics dashboard details.

For more information on Analytics Center, see Analytics Center.

Analytics Center



Building workspaces in AES

Workspace Builder for App Engine is a streamlined, no-code environment that enables you to create a custom workspace from within App Engine Studio (AES) quickly and efficiently.

You can use Workspace Builder to quickly create workspaces for workspaces built starting with Xanadu. Workspace Builder enables you to edit the workspace home page, list pages, and record page contextual side panel directly in the Workspace Builder preview canvas.

For workspaces created before Xanadu, or if you need more robust functionalities and configurations, edit the workspace in UI Builder from a tab within AES.

Parts of a workspace

Build your workspace by including the following pages and elements.

Workspace pages and elements

Component	Definition
Home	Landing page for the workspace, which contains a dashboard and appears to users when first accessing the workspace. **Note:** The home page is an essential part of the workspace, and thus can't be hidden or removed. For more information on configuring the home page, see Configure a workspace home page in Workspace Builder. For information on configuring the basic settings for a workspace, see Configure workspace settings in Workspace Builder.
List	List pages for records from the specified tables, which help users to find individual records in a workspace. **Note:** If you remove the list page/route, you must use UI Builder to edit the workspace, not Workspace Builder. You can create a list category and then add filtered lists for each list category. Filtered lists apply unique conditions, ensuring that only the filtered subset of records appears. For more information on adding and editing lists in Workspace Builder, see Configure lists for a workspace in Workspace Builder.
Analytics	The customizable Analytics Center enables users to track and analyze records and usage with dashboards, data visualizations, and insights on your instance. **Note:** If Analytics Center page/route doesn't exist for a workspace, Workspace Builder is still available, but the Analytics Center tab doesn't appear in the workspace.

Workspace pages and elements (continued)

Component	Definition
	For more information on enabling the Analytics Center for a workspace, see Configure analytics for a workspace in Workspace Builder.
Record pages	Pages in a table that give users useful information to work on a task, issue, or incident. Useful information can include priority, state, activity, and so on. Create and customize record pages by using containers and components in different sections of the layout. You can also add a Playbook Experience to a record page. For more information, see Configure a record page for a workspace in Workspace Builder.

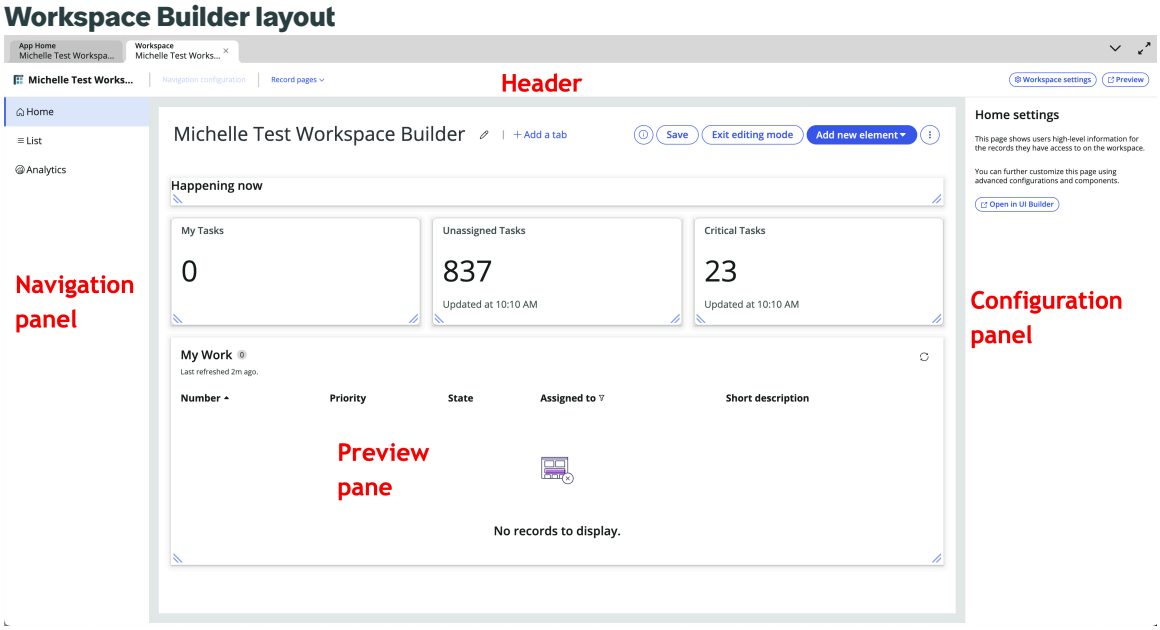
Note:
Your workspace must have a home page and lists for you to edit it in Workspace Builder. If they aren't present, you must edit the workspace in UI Builder.

Layout for Workspace Builder

The first time you access Workspace Builder, an onboarding modal displays a brief tour of what you can do in Workspace Builder.

Workspace Builder consists of the following sections:

- A left navigation panel to configure the pages that users can navigate to within the workspace.
- A preview of the workspace in the middle canvas.
- A right configuration panel for configuring the selected workspace component.
- An option to preview the workspace in a new browser tab.



The Workspace Builder header has the following navigation options just below the tab, where you can work with the following:

- Navigation configuration
- Record pages

Workspace Builder plugin

To use Workspace Builder, you must enable the com.devsnc_sn_workspace_builder plugin in the sn_ws_builder scope. It is a dependency of com.snc.aes.starter_workspace_template.

Configure a workspace in Workspace Builder

Edit a workspace in Workspace Builder to make customizations in App Engine Studio (AES).

This video shows you how to perform the following procedure.

Configure a workspace in Workspace Builder

Before you begin

Before you can configure a workspace in Workspace Builder, you must first create the workspace. See Add a workspace.

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

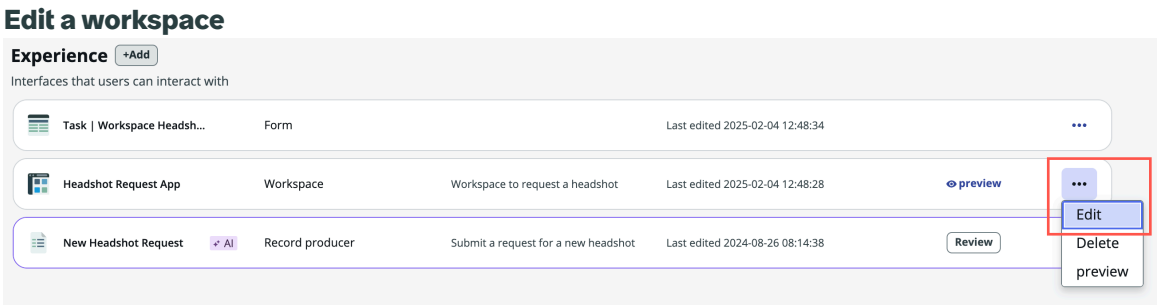
About this task

You can customize essential elements and components in Workspace Builder. For more complex configurations, or if you don't have the full entitlement for AES, you must edit the workspace in UI Builder.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the workspace you created, select the additional actions icon (•••) and then select **Edit**.

You can also select the bar that lists the workspace in the Experience section of your app.

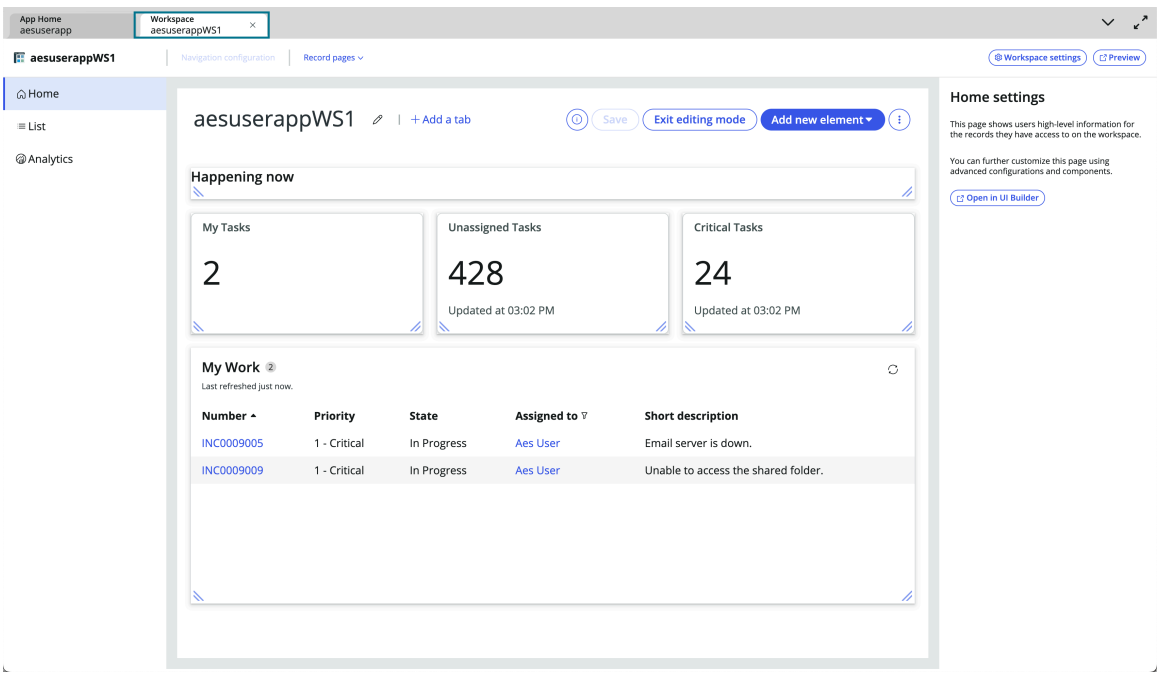


The Workspace Builder tab displays your workspace within AES:

- The left navigation panel is a list of pages or components, or building blocks of a page, that you can include in your workspace.

  Note:
  The home page is an essential part of the workspace, and thus can't be hidden or removed.

- The middle canvas is an in-line editing space for the home page for workspaces created in AES after Tokyo. For lists and other pages, the middle canvas is a preview of what you configure in the right configuration panel. You can preview workspace home pages built before Tokyo in Workspace Builder, but you must edit them in UI Builder.
- The right configuration panel is the configuration panel for working with selected components.



If you created the workspace that contains a technical dashboard, Workspace Builder prompts you to **Open in UI Builder** when you try to edit the workspace.

4. Edit the basic settings for the workspace by selecting **Workspace settings**.
   For more information, see Configure workspace settings in Workspace Builder.
5. Adjust the home page elements and widgets by resizing, reorganizing, or reconfiguring them.

For more information on editing a clickable home page, see Configure a workspace home page in Workspace Builder.

Note:
If the workspace was created before Tokyo, you must edit it in UI Builder. See UI Builder for more information.

6. Add or edit a list category and any subsequent filtered lists.
   For more information, see Configure lists for a workspace in Workspace Builder.
7. **Optional:** Enable an Analytics Center for the workspace.
   For more information, see Configure analytics for a workspace in Workspace Builder.
8. Add and edit a record page to configure the settings, tables, and related links that the workspace should support.

   When you create a record page, you're creating the metadata view, or shell, using a record page template for a type of record. You then edit the record page in Table Builder. Within Workspace Builder, you can only change the contextual side panel and the related items.

   For more information, see Configure a record page for a workspace in Workspace Builder.
9. Preview the workspace in a new browser tab by clicking **Preview**.

Result

If you made additional, more complex configurations to the workspace in UI Builder, such as custom additions to the contextual side panel or custom components, those changes may not appear in Workspace Builder.

Note:
Your workspace must have a home page and lists for you to edit it in Workspace Builder. If they aren't present, you must edit the workspace in UI Builder.

Configure workspace settings in Workspace Builder

Define workspace settings in App Engine Studio (AES) to control the basic functionality of the workspace, such as name and record page navigation.

This video shows you how to perform the following procedure.

This video shows you how to configure workspace settings in Workspace Builder.

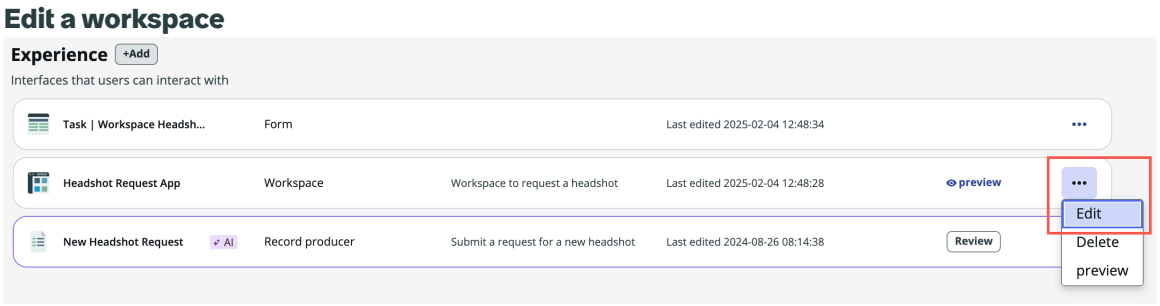
Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

Procedure

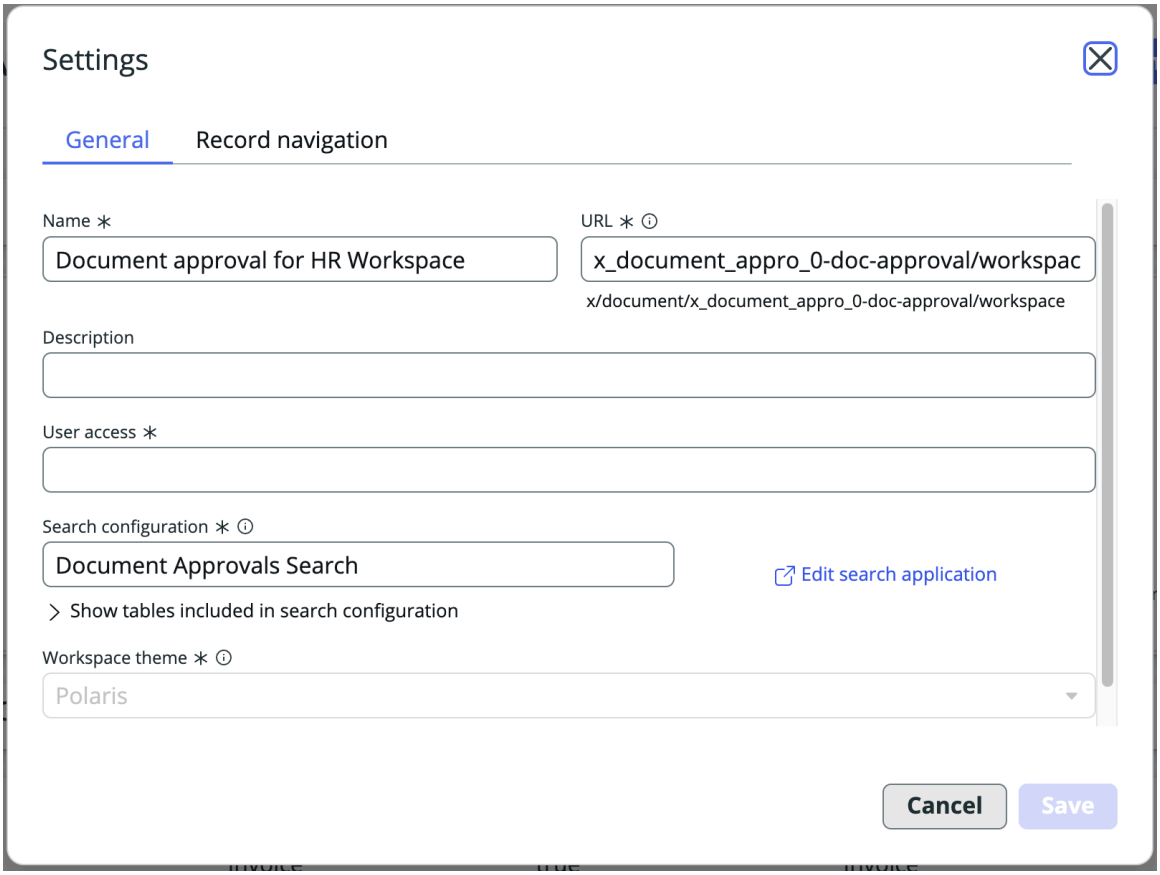
1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the workspace you created, select the additional actions icon (•••) and then select **Edit**.

   You can also select the bar that lists the workspace in the Experience section of your app.



The workspace appears in Workspace Builder.

4. Display the Settings form by selecting **Workspace settings** in the Workspace Builder header.



5. On the **General** tab of the form, specify or change the basic settings for the workspace by filling in the editable fields.

Note:
Some settings are specified when adding a workspace, and can't be changed.

General workspace settings

Field	Description
Name	Name for the workspace, which is visible to users.
URL	Relative URL for the workspace. The path is automatically created based on the name of your workspace, and is preceded by your

Field	Description
	instance URL. However, you can customize the path in this field.
Description	Brief description of the purpose of the workspace.
User access	People and groups that can access the workspace. To give unrestricted access to the workspace, leave this field empty.
Search configuration	Source of information when users perform a search in the workspace. You can reuse an existing search configuration by selecting the profile from the menu. Select the **Edit search application** link to make additional, advanced configuration changes. If you select an AI Search profile, the following features are available: ○ Selection of multiple facet values simultaneously, which you can apply using the condition builder. ○ Ability to view parent documents with linked attachments, which you can download. ○ Dark mode. **Note:** AI Search must be enabled for your instance to select an AI Search profile. For more information, see AI Search.
Show tables included in search configuration	Option that displays one or more tables that the workspace searches as sources.
Workspace theme	Theme to define the visual style of the workspace, which you can select only if Polaris isn't enabled for the instance. Themes are retrieved from the UX Themes [**sys_ux_theme**] table..

Trouble?
Add only one **User access** role at a time. For more information, see this Knowledge article.

6. Select **Save**.
7. Specify what type of records users can create by selecting the **Record navigation** tab of the form and then filling in the fields.

Record navigation workspace settings

Field	Definition
Navigation type	How users interact with or navigate the workspace.

Field	Definition
	Tabbed navigation creates separate tabs for each page, enabling users to work on multiple records at a time. Choose tabs if you anticipate that users must change between records quickly. Tabs also enable users to create records directly from the navigation. **Note:** The default navigation type for workspaces built in AES is tabbed, and can't be changed after the workspace is created. You must create the workspace in UI Builder to specify breadcrumb navigation. You can't change the navigation type after the workspace is created.
Table	Data table that is added to when a user creates a record. The table you select automatically has associated lists. **Note:** The tables specified when the workspace was created appear automatically. Select **Add a table** to incorporate additional tables into the workspace. You can add each table only once. To remove a table, select its delete icon ( ). No confirmation message appears, but you can select **Cancel** to revert the deletion.
Label	Brief label for the option that appears when users select the new record menu. Specify a label that tells users what the record is used for.

8. Select **Save**.

Configure a workspace home page in Workspace Builder

Edit the home page for a workspace in App Engine Studio (AES). Adjust what users see when first accessing the workspace, which is a dashboard.

This video shows you how to perform the following procedure.

This video shows you how to configure a workspace home page in Workspace Builder.

Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task

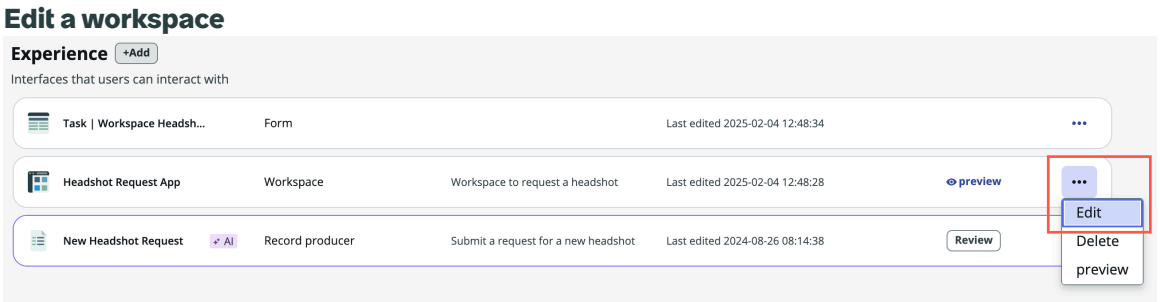
You can edit workspace home pages in Workspace Builder only for workspaces created in Tokyo and going forward. You can edit older workspaces in UI Builder.

Note:
The home page is an essential part of the workspace, and thus can't be hidden or removed.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the workspace you created, select the additional actions icon (•••) and then select **Edit**.

   You can also select the bar that lists the workspace in the Experience section of your app.

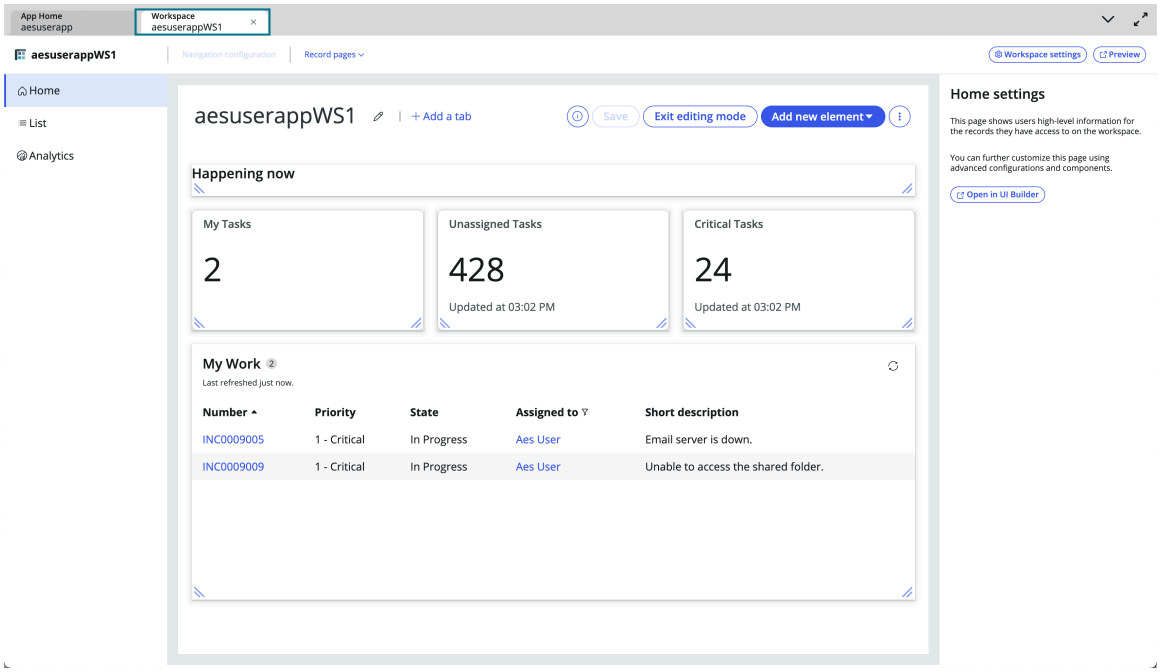


   The home page for the workspace appears in Workspace Builder, where you can drag elements and components to design the page.

   If you created the workspace that contains a technical dashboard, Workspace Builder prompts you to **Open in UI Builder** when you try to edit the workspace.

4. Add an element to the page.
   You must be in the **Navigation configuration** tab. If you're already in the **Navigation configuration** tab, you can't select it.

a. In the navigation panel, select **Home**.



b. In the home page canvas, select **Add new element**.

c. From the list, select the element or component that you want to add.

For example, to add a graphic of data from the specified table, select **Data visualization** and choose a **New data visualization** or a **Saved data visualization**.

You can add the following elements to the home page in Workspace Builder.

Home page elements

Element	Description	More information
Data visualization	Graphic, visual representation of current instance data. You can create a new data visualization or repurpose an existing data visualization by choosing it in the Select visualization from library modal and selecting **Add to dashboard**. In addition to data visualizations, you can select the **Library** element to add an already-saved data visualization to the home page.	▪ Data visualizations in Platform Analytics ▪ Create a single score data visualization in the Visualization Designer or on a dashboard ▪ Developer site data visualization component listing

Element	Description	More information
Filter	Refine the visualizations on a dashboard based on specified criteria. The filters can apply to both tables and indicators.	▪ Filters in Platform Analytics ▪ Developer site filter component listing
Heading	Title or text for the top of the home page or section.	Developer site heading component listing
Image	Static or animated images or pictures for the workspace.	Developer site image component listing
List - Simple	Table data that you can customize for the dashboard audience.	▪ Simple List widget or List - Simple Usage - Developer site ▪ Developer site List - simple component listing

The Workspace Builder components for home pages has details on home page elements. For more information, see Platform Analytics dashboard overview.

If you want to include additional elements that aren't available in the **Add new element** menu, you must edit the workspace in UI Builder. For a complete list of elements you can add in UI Builder, see the components documentation on the ServiceNow Developer Site.

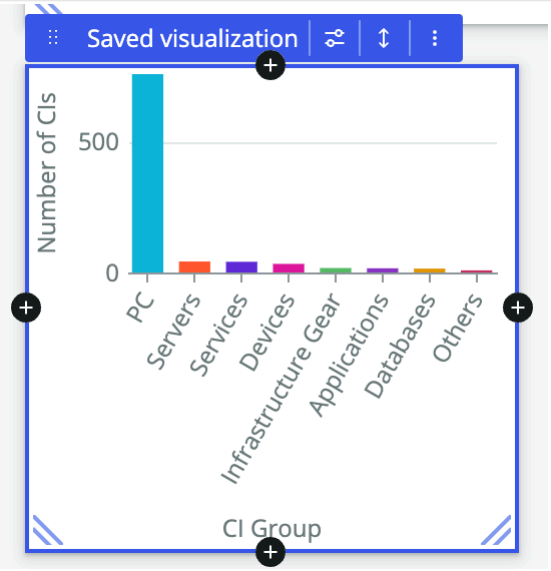
The element appears on the dashboard, where you can make edits.

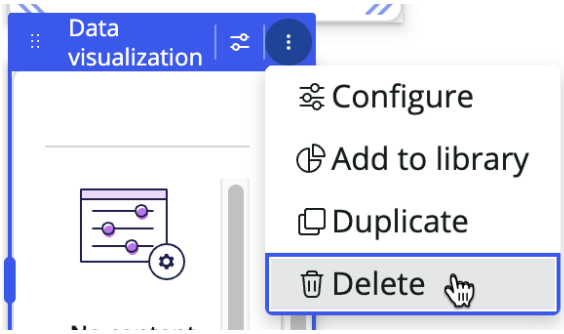
d. Drag the element to move or resize it.

e. In the canvas header, select **Save**.

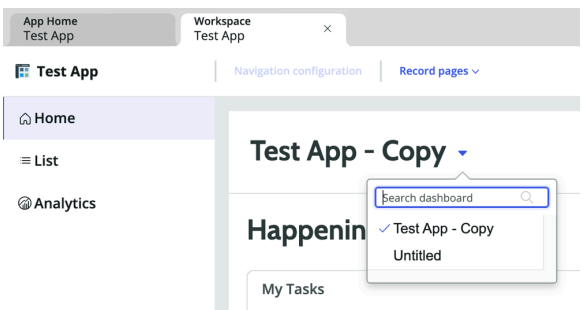
5. **Optional:** If needed, complete any additional changes to an element.

Option	Description
Edit an element	**a.** Select the element that you want to edit, such as a data visualization. (Optional) A header appears with the name of the element type.

Option	Description
	 **b.** Select the settings icon ( ) in the element header to open the configuration panel within the canvas. **Note:** You can also select the context menu icon ( ) and select **Configure** to edit the element. **c.** Edit the configuration of the element in the configuration panel as needed. (Optional) For example, if you're editing a data visualization, you must specify the source table. For more information on edit ing components, see Add and configure components. **d.** In the canvas header, select **Save**.
Copy an element to reuse on the home page	Select the context menu icon ( ) and select **Duplicate** to copy the element. You can then make additional changes to the copied ele ment.
Add an element to the library for reuse	Select the context menu icon ( ) and select **Add to library** to include the element in your library to reuse on other dashboards.
Delete an element from the workspace dash board	**a.** Select the element that you want to delete. (Optional) A header appears with the name of the element type.

Option	Description
	b. In the header, select the context menu icon ( ) and select **Delete**.  **Note:** There's no confirmation message. The element disappears from the dash board.

6. **Optional:** If needed, complete any additional configuration changes to the dashboard.

Option	Description
Configure the dashboard layout	Select and drag an entire element to move it around the dashboard layout. (Optional) Resize an element by selecting and dragging the handlebars ( ) in the low er corners of the element.
Duplicate the dashboard	To make a copy of the dashboard: ○ Select the more options icon for the dash board ( ). ○ Select **Duplicate**. ○ Enter the **New name** and **Description** for the copied dashboard on the modal. ○ Select **Duplicate**. Access the duplicated dashboard by exiting out of editing mode and selecting the new dashboard from the dashboard name drop-down menu. 

Option	Description
Add tabs to the workspace and edit them as needed	**a.** Add a tab to the home page by selecting **Add a tab**. When you add your first tab to a home page dashboard, a new part of the layout ap pears below the title. Elements you add to the tab are visible no matter which tab the user is on. A tab labeled **untitled** is added next to any existing tabs. **b.** Rename a tab by selecting the tab name, selecting the edit icon ( ), typing the new name, and pressing **Enter**. **c.** Reorder a tab by selecting the tab and drag ging it to the new position. **d.** Delete a tab by selecting its delete icon ( ) then selecting **Delete**. **Note:** No confirmation message appears. The tab disappears from the dash board.
Preview a workspace and make in-line edits	Make basic edits to a workspace when pre viewing it using the In-line editor. **a.** In the Workspace Builder header, select **Preview**. **b.** In the preview that appears in a new brows er tab, select **Edit**. **c.** Adjust the home page elements and wid gets by resizing, reorganizing, or reconfigur ing them. For more information, see Edit Platform Ana **d.** Select **Add new element** to add items to your workspace home page. **e.** Select **Exit Editing Mode** to stop editing in the In-line editor.
Open a printer-friendly version of the dash board	Select the more options icon for the dash board ( ) and select **Printer friendly** to dis play the workspace home page in a format suitable for printing.
Add a bookmark to Analytics Center	Select the more options icon for the dash board ( ) and select **Add to bookmarks** to include a bookmark for the workspace in the Analytics Center Bookmarks section.

Option	Description
Make additional or more advanced configurations to the home page	In the configuration panel, select **Open in UI Builder**.
Delete the dashboard	Select the more options icon for the dashboard ( ) and select **Delete**. You must select **Delete** again to conform. **Note:** Workspaces should have at least one dashboard home page.

7. Share the dashboard.

   a. Select the more options icon for the dashboard ( ) and select **Share**.

   b. On the modal, fill in the fields.

   Share dashboard options

Option	Description
Grant access to	Users and groups that can access the dashboard.
Allow recipients to add, edit, or delete sharing permissions associated with this dashboard	Option that enables the users and groups you share the dashboard with to add, edit, and delete the sharing rights for any user, group, or role.
Add as viewer / Add as editor	Option to specify read-only or write access for the dashboard.
Manage access	Roles that can access the dashboard. Search for and select a role in the search box. For each role with access, select the access level: ▪ **Can edit** ▪ **Can view** ▪ **Can share**
Copy link with filter	Button that copies the dashboard URL for the specified users and roles with the currently applied filters.
Copy link	Button that copies the dashboard URL for the specified users and roles without any filters.

   c. Select **Confirm**.

8. In the Workspace Builder header, select **Save**.

Workspace Builder components for home pages

Several UI Builder components are available as elements to build home pages and contextual side panels in Workspace Builder. Workspace Builder helps you quickly create workspaces in App Engine Studio (AES).

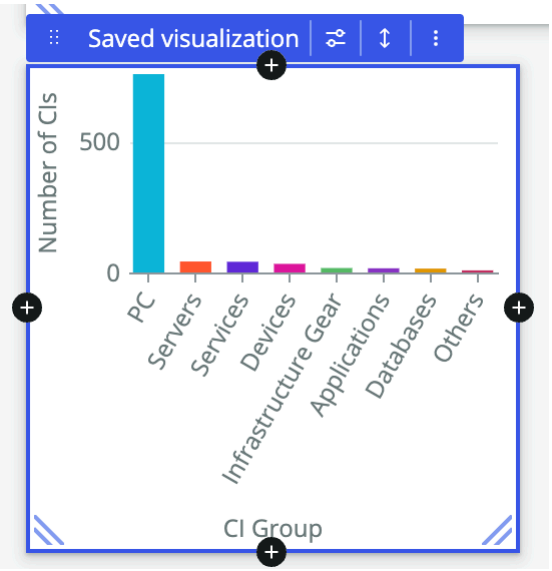
For complete details on all elements and components, see the ServiceNow® Developer site.

Data visualization components

The data visualization component is a container for available chart types for use in Workspace Builder home pages. Select the chart type to display data in different ways depending on your application. You can create a new data visualization or repurpose an existing data visualization by choosing it in the Select visualization from library modal and selecting **Add to dashboard**.

For more information on data visualization components, see the following topics and sources:

- Data visualizations in Platform Analytics
- Create a single score data visualization in the Visualization Designer or on a dashboard
- Data visualization component listing on the Developer site



Filter components

Filters enable you to filter data visualizations without modifying the visualizations. Add a filter element when working with the home page in the canvas. Then select a table, field, and other properties to configure the filter. When a user selects one or more values of the field, the filter applies to those data visualizations that are based on the same data source.

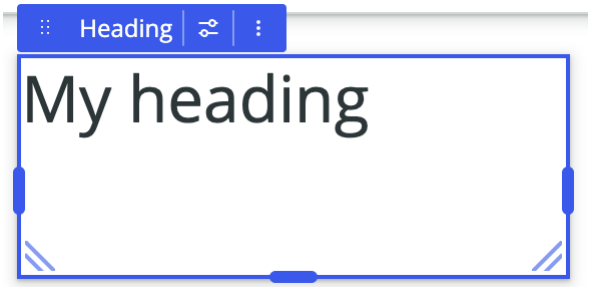
For more information on using filters components to build a workspace, see the following topics and sources:

- Next Experience filters
- Developer site filter component listing

Heading components

Heading components are available in the home pages for workspaces. Add a title to your landing page using a heading component.

The following image shows a "My Heading" heading.

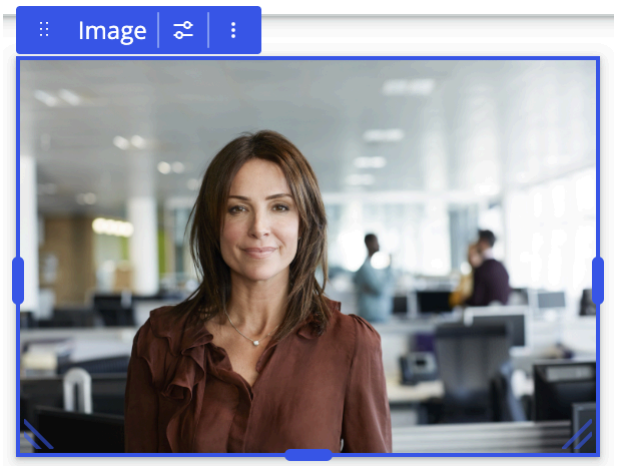


For more information on heading components, see Heading component listing on the Developer site.

Image components

Add images to your landing page with an image component.

When you drag the image component icon (Image) into the preview pane, a stock image appears that you can change and resize. You can change the stock image by supplying your image's URL.

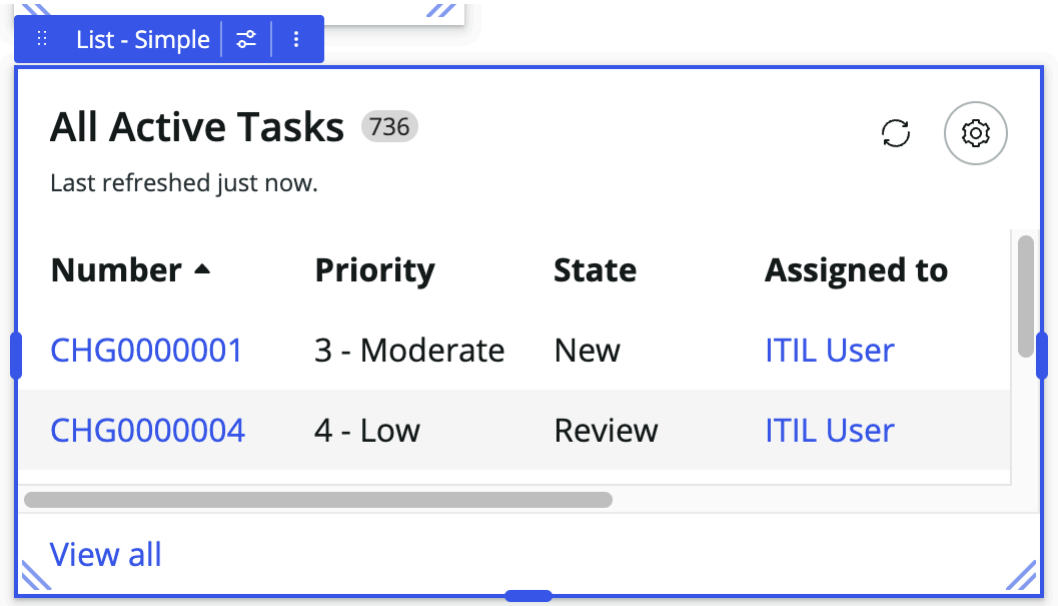


For more information on image components, see the Image component listing on the Developer site.

List components

Use a list component to add lists to your landing page.

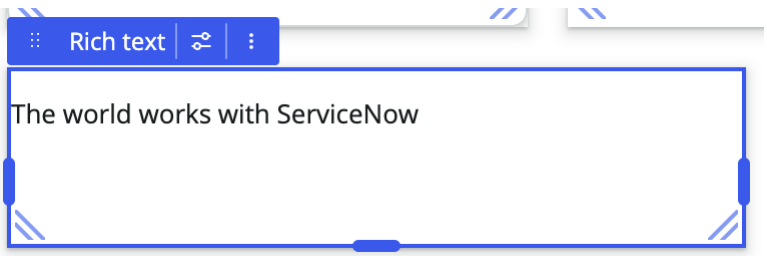
When you drag the list component icon (List) into the preview pane, a sample list appears.



For more information on list components, see the List component listing on the Developer site.

Rich text components

Add text to your landing page with a rich text component to explain parts of your landing page, for example, "Tackle these priority 1 incidents first."



For more information on rich text components, see the Rich text component listing on the Developer site.

Configure a record page for a workspace in Workspace Builder

Configure a record page for a workspace in App Engine Studio (AES). Use containers and components to guide a user through an experience. Each record page is linked to only one data table.

Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task

Components are the building blocks of your page. Add components to your page to build or customize your workspace experience. For example, add a button component to your page that lets users submit requests.

A container is a type of component that can contain other components. Containers have layout properties applied to them. Change the page layout and visual styling to make the record page a unique experience.

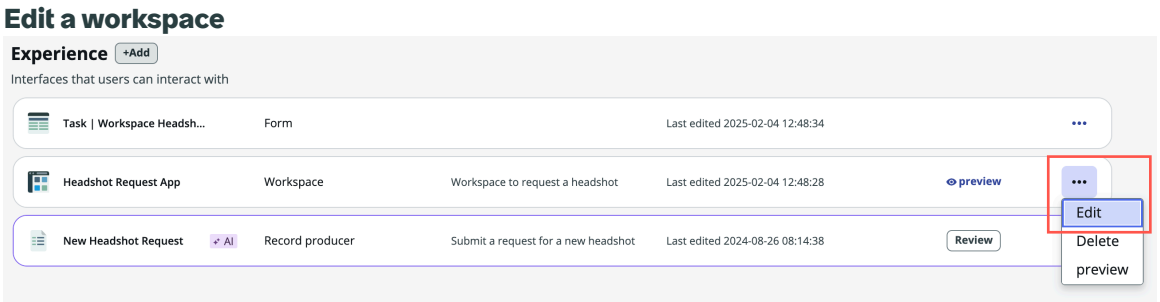
Note:
Starting with workspaces built after the AES Workspace UI Template v22.0.3 release, you can work with record pages in Workspace Builder. For older workspaces, you must edit the record pages in UI Builder.

If you don't see the record page for a table you configured for the workspace, see this Knowledge article for more information.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the workspace you created, select the additional actions icon (•••) and then select **Edit**.

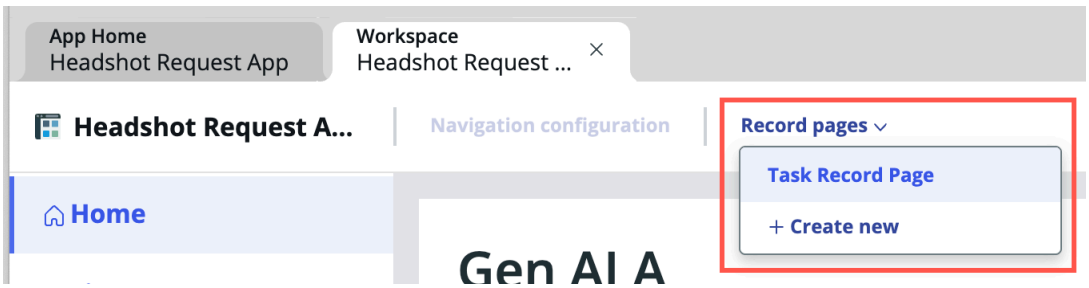
   You can also select the bar that lists the workspace in the Experience section of your app.



   The workspace appears in Workspace Builder.

4. Create a record page.
   AES automatically generates record pages for the tables that you selected when you created the workspace.

   a. In the Workspace Builder header, from the **Record pages** menu, select **+ Create new**.



   b. On the form, fill in the fields.

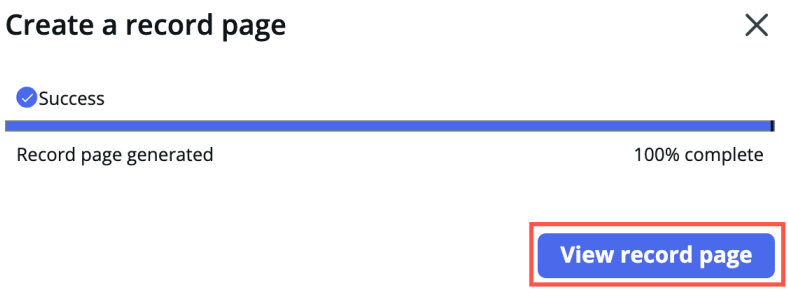
Record page settings

Field	Description
Name	Brief name that identifies the purpose of the record page.
Table	Table that the record page is linked to and updates. The following rules apply:

Field	Description
	▪ If you're using a table that you or someone at your organization created, the table must contain at least one new field or column. Otherwise, the form view and the record for the table can't be created. ▪ You can link each table to only one record page. You can select from two sections: Tables that are already available in your app, and all tables outside the app scope.

c. Select **Create**.

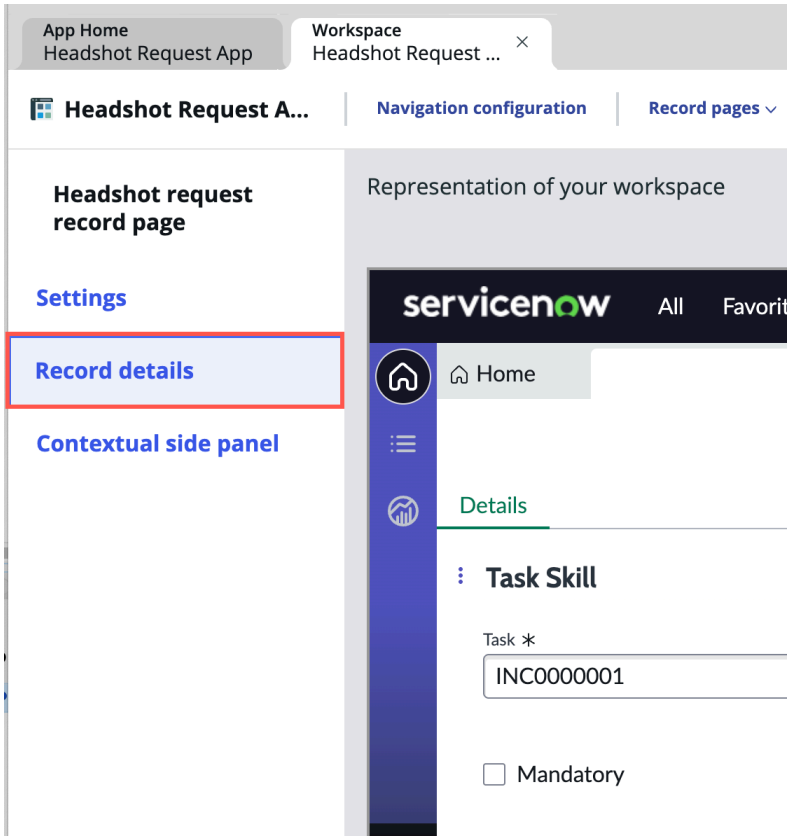
d. Select **View record page**.



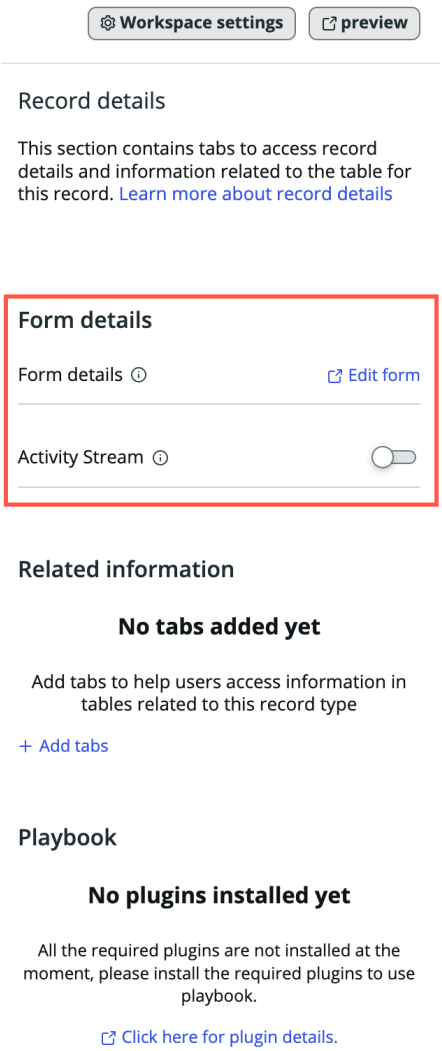
The page appears in the preview panel. You can select the page from the **Record pages** menu.

5. Edit the record details for the record page.

a. In the navigation panel, select **Record details**.



b. In the **Form details** section of the configuration panel, select **Edit form** and fill in the fields.



Form details for record page

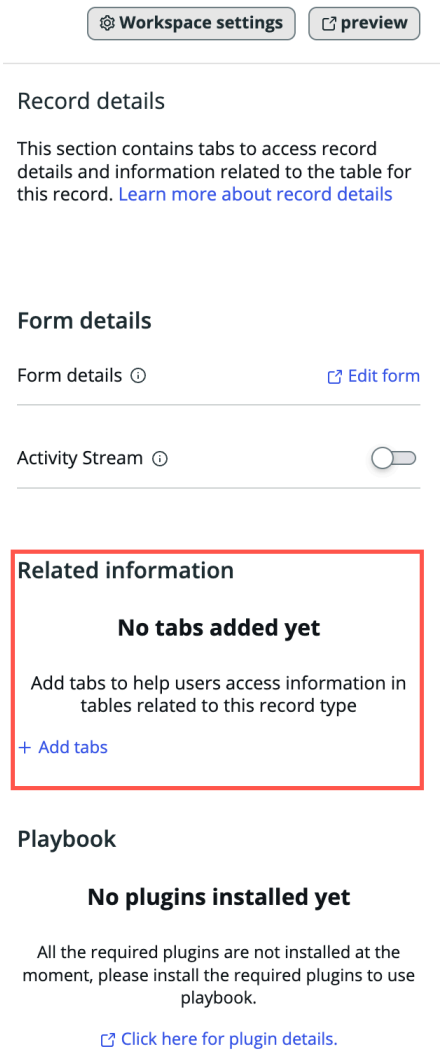
Field	Description
Form details	Select the **Edit form** link to edit the form in a Table Builder tab inside AES. **Note:** After you leave the record page in the Table Builder tab, a dialog prompts you to **Reload this page**. For more information on editing forms for a record page, see Forms in Table Builder.
Activity stream	Option to enable the Activity stream for the record page. The Activity stream enables your workspace users to communicate with requesters and

Field	Description
	make internal notes about the work done on a record. For more information on the Activity stream, see Activity streams.

6. Configure the related information in the Related information section of the configuration panel for the record page.
   Related information includes tables that are related to the record, which appear as tabs, such as Child Incidents. For more information on configuring related information, see Related lists.

   Note:
   You can't create a table for the related information in a record page. The table must exist in ServiceNow AI Platform.

   a. If no related lists exist yet, add a new related list by selecting **+ Add tabs** in the **Releated information** section.



   If related lists exist for the record page, select the **Manage related information** link in the configuration panel.

b. In the form that appears in a new tab within AES, move the related list that you want to appear as a tab from the **Available** panel to the **Selected** panel.

c. Use the arrow buttons to move the selected lists into the order you want them to appear.

d. Select **Save** and return to the Workspace Builder tab within AES.

e. View the changes that you made by selecting **Reload this page** in the dialog box that appears.

Each related information component has an information icon (ⓘ) that, when selected, displays its related table.

7. Provide fulfillers with visibility into cross-business workflows and the actionable tasks used to complete these workflows by specifying the playbook.
For more information on playbooks, see About Playbook Experience.

> **Note:**
> You must have the Playbook Experience plugin installed. If you used a template to create the app, you can't add a playbook to the record page. For details on how to add a playbook for workspaces created from templates, see this knowledgebase article.

a. In the Playbook section of the configuration panel, select **Add a playbook**.

b. On the form, fill in the fields.

Playbook details

Field	Description
Tab name	Label for the record tab that contains the playbook.
Playbook experience	Existing playbook to add to the record. **Note:** You can choose only from the playbooks that are available for the table.

c. Select **Add**.

d. Update the playbook by selecting the settings icon (⚙), making the change, and then selecting **Update**.

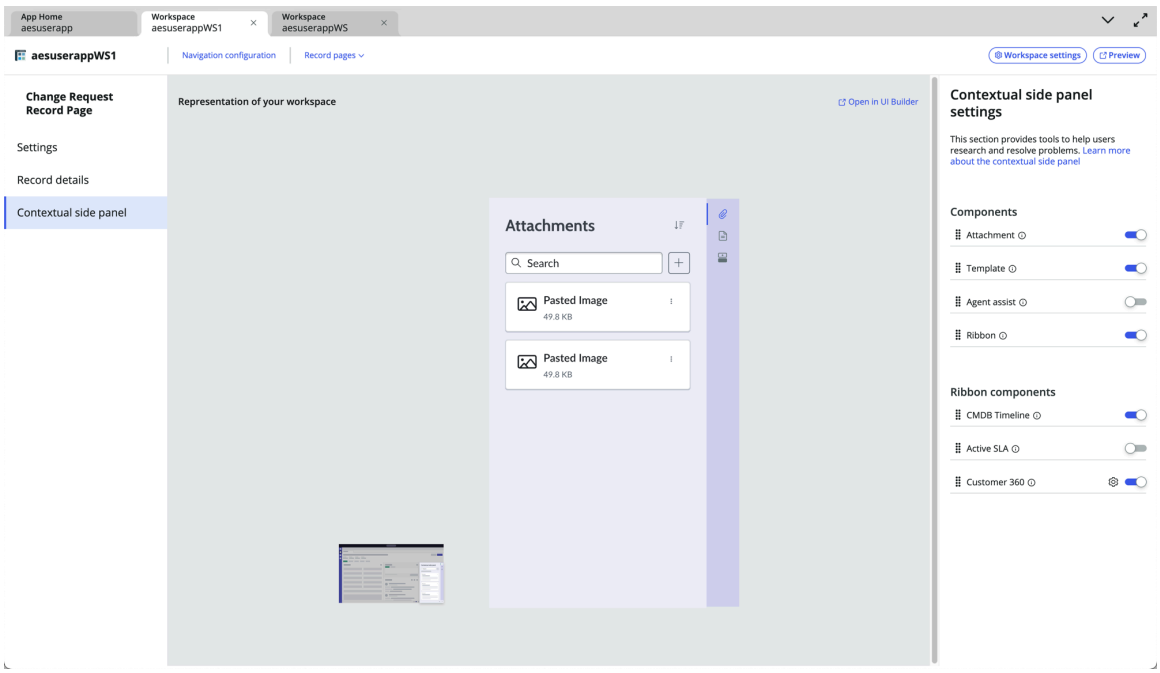
> **Note:**
> If you want to make additional changes to the playbook, you must open and edit the playbook in Playbooks.

e. Delete a playbook from the workspace by selecting its delete icon (🗑) and selecting **Delete** on the confirmation dialog.

8. Configure the contextual side panel for the record page, which contains the icons for the tools that appear on the right side of the page.
   The contextual side panel contains icons that enable agents to research and resolve issues using various ServiceNow AI Platform tools. If your organization created any custom right tab components in UI Builder, you can add them as well. For more information, see Tabs sidebar.

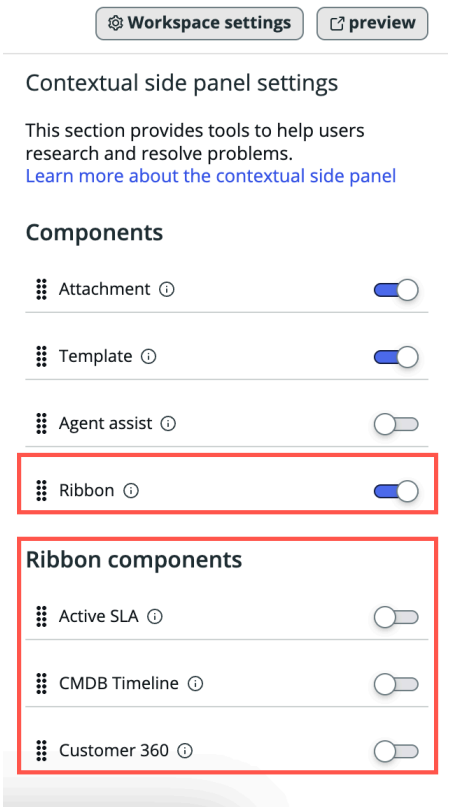
   Note:
   If you used a template to create the app, you can't add a contextual side panel to the record page.

   a. In the navigation panel for the record page, select **Contextual side panel**.



   b. In the **Context side panel settings**, select the toggle switch for each component and ribbon component that you want to appear on the record page.
   For example, you can include Agent assist and attachments for the record. Attachments and templates are enabled by default, though you can disable them by deselecting their toggle options.

   To see the ribbon components, set the **Ribbon** component toggle to on.



To add pre-existing custom components, edit the workspace in UI Builder.

Contextual side panel components

Tool	Type	Description	More information
Active SLA	Ribbon component	Display active Service License Agreements (SLAs) for the case, including time remaining, the SLA state, and any breaches.	SLA ribbon
Agent assist	Component	Display automatic search results that show possible solutions for records that agents open.	Agent Assist
Attachment	Component	Add attachments to the record, such as images and PDFs.	Attachments
Configuration Management Database (CMDB) Timeline	Ribbon component	Display a chronological summary of case activities, including state changes and interactions between the agent and the	Timeline ribbon component

Tool	Type	Description	More information
		requester (usually the customer). The ribbon also displays how much time that the fulfiller and requester spent on the case.	
Customer 360	Ribbon component	Display customer information, such as the contact or consumer name, email address, and phone numbers.	Customer 360 ribbon component
Ribbon	Component	Display the contextual side panel ribbon, which contains tools to help agents resolve issues. Ribbon components can have additional components added to them.	Viewing ribbon information in the contextual side panel
Template	Component	Display templates that help reduce the time agents spend responding to requesters by automatically filling in record fields.	Form templates

The selected tools appear in the record page preview canvas.

c. In the configuration panel, drag the components to rearrange them.

d. Optional: Complete any additional configuration for components by selecting the settings icon (⚙).
For example, Customer 360 requires that you specify additional settings before the component works on the record page.

Workspace Builder automatically saves the changes to the contextual side panel.

9. Complete any additional changes and configurations.

Option	Description
Update the name of the record page	**Note:** You can't change the table that a record page is linked to after you create the page. To change the table for the record page, you should create a separate record page. **a.** From the Record pages menu, select the page that you want to edit. A preview of the record page appears. **b.** In the navigation panel, select **Settings**. **c.** In the Record page settings configuration panel that appears, enter the new page name. **d.** Select **Save**.
Preview the record page in a new browser tab	Select **Preview** in the Workspace Builder header. The record page appears in the workspace. If no data exists for the table, AES displays a page where you can add data to the table.
Delete a record page	**a.** From the Record pages list, select the page. **b.** In the navigation panel, select **Settings**. **c.** At the bottom of the configuration panel, se lect **Delete record page**. **d.** On the confirmation dialog, select **Delete**.

Configure lists for a workspace in Workspace Builder

Create list categories in App Engine Studio (AES) to add pages that list table records. You can add filter conditions and change columns to create variations on the list.

Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task

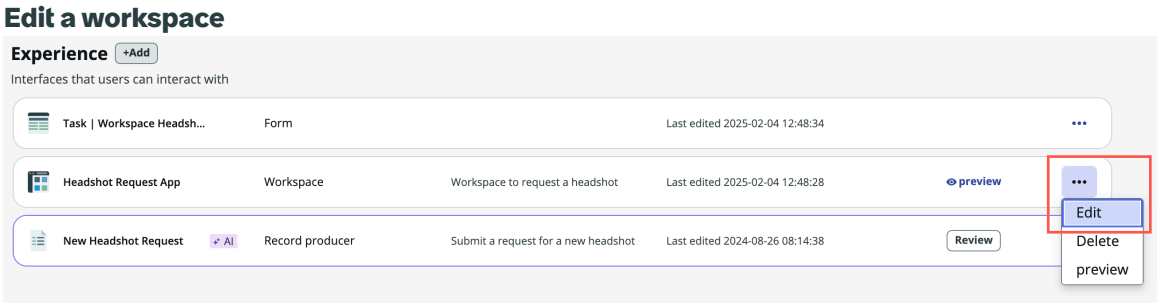
Note:

- You can launch and view workspaces in the Workspace Builder. However, you can only edit workspaces created through ServiceNow Studio and App Engine Studio within those specific tools.
- If you remove the list page/route, you must use UI Builder to edit the workspace, not Workspace Builder.

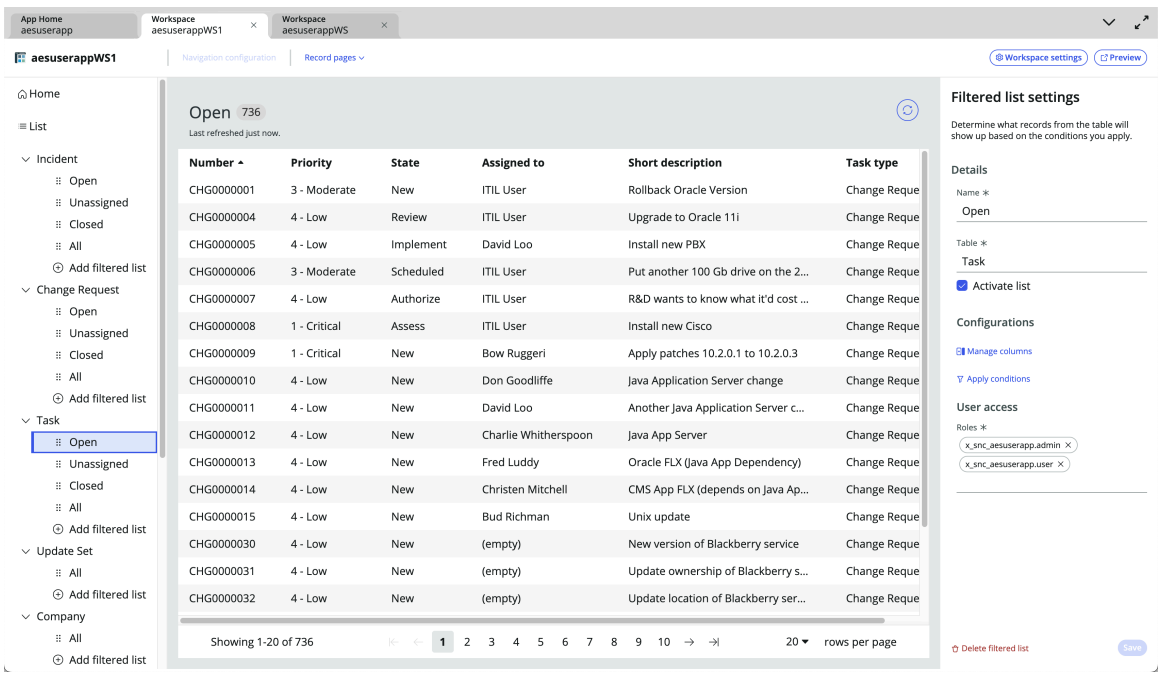
Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the workspace you created, select the additional actions icon (•••) and then select **Edit**.

   You can also select the bar that lists the workspace in the Experience section of your app.



4. If they aren't already activated, activate lists to enable them for the workspace.
   You must be in the **Navigation configuration** tab. If you're already in the **Navigation configuration** tab, you can't select it.
   a. In the navigation panel, select **List**.
   b. In the list, select **Task** > **Open**.
   c. In the configuration panel, select the **Activate list** option.



5. Add a list category.
   a. In the navigation panel, select **Add list category**.
      If the table associated with the workspace isn't a core table, you must add new list categories.
   b. On the form, fill in the fields.

List category settings

Field	Definition
Name	Name to identify the list category.
Description	Brief description of the contents of the list category to help developers understand the purpose. **Note:** Workspace users don't see the description.

c. Select **Add**.

6. **Optional:** To rearrange the order in which list categories appear, drag them to a new position.
7. Add a filtered list to a list category.
   For example, in the list category for the Task table, you could create separate filtered lists for Opened, Unassigned, and Closed tasks.

   a. In the navigation panel, expand the list category you want to create a filtered list for.

   b. Select **Add filtered list**.

   c. On the form, fill in the fields.

   Filtered list settings

Field	Definition
Name	Name to identify the filtered list.
Table	Table that contains the subset of records that appear in the filtered list.You can select from two sections: Tables that are already available in your app, and all tables outside the app scope.

   d. Select **Add**.

   The preview canvas displays the contents of the filtered list.
8. Refine how the filtered list displays its content by adding filter conditions to narrow what it displays.
   You can make additional configurations to the filtered list.

   a. In the navigation panel, select the filtered list to edit.

   b. In the configuration panel, specify the additional settings.

More filtered list settings

Field	Definition
Activate list	Option to enable users to work with the filtered list.
Manage columns	Option to select and arrange the columns that appear. In the Manage columns dialog box, move columns from **Available columns** to **Selected columns**, drag to rearrange the columns, and select **Apply**. You can select a field on a related table by dot-walking to it. For more information, see Dot-walking to data in related tables.
Apply conditions	Option to add a filter condition to the list. In the dialog box that appears, you can do the following: ▪ To apply a predefined filter, select **Use existing filter** and search for the filter. ▪ To view existing conditions, expand the **Filter overview** section. ▪ To create a filter condition, use the condition builder. For more information on building conditions, see Condition builder. ▪ To change the sort order of the filtered results, expand the **Sort by** section and make changes. ▪ To save the filter condition that you created for reuse, select **Save filter**. When you're done creating the filter condition, select **Apply filter**. For more information on working with filters for lists, see Filters.

c. Select **Apply filter**.

d. In the configuration panel, select **Save**.

9. Complete any additional actions on the list.

Option	Description
Edit a list category	**a.** In the navigation panel, select the list category. **b.** In the configuration panel, specify its **List category settings**. **c.** Select **Save**.
Change how the filtered list appears by selecting the list in the navigation panel	**a.** In the configuration panel, select **Manage columns**. **b.** In the form that appears, move columns from **Available columns** to **Selected columns**, drag to rearrange the columns, and select **OK**. **c.** Refresh the canvas preview by selecting the refresh icon ( ) in the canvas header.
Delete a list category	**a.** In the navigation panel, select the list category. **b.** In the configuration panel, select **Delete list category**. **c.** In the confirmation dialog, select **Delete**.
Delete a filtered list	**a.** In the navigation panel, select the filtered list. **b.** In the configuration panel, select **Delete filtered list**. **c.** In the confirmation dialog, select **Delete**.

Configure analytics for a workspace in Workspace Builder

Enable the Analytics Center for a workspace in App Engine Studio (AES). The Analytics Center enables users to track and analyze records and usage with dashboards, data visualizations, and insights on the instance.

Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task

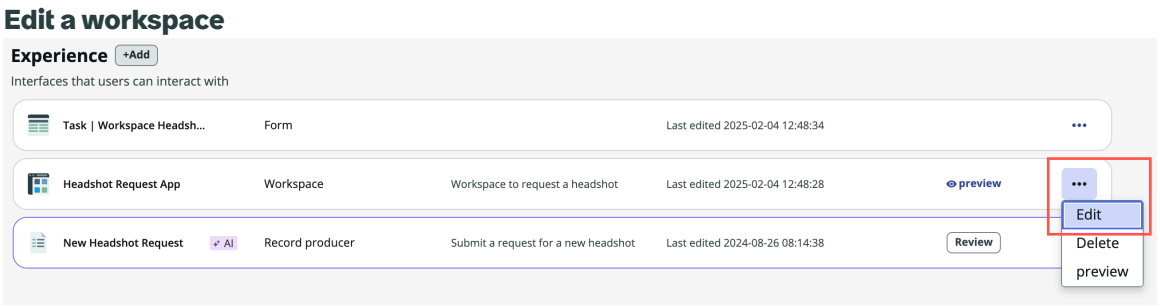
If Analytics Center page/route doesn't exist for a workspace, Workspace Builder is still available, but the Analytics Center tab doesn't appear in the workspace.

The Analytics Center always displays current, refreshed content.

Procedure

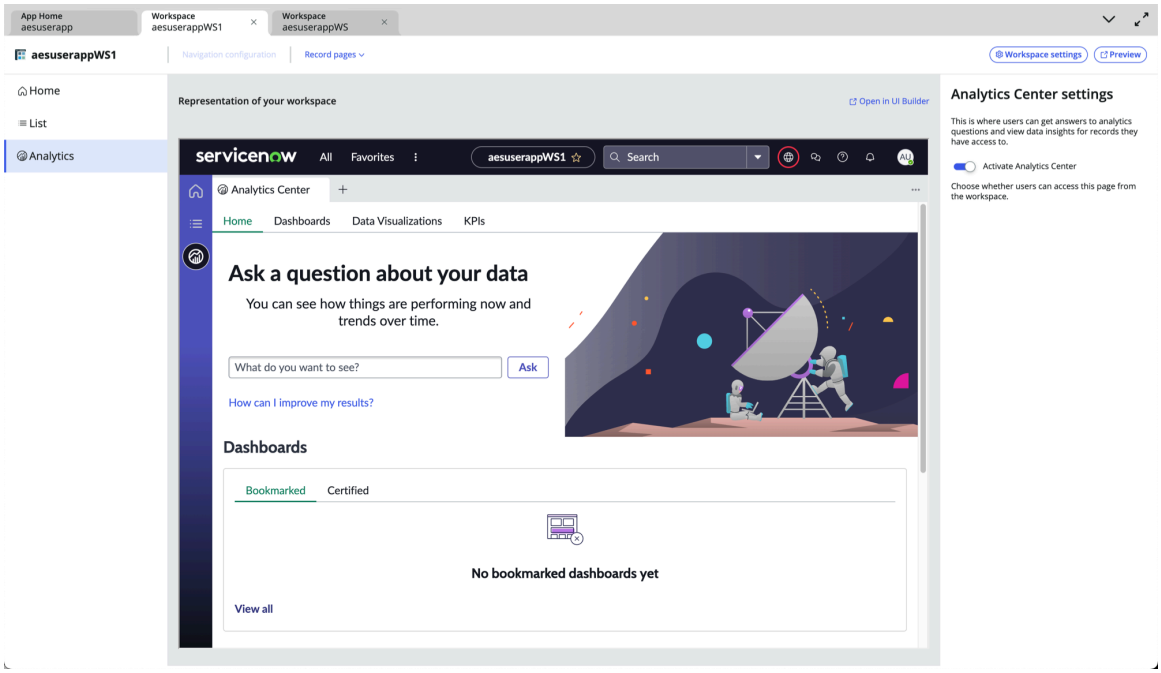
1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to the workspace you created, select the additional actions icon (•••) and then select **Edit**.

You can also select the bar that lists the workspace in the Experience section of your app.



The workspace appears in Workspace Builder.

4. In the navigation panel, select **Analytics**.
   You must be in the **Navigation configuration** tab. If you're already in the **Navigation configuration** tab, you can't select it.
   A mock-up of the Analytics Center appears in the preview panel.



5. In the configuration panel, select the **Activate Analytics Center** toggle option.
6. Make additional configuration updates to the Analytics Center by selecting the **Open in UI Builder** link in the canvas header.

Result

For more information on working with the Analytics Center, see Dashboards in Platform Analytics.

Add a portal

Create a portal in App Engine Studio (AES) to give your users a site where they can find information, create requests, and complete business tasks.

Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task

When you add a portal you're creating a basic version of it, which you must then continue editing in UI Builder before it's ready to deploy.

Note:
If you created an application using a template, a portal may already be added to your application.

Most companies implement a maximum of three portals, which they then populate with catalogs.

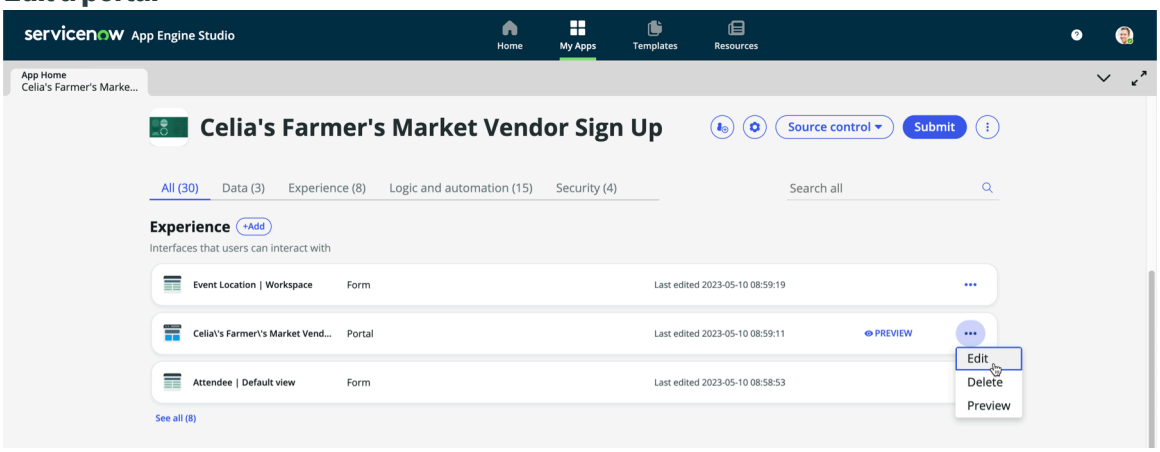
Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, next to Experience, select **Add**.
4. Select **Portal**, and then select **Begin**.
5. On the form, fill in the fields.

Field	Description
Name	Name of the portal. By default, the portal shares the same name as your application. **Tip:** Use a name that uniquely identifies the portal from other experiences. For example, `Office art portal`.
Description	Description of the portal.
URL	Web address of the portal. By default, the URL is based on the application name.
Roles	User roles to limit who can access the portal. To use a custom role for your portal, you must create one in **Security** first. For more information, see Add application security.

6. Select **Continue**, and then select **Done**.
   After you create the portal in AES, you must edit it in UI Builder by adding or configuring components on each page.
7. Next to the portal you created, select the menu icon (•••) and then select **Edit**.

Edit a portal



8. In UI Builder, edit the portal by adding or configuring components on each page. You can also add more pages to the portal, if needed.

> **Note:**
> The default portal pages are read-only and can't be edited. To change the content of each page, create a page variant and edit the variant.

What to do next

For more information on using UI Builder to edit portals, see UI Builder.

Sample portals you can build

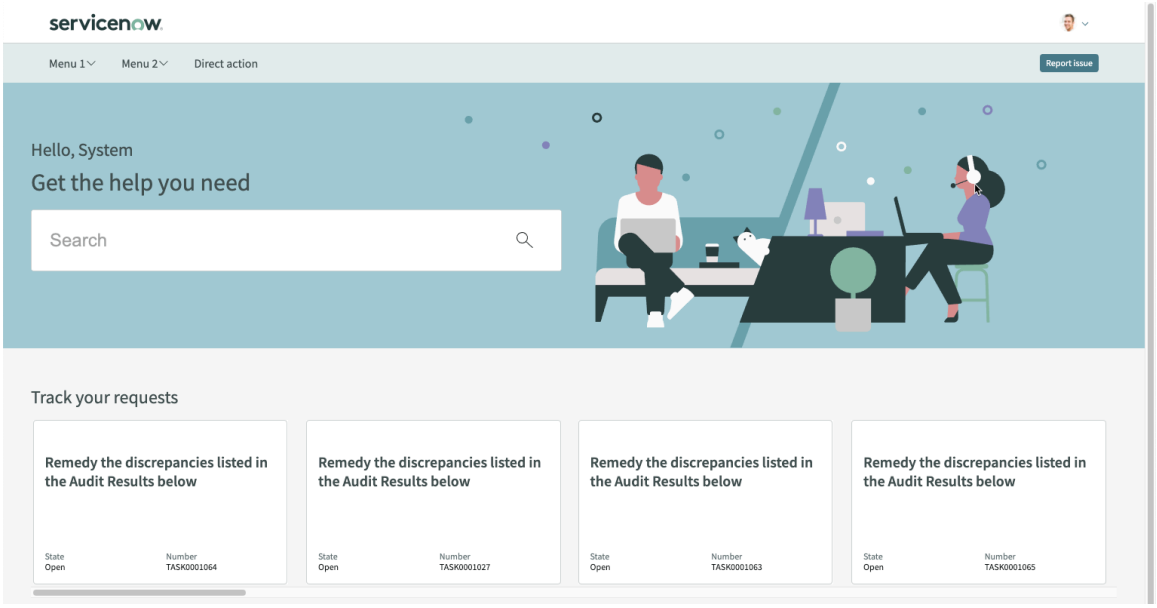
A portal is a site where users inside of your organization can find information, submit requests, and complete business tasks.

The portal that you create in App Engine Studio includes several default pages. You can use these pages as-is or edit them to suit your business needs.

- Article page
- Catalog item page
- Landing page
- Login
- Order success
- Record page(request)
- Record page(task)
- Search page
- Settings page
- Task approved
- Task rejected

> **Note:**
> Unlike Service Portal, the portal that you can create in App Engine Studio is built on the Next Experience UI Framework. You edit the portal by configuring pages and components in UI Builder.

Home page



Add a mobile experience

Add a mobile experience, or interface, to enable users to access your application from a ServiceNow native mobile app.

Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task

When you add a mobile experience, you're creating a basic version of the mobile app, which you must then continue editing in Mobile App Builder before it's ready to deploy.

Use Mobile App Builder to build and manage screens and records that make up workflows within the mobile apps that you build with AES. For example, you can add a webpage or a list that appears as a tab on a phone app.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, next to Experience, select **Add**.
4. Select **Mobile experience**, and then select **Get started**.
5. Select the kind of mobile experience that you want to add and select **Continue.**
   - Mobile Agent
   - Now Mobile

   For more information about the mobile experience options and important considerations when choosing a mobile experience, see Design considerations for mobile apps.
6. On the form, fill in the fields.

Add experience form

Field	Description
Add an icon	Icon that best reflects the content of the screen.
Label for experience	Description for the screen.
Roles	Roles that have access to the screen. You can specify multiple roles, but you must select them one at a time from the list that appears when you start typing in the **Roles** field. **Note:** If you do not specify a role, all users will have access to the mobile experience.
Available offline	Option to make the experience available offline. **Note:** Offline experiences are available only in Mobile Agent apps.

7. Select **Continue**.
8. Select the type of screen to add and select **Continue**.

Type of screen options

Option	Description
List	Screen that lists information retrieved from an existing data table, which you choose in the next step. **Note:** Creating a List screen adds all of the data table's values to the mobile experience.
Webpage	Screen that displays an existing webpage, which you specify the URL for in the next step.

9. To add a list page to the mobile experience, do the following:

   a. Select the table that will appear as a list on the screen.
   After you select a data table, some of its contents appear on the page.

   b. To make the table conditional, select **Set conditions for this table** and define the conditions that must be met for the data to appear using the condition builder.

   For more information, see Condition builder.

c. Select **Continue**.

d. Specify attributes for the data extracted from the table and select **Continue**.

Identifiers for table data

Field	Description
Add an icon	Icon that reflects the data in the table.
Screen label	Brief label to describe the list screen.

10. To add a webpage to the mobile experience, do the following:

a. Specify the attributes for the table data and select **Continue**.

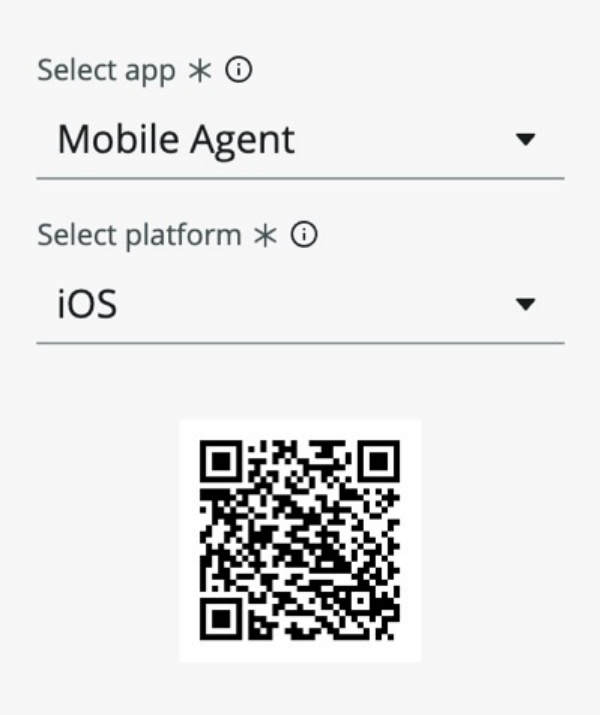
Identifiers for table data

Field	Description
Add an icon	Icon that reflects webpage.
Screen label	Brief label to describe the webpage.
URL	Absolute or relative URL for the webpage.

11. Depending on if you are done or not, do one of the following:
    - If you are done adding screens to the mobile experience, select **Done adding screens**.
    - To add more screens to the mobile experience, select **Add more screens**.
12. To finish creating the mobile experience, select **Done**.
    After you create the mobile experience in AES, you must edit it in Mobile App Builder.
13. Next to the mobile experience you created, select the menu icon (•••) and then select **Edit**.
14. Edit the mobile experience in Mobile App Builder.
    For more information, see Mobile App Builder.

What to do next

Scan or copy the QR code for your mobile experience using the **Select app** and **Select platform** fields.



For more information on modifying mobile experiences, see Mobile App Builder.

Sample mobile experience

A mobile experience enables users to access your application from a ServiceNow native mobile app.

Example mobile experience

The following example illustrates a sample mobile experience that you can build using the Mobile App Builder integration in App Engine Studio.

For more examples of mobile experiences that you can build, see Native Mobile Examples.

Editing an experience in App Engine Studio

Tailor your app's experience to suit your business needs by building interfaces in App Engine Studio (AES), such as for mobile apps or catalog items.

Editing workspaces in Workspace Builder

Workspace Builder is a streamlined no-code tool that enables you to configure custom workspaces built in AES quickly and easily. For more information on using Workspace Builder, see Building workspaces in AES. For more complex customizations, edit the workspace in UI Builder.

Editing forms created in Table Builder

When you add data to your app, any associated form views display in the **Experience** section of your app. To edit the form, select it in the **Experience** section. Form views can be edited in the **Forms** tab of Table Builder. For more information, see Forms in Table Builder.

Preview an experience

See what an application experience built in App Engine Studio (AES) looks like to your users by previewing it in a browser.

Before you begin

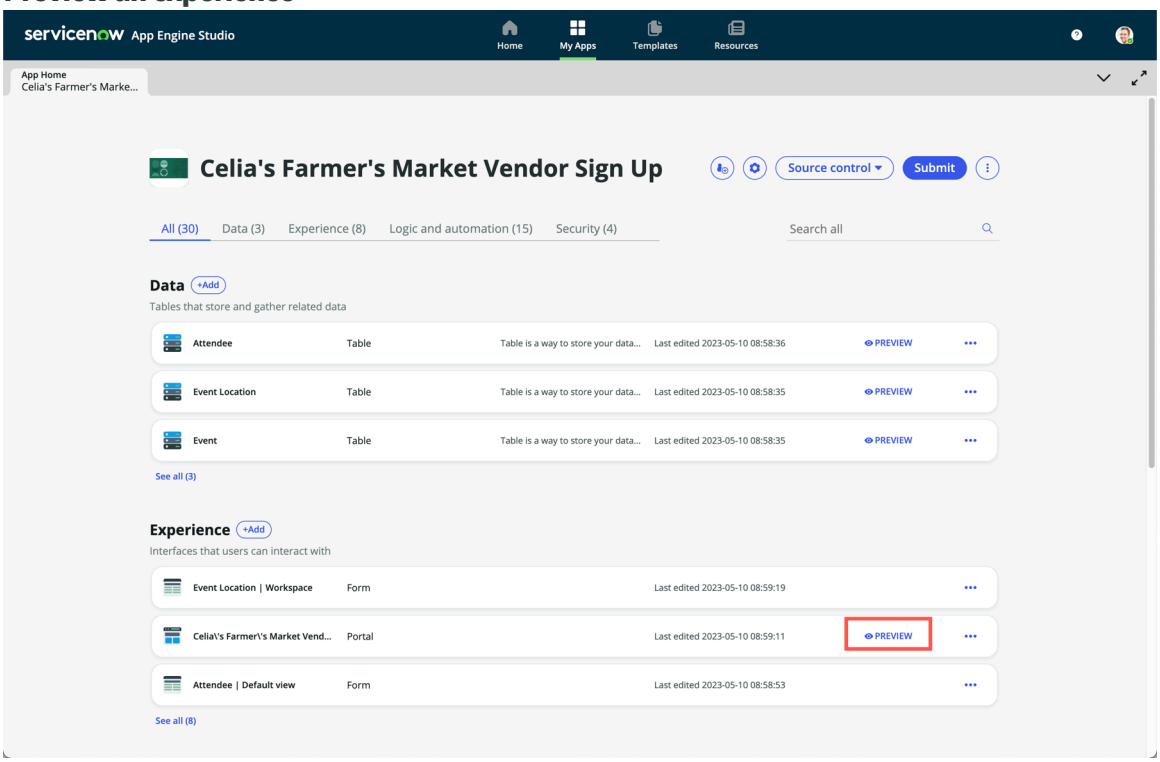
Add an experience to your application. For more information, see Add an application experience.

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to an experience, select **Preview**.

Preview an experience



4. On the browser tab that opens, review the experience.

What to do next

If the experience looks or behaves differently than expected, edit the experience.

Delete an experience

Delete an experience, such as a portal or record producer form, that you no longer need within App Engine Studio (AES).

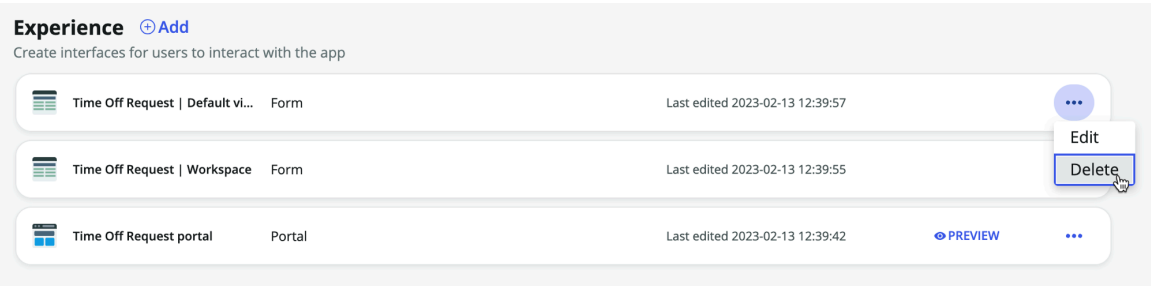
Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to an experience, select the menu icon (•••) and then select **Delete**.

Delete an experience



4. On the dialog box that appears, enter `delete` and then select **Delete**.

Add logic and automation

Replace manual work by adding logic and automation to your app in App Engine Studio (AES).

Application templates automatically add logic and automation to your application.

Types of automation

Automation	Definition	Example	Reference
Decision	Decisions decouple logic from code by creating decision rules, such as inputs that evaluate data from existing records, external data sources, or data provided at runtime.	Automatically apply cascading discounts to different products based on the increasing number of units ordered.	• Add a decision • Modify decision table structure in Workflow Studio
Email notification	Notifications are emails that are automatically sent when the specified conditions are met.	Automatically send an email when a new support record is created.	• Add an email notification • Modify an email notification
Flow	A flow is a sequence of reusable actions, initiated by a trigger event, and passing variables between actions.	Automatically generate a Twitter post when an incident is closed or send an email when a record in a specific table is	• Add a flow from scratch • Modify a flow

Types of automation (continued)

Automation	Definition	Example	Reference
		updated or created (record-based flow).	• Work with record-based flows in Table Builder **Note:** This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.
Process	Processes are cross-enterprise workflows that enable you to create a single, unified process. Processes can include multiple flows.	Standardize and automate how agents handle chat interactions with VIP users.	• Create a playbook • Edit a process

Note:
If you're using source control to collaborate with other developers, only changes that have been checked in are available to other developers. For example, if an admin creates a new flow for an app that's linked to Git, the new flow won't be available in the app for other AES users until the admin checks the flow into Git.

Add a pre-built flow using a template

Use a pre-built flow to quickly automate manual work in the app you're building in App Engine Studio (AES).

Before you begin

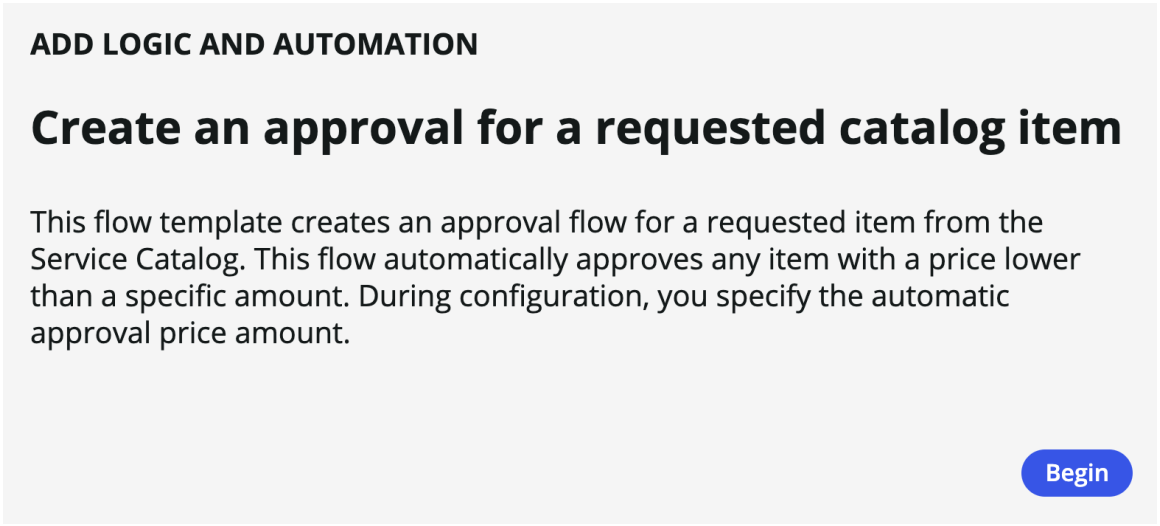
Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, in the Logic and Automation section, select **Add**.

4. Select the card for **Flow**.
5. From the gallery of automation templates, select a template and then select **Begin**.

Template automation



6. Complete the wizard for the template that you selected.
7. On the summary screen, select **Done**.

What to do next
You can edit the flow to tailor it to your business needs. For more information on working with automation, see Modify a flow.

Add a flow from scratch

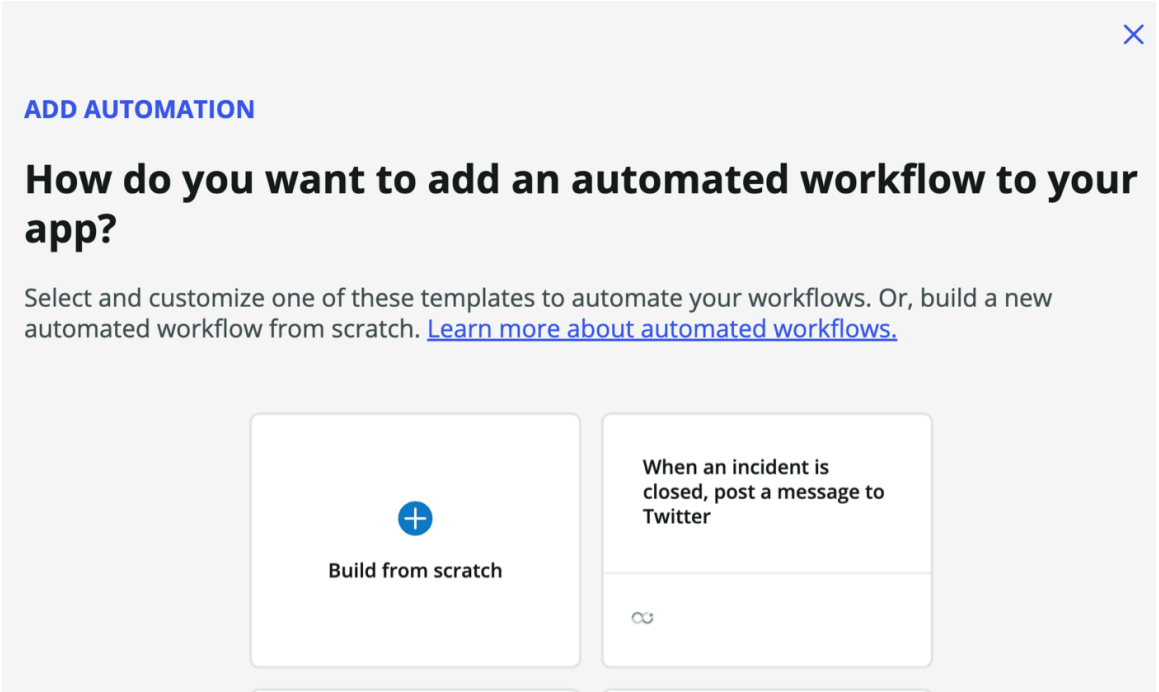
If an existing automation template doesn't fit your application goal, define custom automation in App Engine Studio (AES) by building a new flow.

Before you begin
Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

About this task
If a pre-built flow already exists, you can easily add its template to your app and then customize it. For more information, see Add a pre-built flow using a template.

Procedure
1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, in the Logic and Automation section, select **Add**.
4. Select the card for **Flow**.
5. Select **Build from scratch**.

Build from scratch

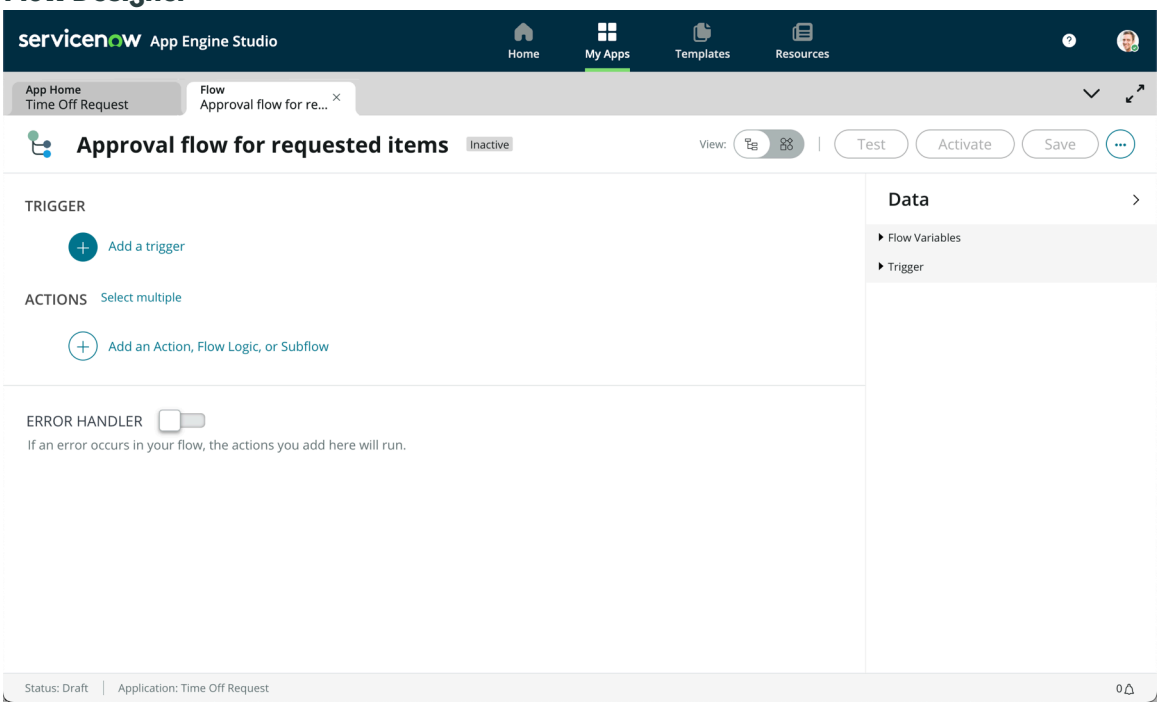


6. On the form, fill in the fields and then select **Continue**.

Field	Description
Name	Name to identify your flow.
Description	Description of your flow.
Protection	Access settings for the flow. You can select one of the following options: **-- None --** Option to enable other users to edit the flow. **Read-only** Option to enable other users to view the flow but not edit it.
Run as	Selection to specify if the flow runs as a system user or the user who initiates the session. If updates should come from the user who triggered the flow, select **User who initiates session**. For example, when you want incident record comments to come from the current user, or approval emails to originate from the approver.
Run with roles	Roles that the flow runs with. This option is available only when **Run as** is set to **User who initiates session**. To use a custom role for your flow, you must create one in **Security** first. For more information, see Add application security.

7. On the summary screen, select **Edit this flow** to open your flow.

Flow Designer



For an example of how to build a flow, see Build your first flow in Flow Designer.

8. Add a trigger to your flow.

   a. Under the TRIGGER section, select **Add a trigger**.

   b. From the Trigger list, select a trigger that will start running your flow.
   For more information on trigger types, see Workflow Studio flow trigger types.
   The system displays a set of fields depending on the type of trigger that you've selected.

   c. Set up your trigger by filling in the fields.
   For a record-based trigger, for example, select a table and set field conditions that, when met, will start running your flow.

   d. Click **Done**.

9. To add actions, flows, subflows, or flow logic, select **Add an Action, Flow Logic, or Subflow**.

   a. Select an option.

Option	Description
Action	Select the desired action. Workflow Studio includes Workflow Studio actions that are available to flows and subflows. Alternatively, a user with the action_designer role can create additional actions to add to flows. The Integration Hub and Spokes plugins install additional actions. To add draft actions from the More Actions menu, set **Show draft actions** to **true**.

Option	Description
	To view spokes that are available in the ServiceNow Store, set **Show store spokes** to **true** from the More Actions menu. **Note:** Under **Not Installed Spokes**, the system displays spokes that are available in the ServiceNow Store based on compatibility with the ServiceNow version and application dependency on Workflow Studio.
Flow Logic	Select an option to specify conditional or repeated operations.
Subflow	Select a published subflow and define the input values. In addition to adding a subflow as a flow action, you can enable the **Show triggered flows** option from the More Actions menu to select an activated flow and define the required inputs. Running a triggered flow ignores its trigger conditions and runs all actions.

To change the order of an action in a flow, drag the handle on the left side of the action to the desired location.

The system displays a set of fields depending on the option that you selected.

b. To configure the action, flow logic, or subflow, fill in the fields.

c. Select **Done**.

d. Repeat adding actions until complete.

10. To specify how the app will handle errors, enable the **ERROR HANDLER** switch and fill in the fields.
    For more information on configuring how errors are handled in flows, see Flow error handler.
11. Select **Save**.
    The ServiceNow AI Platform saves a draft of the flow, trigger, and actions.

What to do next

Test your flow until you're ready to activate it. For more information on testing and editing flows, see Activate a flow.

Note:
Your application can trigger only activated flows.

Modify a flow

To update or change an existing flow for an app you built in App Engine Studio (AES), edit it.

Before you begin

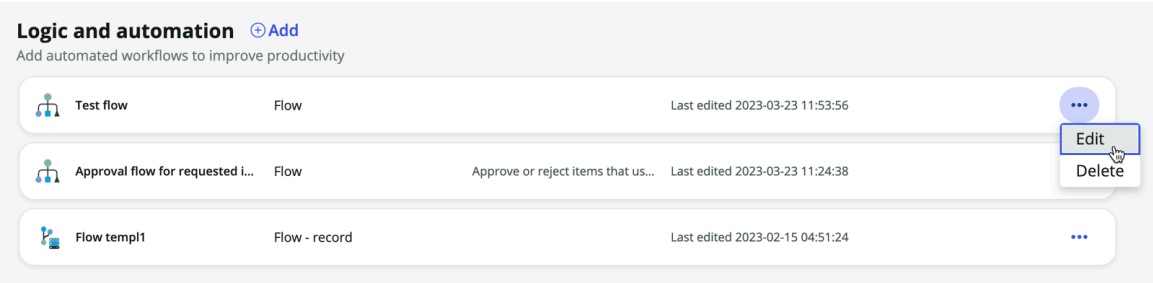
Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

Add logic and automation to your application. If you created an application using a template, logic and automation may already be added to your application. For more information on adding custom logic and automation, see Add logic and automation.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to a flow, select the menu icon (•••) and then select **Edit**.

Edit a flow



Note:
When a flow is triggered by a record being created or updated in a data table, it's referred to as a record-based flow. Record-based flows display in the list of logic and automation with a distinct icon ( ). These flows open up within the **Flows** tab in Table Builder if you are licensed for Table Builder for App Engine and have it installed. Contact your account team for details. For more information on flows functionality in Table Builder, see Flows in Table Builder.

4. Take the appropriate actions to edit the flow.

Option	Description
Change the flow name, description, or roles	In the main header, select **Properties**, enter the values you want into the appropriate fields, and then select **Update**.
To edit the trigger	In your flow, select the trigger description, fill in the fields as desired, and then select **Done**. **Note:** Modifying triggers can result in the deletion of referenced action configurations.
To edit an existing action	In your flow, select the action description, fill in the fields as desired, and then select **Done**.
To add a new action	In your flow, select the plus icon in the ACTION section, then proceed as you would for adding an action to a new flow.

For more information on editing flows, see Edit a flow.

5. To save your changes, select **Save**.

What to do next

Test a flow to make certain it works the way you expect. For more information on testing flows, see Test a flow.

Add a decision

Create automation for decisions in App Engine Studio (AES). The automation decouples decision logic from code by enabling you to create and manage decision rules.

Before you begin

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, in the Logic and Automation section, select **Add**.
4. On the Add logic and automation screen, select **Decision** and then **Get started**.
5. Specify the name and scope of the decision table.

Decision Table form

Field	Description
Name	Name for the decision table.
Accessible From	Scopes that can use the decision table. Available values are as follows: ○ **All application scopes** ○ **This application scope only**

6. Select **Done**.

What to do next

After you create the decision table, you must edit it to specify inputs and triggers. For more information on editing decision tables, see Modify decision table structure in Workflow Studio.

Learn more about decisions in AES	Additional ServiceNow resources
ServiceNow provides several additional resources on adding decisions in App Engine Studio.	 Getting started with Decision Builder YouTube video
	 Build my first App Engine Studio application

Learn more about decisions in AES	Additional ServiceNow resources
	ServiceNow University Introduction to App Engine Studio on-demand course

Edit a decision

To update or change an existing decision for an app built in App Engine Studio (AES), edit it in Workflow Studio.

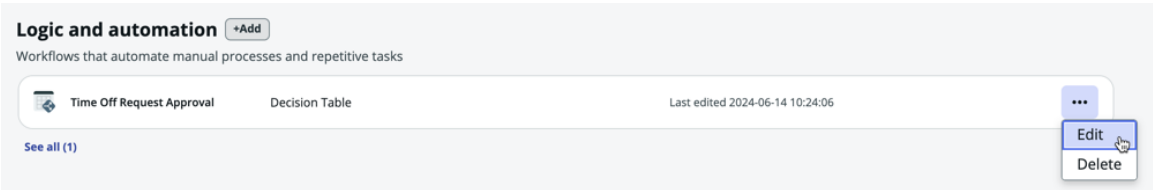
Before you begin

Add logic and automation to your application. If you created an application using a template, logic and automation may already be added to your application. For more information on adding custom logic and automation, see Add logic and automation.

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to a decision, select the menu icon (•••) and then select **Edit**.



4. Take the appropriate actions to edit the decision.
   For more information on editing decision tables, see Modify decision table structure in Workflow Studio.
5. To save your changes, select **Save**.

Add an email notification

Build notifications in App Engine Studio (AES) to create automated emails for events using templates or custom messages.

Before you begin

You can perform this task if you're either an administrator or a developer for the application. The required developer permission is **All File Types**. For more information on being a developer for an application, see Delegated development in App Engine Studio.

Role required: admin or delegated_developer

About this task

Email notifications enable you to send emails to the selected users about activities in the system, such as updates to incidents or change requests.

Note:
You can't currently create any other types of automatic notifications, such as SMS messages.

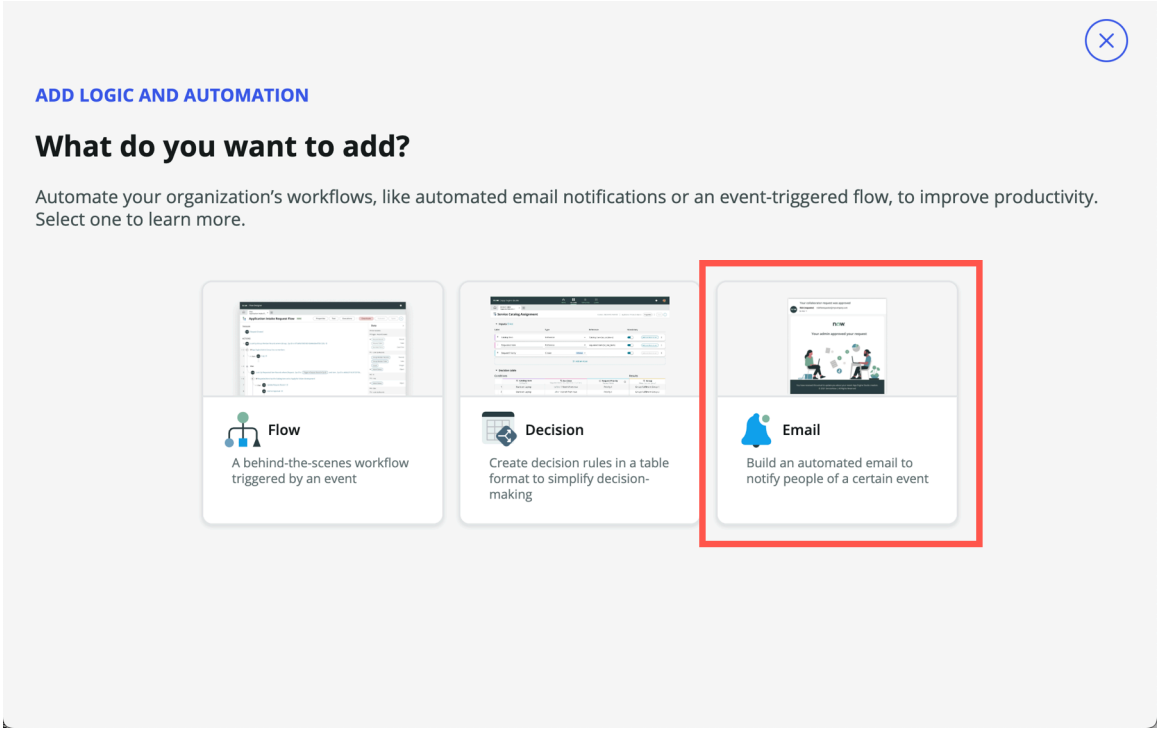
Creating an email notification involves specifying when to send it, who receives it, and what it contains.

If your administrator has created an email template in the ServiceNow AI Platform, you can use it as is or override its content. For more information on email templates, see Email Templates. Also check with your administrator to see if they are using email layouts to control the header, body, and footer. For more information on email layouts, see Email layouts.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, in the Logic and Automation section, select **Add**.
4. Select the card for **Email**.

Add email notification



5. On the form, fill in the fields for the basic settings.

Add notification form basic settings

Field	Description
Name	Name of the email notification. Descriptive names help identify the purpose of the email notification. For example, `Incident Opened & Unassigned`.
Short description	A brief explanation of the purpose of the notification.

6. On the Notification trigger settings section of the form, fill in the fields.

Add notification form Notification trigger settings

Field	Description
Send email notification	Situation where a notification is sent, such as after something happens or when a table is modified. Choices are as follows: ○ **When a record is created or updated** ○ **Only when a record is created** ○ **Only when a record is updated** ○ **When triggered via Flow Action** ○ **When event is fired** This field is automatically set to **When a record is created or updated**. After you create an email notification with a Flow Action as the trigger, you specify the notification when creating a flow in Workflow Studio. For more information, see Notification step.
Table	The database table that hosts the records to link the notification trigger to. For example, the Incident [incident] table. If you set the **Send email notification** field to one of the following values, the table is required: ○ **When a record is created or updated** ○ **Only when a record is created** ○ **Only when a record is updated** You can select from two sections: Tables that are already available in your app, and all tables outside the app scope.
Event name	Event that triggers the notification. This field appears only when **When event is fired** is selected from the **Send email notification** field.

If the same trigger generates multiple notifications, the system only sends one notification. The system considers all other notifications, even if they have a different subject and body, as duplicates. The Ignore Duplicates business rule controls this functionality.

7. Make the trigger conditional by using the condition builder.

Define the condition sets and values that must be met for the notification trigger. For more information, see Condition builder.

Note:
If you specify that the notification trigger is a Flow Action, the condition builder isn't available.

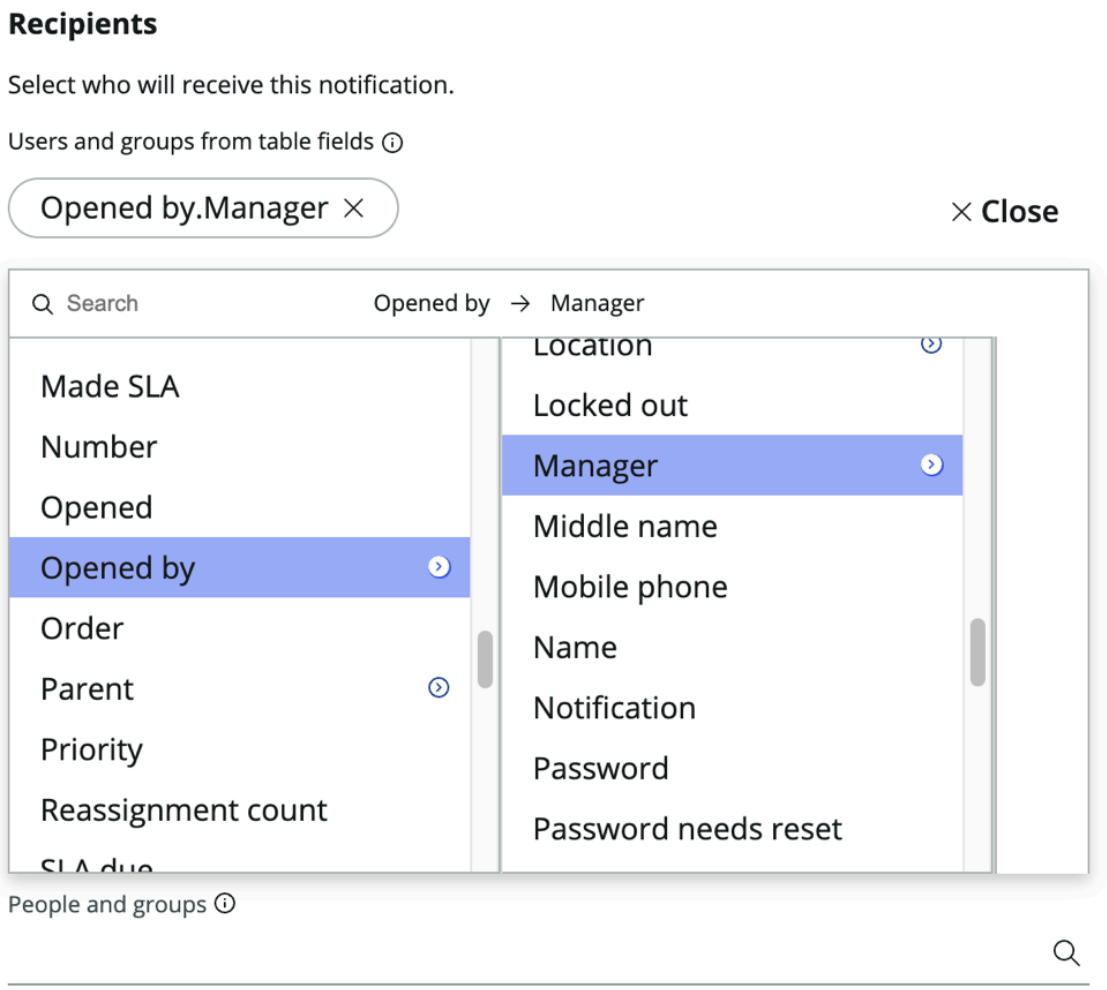
8. In the Recipients section of the form, specify who receives the notification.

   You can:

   - Search for users or groups.
   - Define dynamic recipients by selecting **Add**, which requires you to specify a table in the notification trigger details.

Add notification form Recipients settings

Field	Description
Users and groups from table fields	Dynamic users and groups who receive the email based on the table. For example, if a notification uses the Incident [incident] table, you can select the **Opened by** field to send the notification to users or groups who opened the incident. For some fields, you can also select values contained by a reference field, such as **Opened by** > **Manager**. Select the **Add** button to search for dynamic users and groups based on fields from the specified table. **Note:** If you don't specify a table, the **Add** button isn't available.
People and groups	Static addresses for users and groups who receive the email notification. You can also search for users and groups.



9. In the Email content section of the form, select a template or write original content.

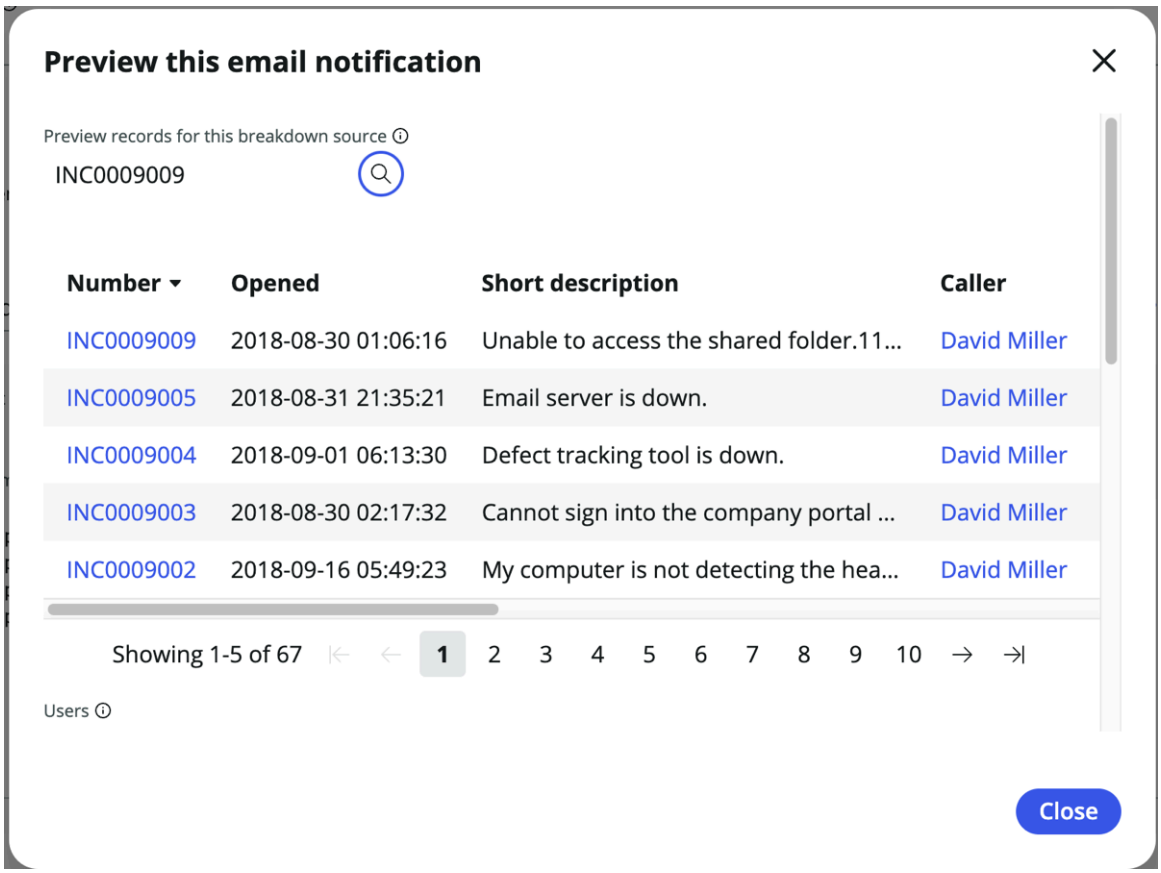
Add notification form Email content settings

Field	Description
Email template	Predefined email template that you can use to apply existing content, which you can override. You can select only templates that have the following: ○ The same table as the notification ○ No specified table If you use a template and don't override it, then when your administrator updates the template, the content of your email notification is also automatically updated.
Override template	Option to override the contents of the selected email template. If you use an email template, any images from the template still belong to the template regardless of whether you override it. Images

Field	Description
	in the notification are automatically updated when administrators update the template. If you override an email template and then change the selected template, the system removes the overrides and replaces them with content from the new template.
Subject	Subject line for the email. To make the subject dynamic, use the **Email variables** field next to the **Email message** to generate variable syntax in the message body. You cam then copy and paste the variable syntax in the subject line. If empty, the system uses the **Subject** value from the email template. If you enter a value in this field, it overrides the template.
Email message	Content of the email notification message, which can include variables. You can format the contents of the email message using the rich text editor icons. If empty, the system uses the **Message** value from the email template. If you enter a value in this field, it overrides the template value.
Add variables	Dynamic variables, or values, derived from fields in the specified table. **Note:** You must specify a table for the **Add variables** button to appear. Select variables in the **Email variables** field and use the arrow to move them into the **Email message**. Use variables to include values from a record in the table, such as an incident short description or comments and work notes. Variables are available fields from the specified table.

10. Select **Create**
    The notification is saved, and you can preview it.
11. View how the email notification appears by selecting **Preview**.

Check the preview that appears. To see how a notification renders, select the search icon in the preview and select the record you want to preview.



Result

When you create a notification in AES, the notification is saved only to the application that you're currently building.

Modify an email notification

Edit an email notification created for an app in App Engine Studio (AES) to change it. For example, modify when to send the email notification, who receives it, and what it contains.

Before you begin

You can perform this task if you're either an administrator or a developer for the application. The required developer permission is **All File Types**. For more information on being a developer for an application, see Delegated development in App Engine Studio.

Role required: admin or delegated_developer

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to a notification, select the menu icon (•••) and then select **Edit**.
4. Edit the notification by taking the appropriate actions.
   For more information on working with notification fields, see Add an email notification.
5. Change the template you're using.

a. Select a different template from the **Email template** field.

   If you override an email template and then change the selected template, the system removes the overrides and replaces them with content from the new template.

b. In the dialog box that appears, select **Change template** to confirm that you want to lose the changes you made by overriding the previous template.

6. To save your changes, select **Save**.

Edit a process

To update or change an existing process for an app built in App Engine Studio (AES), edit it in Playbooks.

Before you begin

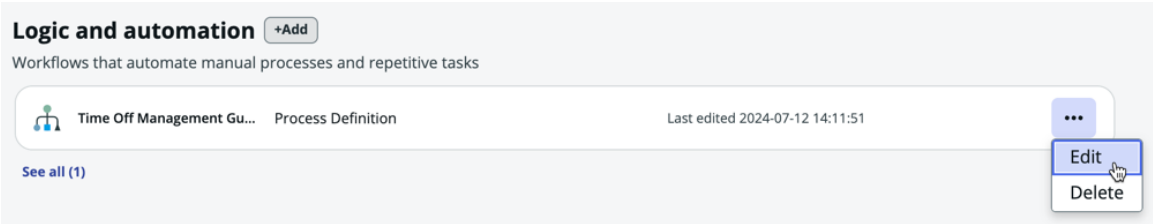
Before you edit a process, you must add a process in Playbooks. For more information, see Create a process definition.

You can edit processes in App Engine Studio, but you must activate Playbooks for App Engine Studio and enable the **Process Automation Designer for App Engine [com.glide.pad.license]** plugin to get started. For more information, see Activate playbooks.

Role required: sn_app_eng_studio.user or delegated_developer. For more information, see Delegate developers using AES.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to a process, select the menu icon (•••) and then select **Edit**.'



4. Take the appropriate actions to edit the process.
5. To save your changes, select **Save**.

What to do next

Test a process to make certain it works the way you expect. For more information on testing processes, see Test a process.

Add application security

Control who is permitted to use or edit your application by configuring roles in App Engine Studio (AES).

Adding security in App Engine Studio, UI Builder, and Catalog Builder

Working with roles

You add security to your application by defining roles. A role is used to limit and control access to your application. The access that you define for a role is granted to all users who are assigned to the role.

Note:
If you don't have permission to work with roles for an app, the **Add** button for the Security section appears, but you can't select it.

An administrator assigns a role to the appropriate user based on the kind of work that the role permits. For example, to enable a manager to read and approve employee requests, an administrator would assign a role that has access to read and update records from the relevant table. The role may also need access to the portal or workspace where the manager would see the request.

Application templates automatically add security to your application. If you use a template to create your application, you can edit the roles that were added or add different roles.

Learn more about AES roles and security	Additional ServiceNow resources
App Engine Studio includes default roles that you can use for your application. Alternatively, you can build custom roles for your application. After you submit your application, an administrator on-boards the requesting team by assigning the roles that you've defined.	What is Security? - ServiceNow Developers site Security and Roles - ServiceNow Developers site Elevating Privileges - ServiceNow Developers site

Build a new role for your application

Create a custom role for your application in App Engine Studio (AES).

Before you begin

Role required: admin, security admin, or delegated_developer (with the **Security Management** permission)

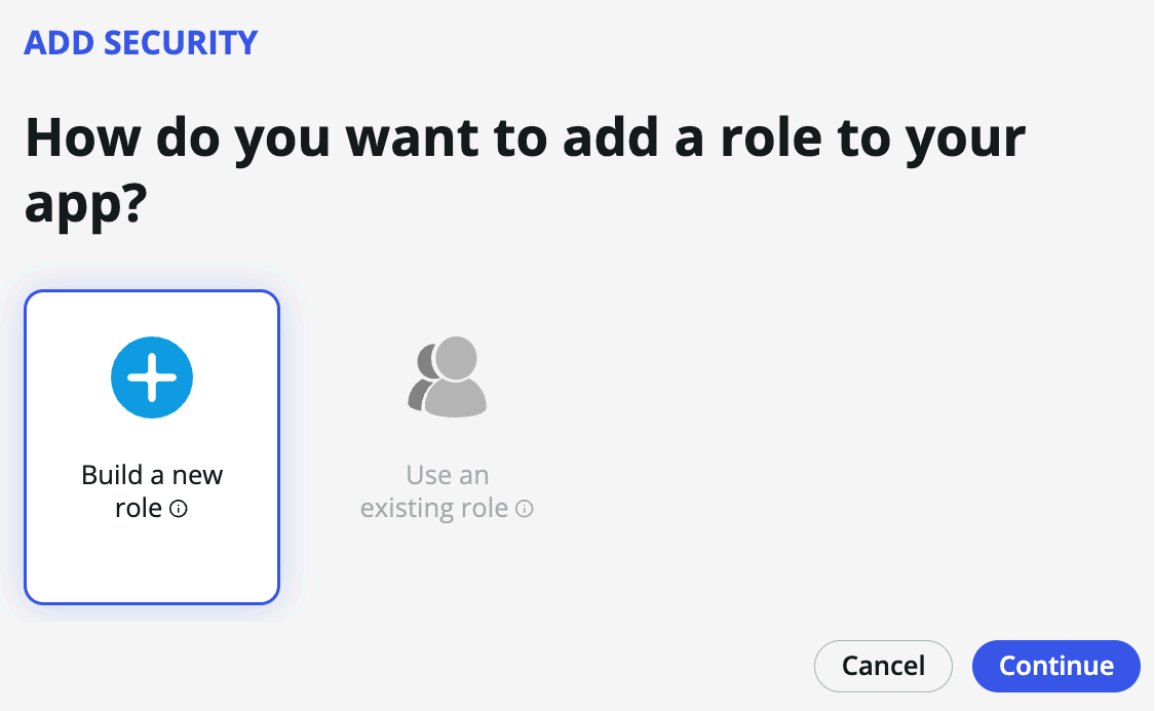
Note:
If you are an admin, be sure to elevate your role.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.

3. In your application, next to Security, select the add icon ( ).
4. Select **Build a new role** and then select **Continue**.

Option to build a new role



5. Enter a name and description for your role, and then select **Continue**.

> **Tip:**
> For consistency with existing role names, enter a role name that is all lower-case and uses underscores instead of spaces. For example, for an office art requester role, you would enter `office_art_requester`.

6. For each table in your application, select one or more access controls for your role.

Create

Users with the role can create new records on the data table.

Read

Users with the role can read the data that is stored in the table.

Write

Users with the role can update existing records in the table.

Delete

Users with the role can delete table records.

7. Select the **Experience** tab, and then select the experiences that users with the role can access.
8. Select **Continue**, and then select **Done** on the summary screen.

Use an existing role for your application

Use a previously created role for your application in App Engine Studio (AES).

Before you begin

Your application must contain either data or experiences in order to add an existing role. For more information about adding data or experiences to your application, see the following guidance:

- Create a data model for your application
- Add an application experience

Role required: admin, sn_app_eng_studio.user, or delegated_developer

About this task

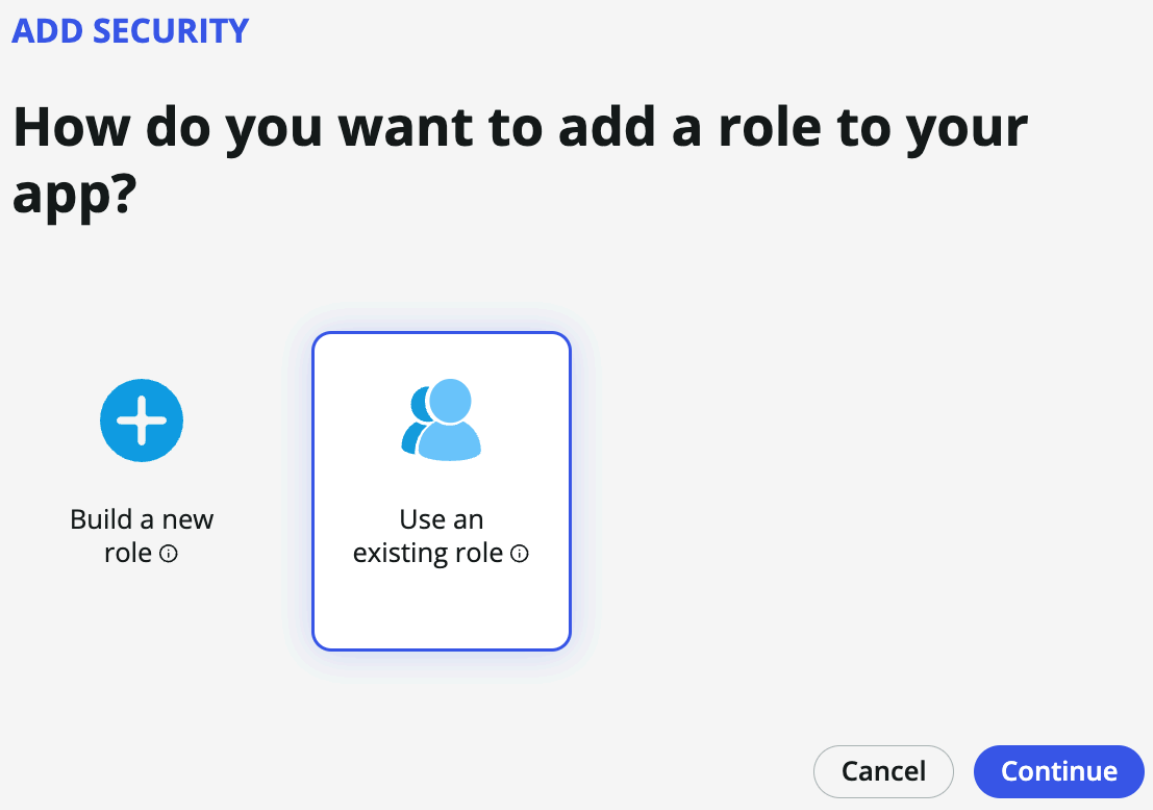
App Engine Studio includes default roles that you can use for your application. For more information on the default roles, see Base system roles.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, next to Security, select the add icon ( ).
4. Select **Use an existing role**, and then select **Continue**.

> **Important:**
> If your application doesn't contain data or experiences, the **Use an existing role option** will appear, but you cannot select it.

Option to use an existing role



5. Select the search field and then select a role from the list.
6. Select **Edit**.
   A new tab opens to set access controls for the existing role.

7. In the Experience section, select the experiences that users with the role can access.
8. In the Data section, set access controls for each available table.

   Create

   Users with the role can create new records on the data table.

   Read

   Users with the role can read the data that is stored in the table.

   Write

   Users with the role can update existing records in the table.

   Delete

   Users with the role can delete table records.

9. Select **Save**.

Change access settings for a role

Control the application permissions for a role in App Engine Studio (AES).

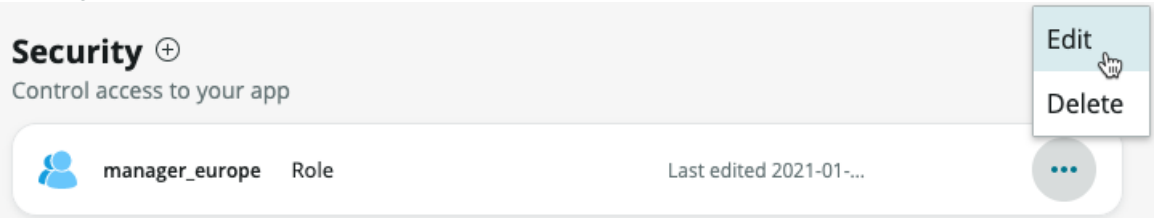
Before you begin

Role required: admin, sn_app_eng_studio.user, or delegated_developer

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to a role, select the menu icon (•••) and then select **Edit**.

Editing a role



A new tab opens to edit the role.

4. In the Experience section, select the experiences that users with the role can access.
5. In the Data section, set the access controls for each available table.

   Create

   Users with the role can create new records on the data table.

   Read

   Users with the role can read the data that is stored in the table.

   Write

   Users with the role can update existing records in the table.

   Delete

   Users with the role can delete table records.

6. Select **Save**.

Delete a role

Delete a role that you no longer need in App Engine Studio (AES).

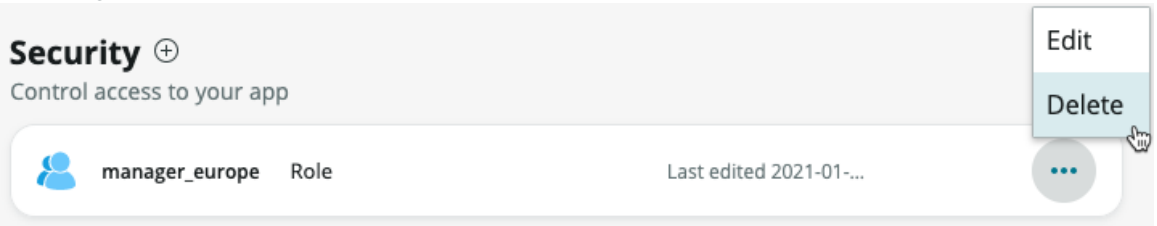
Before you begin

Role required: admin, sn_app_eng_studio.user, or delegated_developer

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Next to a role, select the menu icon (•••) and then select **Delete**.

Deleting a role



4. On the dialog box that appears, enter `delete` and then select **Delete**.

Collaborate with other developers

In App Engine Studio (AES), you can add or remove users and groups to be collaborators on an application.

Users have varying levels of permissions for actions on the Collaboration feature. At a high level:

Users with these permissions	Can perform these actions
Manage Collaborators delegated development permission to the application	Can: • See a list of collaborators and their descriptors • Search for users or groups • Select or change the collaboration descriptor for a user or group (but users can't change customized users or groups) • Add collaborators by sending invitations • Remove users or groups (but users can't remove customized users or groups) Can't: • Customize permissions for a user or group • Deploy via update sets
Invite Collaborators delegated development permission to the application	Can:

Users with these permissions	Can perform these actions
	• See a list of collaborators and their descriptors • Search for users or groups • Add collaborators by sending invitations Can't: • Select or change the collaboration descriptor for a user or group • Remove users or groups • Customize permissions for a user or group
No Collaborator-related roles to the application	Can see a read-only list of collaborators and their descriptors. Can't: • Search for users or groups • Select or change the collaboration descriptor for a user or group • Add collaborators by sending invitations • Remove users or groups • Customize permissions for a user or group
Administrators	Admins must elevate to a security_admin role to work with collaboration features. For more information, see Elevated privilege roles. Administrators can: • See a list of collaborators and their descriptors • Show/hide members of a group • Search for users or groups • Select or change the collaboration descriptor for a user or group (includes changing customized users or groups and changing the owner) • Add collaborators by sending invitations • Remove users or groups (includes removing customized users and groups and removing the owner even if there is only one owner) • Customize permissions for a user or group

Note:
You should create collaboration descriptors in addition to Owner and Editor in the global scope. If you want collaboration descriptors to appear and be used in AES, you should also set them to `standard = TRUE`. AES doesn't support collaboration descriptors that are created in custom scopes, and non-standard collaboration descriptors don't render in AES.

When you add a user or group, a collaboration task is generated and an approval flow kicks off. To find all collaboration tasks, navigate to **All** > **App Engine** > **Collaboration** > **Collaboration Tasks**. The collaboration task provides information on which application a developer is being added to, and what permissions are granted. Approvers sometimes need to review these task records before they add developers to the application.

If you're an admin, you can modify the Collaboration Request flow. The base system Collaboration Request flow handles collaboration requests as follows:

- If the user has AES or delegated developer permissions and isn't new to the platform, the collaboration request approval record is auto-approved.
- If the user does not have AES or delegated developer permissions and is new to the platform, approval is required.

If you're an admin, you can modify the collaboration descriptors that developers use to assign delegated development permissions. The base table provides Owner and Editor collaboration descriptors. By default, Owners have the manage collaborator delegated development permission set, and Editors have the invite collaborator delegated development permission set.

Note:
If you’re using source control to collaborate with other developers, only changes that have been checked in are available to other developers. For example, if an admin creates a new flow for an app that’s linked to Git, the new flow won’t be available in the app for other AES users until the admin checks the flow into Git.

Add a user or group to collaboration

Give users or groups permission to collaborate with other developers in App Engine Studio (AES) using the Collaboration feature.

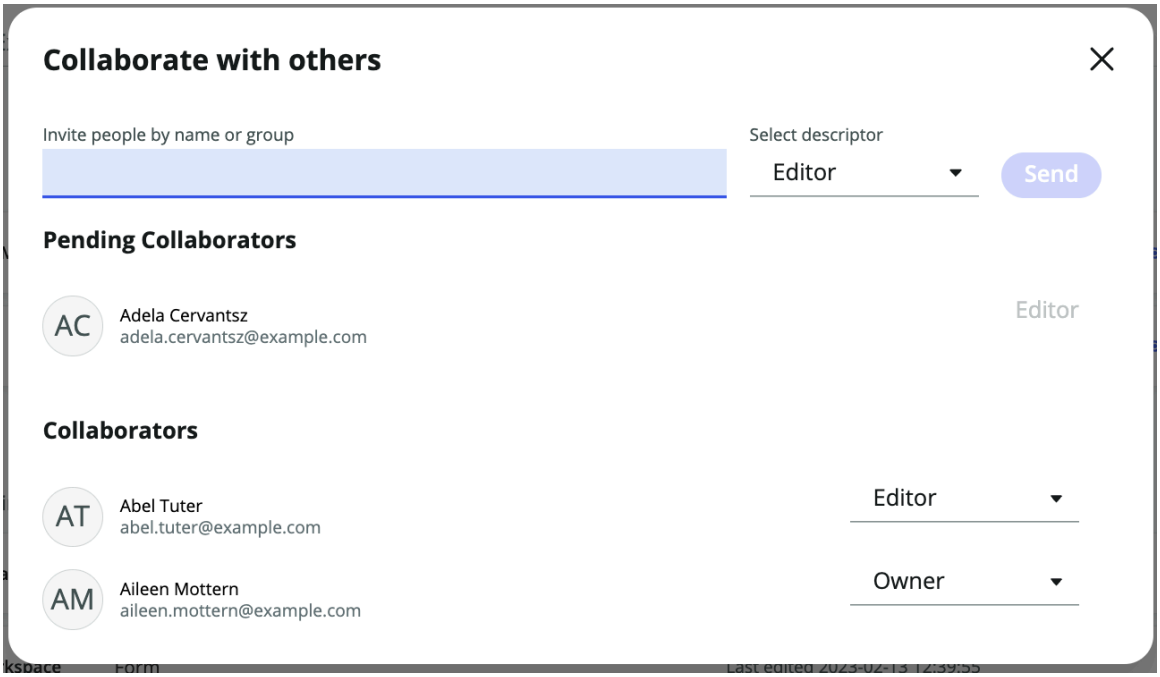
Before you begin

Role required: admin, manage collaborator delegated development role, or invite collaborator delegated development role

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select the Manage collaborators button.

Collaborate with others



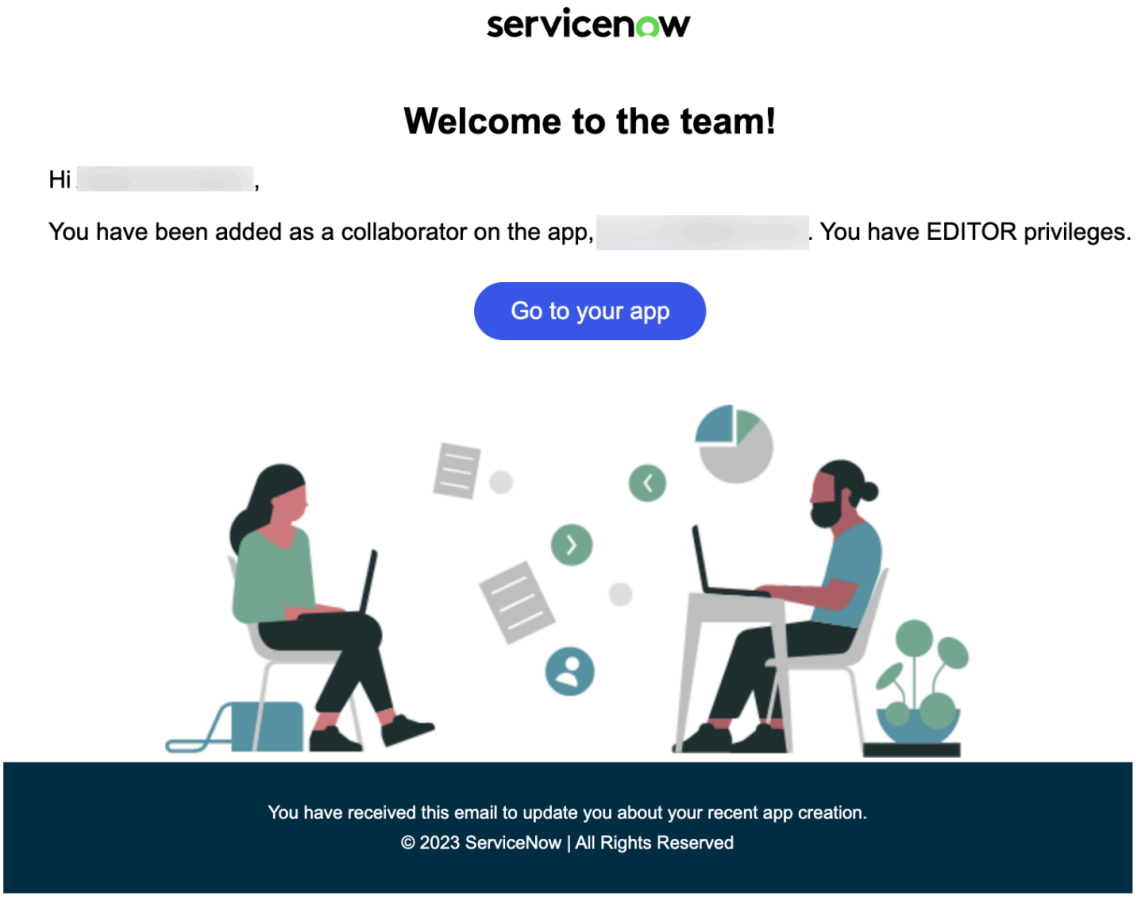
4. To add another user or group as a collaborator, enter the user name or group name in the **Invite people by name or group** field.
5. Search for a specific user or group by entering the first few characters of the name.
   A drop-down list with matching user names and groups appears where you can select the user or group you want to add. If a user or group appears in the drop-down list but you can't select it, it's already been added as a collaborator and can't be re-selected.
6. Select the collaboration descriptor for the user or group that you're adding.
   For a list of collaboration permissions, see Delegated development and collaboration permissions.

   Note:
   Users with Invite Collaborators permissions can't do this, and will default to the Editor option.

   Only App Collaboration Descriptors that are defined in the global scope and have the standard option selected appear in the list.
7. Select **Send**.

   - If the user has AES or delegated developer permissions and is new to the platform, the user is listed under the pending requests section and approval is required. After the request is approved, both the requester and the user

receive an email indicating that the user has been added to the application.



- If the user has AES or delegated developer permissions and is not new to the platform, the collaboration request is auto-approved. Both the requester and the user receive an email indicating that the user has been added to the application.

Note:
If a valid controller has been configured on the instance from which the collaboration request originated, a collaboration request task is created on the controller instance. If the originating instance isn't configured on the controller, work notes are automatically added to the request record indicating that the instance must be configured before approval is granted. If the originating instance is the controller or a valid controller has not been configured, the collaboration request task is created on the requesting instance. For more information, see Configure your controller instance.

Change collaborator permissions

Change the permissions that are assigned to a collaborator or create a custom collaboration permission to enable what people can do in App Engine Studio (AES).

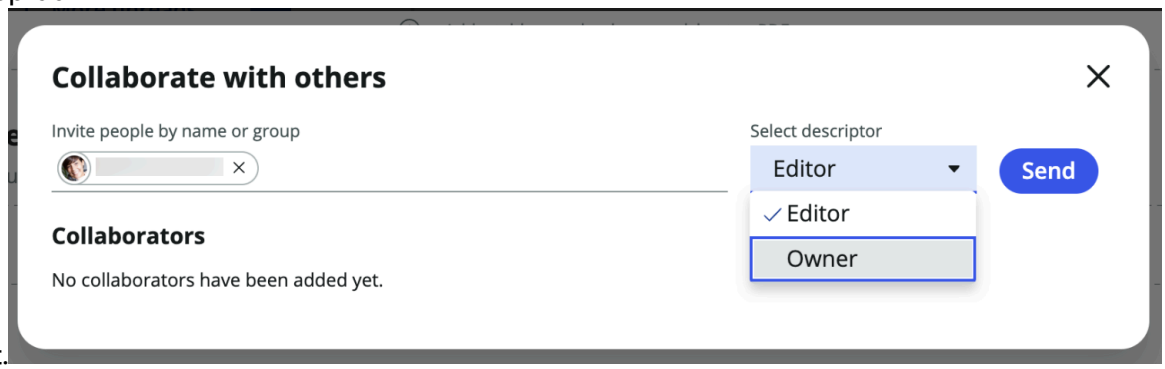
Before you begin

Role required: admin or manage collaborator delegated development role

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select the Manage collaborators button.

4. For the user or group you want to change, select a different collaborator descriptor from the drop-down list.



Customized users and groups are not editable by non-admins.

5. Admin only: Create custom collaboration permissions for specific users in an application.

   a. In AES, select **Customize Permissions** from the drop-down list across from the user's name.

   b. Select or clear delegated development permissions for the user or group.
   For a list of collaboration permissions, see Delegated development and collaboration permissions.

   If the selection matches that of another standard Collaboration descriptor, the system will provide a prompt before proceeding. For details on each delegated development permission, see Delegate development and deployment permissions to personnel.

6. Select **Save**.

Delegated development and collaboration permissions

Collaboration permissions define what users can do in App Engine Studio (AES).

File type access permissions

File type access permissions grant access to application file types.

File type access permissions

Permission	Description
All file types	Grants access to collaborate on all file types. **Note:** This option includes access to additional file types not listed in separate permissions.
Integrations	Grants access to web service APIs, REST APIs, data sources, and Integration Hub - Import.
Reporting	Grants access to reports and scheduled reports.
Mobile builders	Grants access to build mobile experiences, such as with Mobile App Builder.

File type access permissions (continued)

Permission	Description
UI Builder	Grants access to work with UI Builder to build more complex interfaces.
Workflow	Grants access to the Workflow Editor and Activity Creator.
Service Portal	Grants access to work with Service Portal editors and tools.
Workflow Studio	Grants access to the Flows design environment in Workflow Studio to create flows and actions. Script action steps require the **Allow Scripting** permission.
Service Catalog	Grants access to work with catalog-related file types such as catalog items, record producers, and variables to add catalog items to apps.
Tables and forms	Grants access to model and layout-related file types such as table columns, form layout, and list layout.
Playbooks	Grants access to work with the Playbooks design environment to create processes. Editing activity subflows or actions requires the **Flow Designer** permission.
Decision Tables	Grants access to work with Decision Tables to create decision logic based on multiple if-then rules.
Notifications	Grants access to create automatic email notifications in apps.

Security/Entitlement permission

The **Manage ACLs and Roles** permission grants access to security management files, such as Access Control Lists and roles.

Programming tools permission

The **Allow scripting** permission grants access to script fields, such as scripting in business rules, UI actions, and client scripts.

Application management permissions

The application management permissions grant access to basic app management functions, like managing collaborators.

Application management permissions

Permission	Description
Delete application	Grants the collaborator within a scoped app rights to delete the application.

Application management permissions (continued)

Permission	Description
Manage collaborators	Grants access to manage and invite collaborators for apps.
Source control	Grants full access to use source control.
Invite collaborators	Grants access to invite developers to collaborate on an app.

Deployment permissions

The deployment permissions grant access to installing, upgrading, and publishing apps.

Deployment permissions

Permission	Description
Upgrade app	Grants access to upgrade the associated application after it has been installed in the current instance.
Submit for deployment	Grants access to submit the associated application for review and deployment.
Publish to app repo	Grants access to publish the associated application to the application repository in the current instance.
Publish to app store	Grants access to publish the associated application to the ServiceNow Store in the current instance.

Remove a user or group from collaboration

Remove users and groups from collaboration to restrict them from owning or editing an app.

Before you begin

Role required: admin or manage collaborator delegated development role

About this task

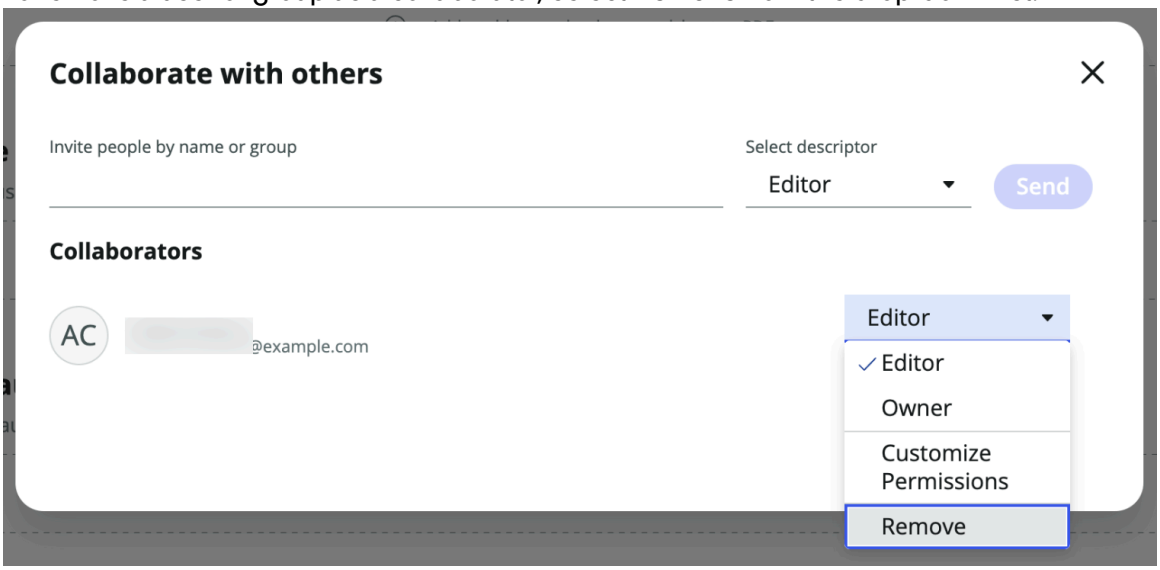
You can remove the final owner and editor for an app. If an app doesn't have any owners and needs one, an admin must add them.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select the Manage collaborators  button.

The Collaborate with others modal appears with a list of the current collaborators.

4. To remove a user or group as a collaborator, select **Remove** from the drop-down list.



Use AES with a Git source control repository

App developers working in App Engine Studio (AES) can manage their data repository in numerous ways.

After your admin has linked an application to Source control, all application developers on a non-production instance can perform these actions:

- Import applications from a Git repository.
- Pull and apply remote changes from a Git repository.
- Commit all local changes on the instance to a Git repository.
- Create tags to permanently link to a given version of an application.
- Create branches to maintain multiple versions of an application simultaneously.

Note:
If you're using source control to collaborate with other developers, only changes that have been checked in are available to other developers. For example, if an admin creates a new flow for an app that's linked to Git, the new flow won't be available in the app for other AES users until the admin checks the flow into Git.

Options available from App Engine Studio

After an application has been linked to source control, application developers can use App Engine Studio to manage the repository. From App Engine Studio, developers can:

- Edit the application repository credentials.
- Commit all local changes on the instance.
- Apply remote changes from the repository.
- Create a branch.
- Switch branches.
- Import an application from a remote repository.

Source control integration does not support managing applications on a production instance. Instead, you can manage applications on a production instance using the application repository,

an update set, or App Engine Studio. For more information about managing applications on a production instance, see Application sharing.

Options available from a Git repository

The ServiceNow platform offers limited support for modifying linked application files outside of an instance. From Git, developers can:

- Move application files to a different Git directory structure.
- Edit application files outside of App Engine Studio.

The system generates a properties text file called `sn_source_control.properties` at the root level of the repository. To move application files to a different Git directory structure, application developers can set the *`path`* parameter to specify the subfolder path containing their application files. For example, if you moved your application to the `src/app` subfolder, set the *`path`* to `path=src/app`.

The system generates a `checksum.txt` file in the Git repository to determine if any application files have been changed outside of App Engine Studio. When the checksum value from the file matches the current checksum value, the integration skips the validation and sanitization process. When the checksum values do not match, the integration validates and sanitizes the application files as part of the source control operation. The sanitization process:

- Creates upgrade log entries for each sanitization action taken.
- Removes unsupported folders and files from the repository.
- Aborts all source control operations when a system application file fails XML schema validation. For example, if a database dictionary record fails XML schema validation, the system aborts all operations.
- Skips the current source control operation when a non-system application file fails XML schema validation.

The source control integration sanitizes only content within the application path listed in the `sn_source_control.properties` file. Repository content outside the application path is ignored.

MID Server support

Use an existing MID Server to connect to a source control repository. Accessing an application through a MID Server enables access to repositories behind a firewall.

Source control operations in App Engine Studio

The source control integration primarily supports operations from App Engine Studio (AES), but can also support some operations directly from the Git repository.

Available source control operations

Operation	Description	Available from
Import from source control	Imports an application from the repository to the local instance. For more information, see Import application or application-customization from source control.	App Engine Studio
Link to source control	Allows developers to manage application changes from a Git repository. For more information, see Link an application or application-customization to source control.	App Engine Studio

Available source control operations (continued)

Operation	Description	Available from
Edit Repository Configuration	Updates the Git repository user credentials. For more information, see Edit a Git repository configuration.	App Engine Studio
Apply Remote Changes	Updates the local version of the application to match the repository version. For more information, see Pull changes from a repository.	App Engine Studio
Commit Changes	Updates the repository version of the application to match the local version. For more information, see Commit changes to a repository.	App Engine Studio
Stash Local Changes	Removes and saves local changes for later work. For more information, see Stash local changes.	App Engine Studio
Switch Branch	Updates the local version of the application to match the repository branch version. For more information, see Switch branches.	App Engine Studio
Create Branch	Creates a branch in the repository to save a different version of the application. For more information, see Create a branch.	• App Engine Studio • Git repository
Create Tag	Creates a tag in the repository to link to a particular application version. For more information, see Create a tag to link to a particular application version.	• App Engine Studio • Git repository
Manage Stashes	Allows developers to apply or delete stashed changes. For more information, see Manage stashes.	App Engine Studio
Create repository	Creates a repository to store application changes.	Git repository
Create credentials	Creates credentials to the repository.	Git repository
Grant access to repository	Provides read and write access to the repository tied to a specific set of credentials.	Git repository

Import application or application-customization from source control

Import an app or app customization from a source control repository into App Engine Studio (AES) to continue developing it on this instance.

Before you begin

- Role required: admin
- Verify that the non-production instance has network access to the Git repository.

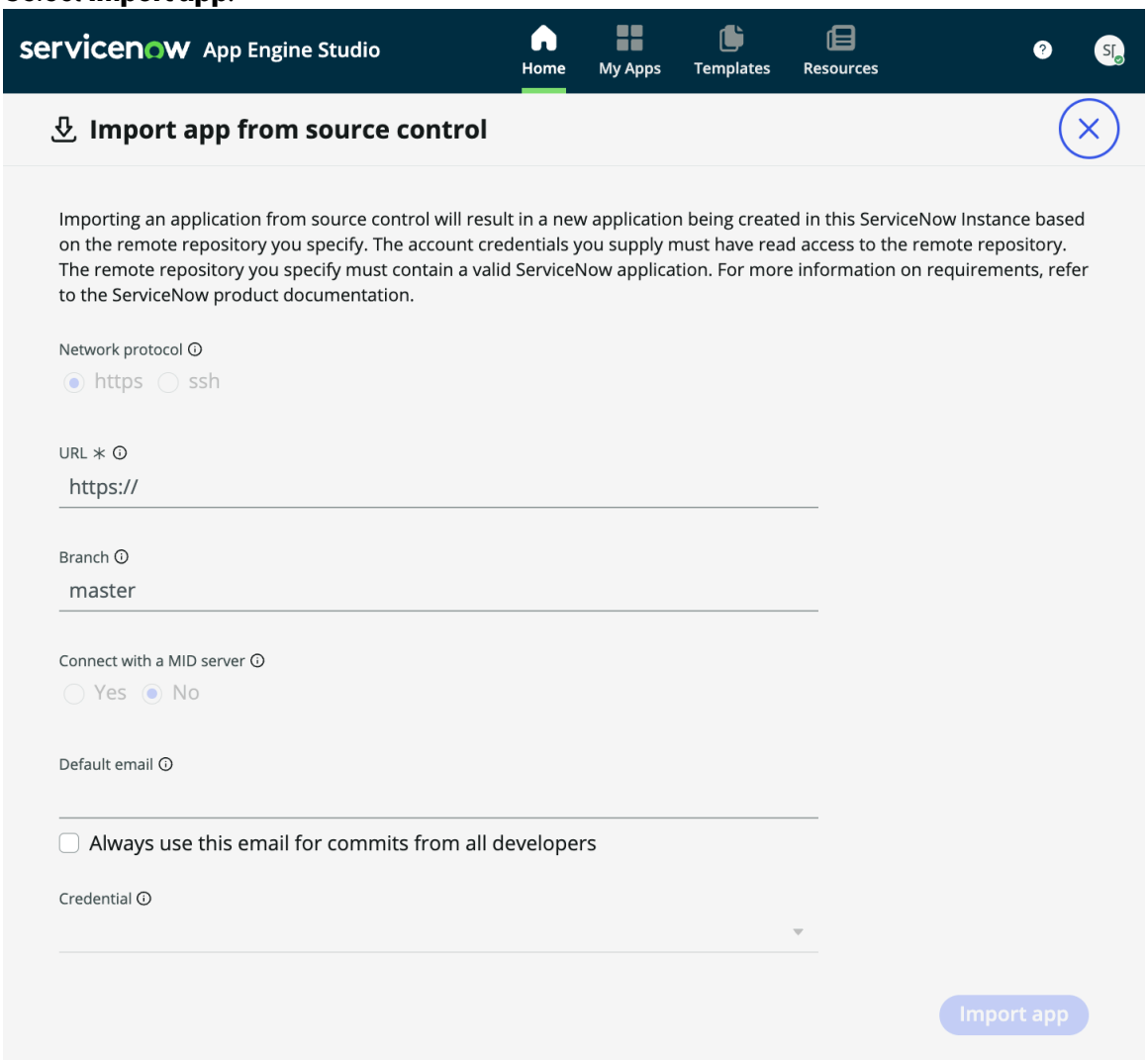
- Verify that the repository contains a valid application.
- Ensure that users add the email address to their respective Users table [sys_user] record that they use in their commits to the Git repository.
- Learn more about application-customizations Manage customizations to applications.

About this task

The source control integration does not support importing an application on a production instance. Instead install applications on a production instance from the application repository, an update set, or the App Engine Studio.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. Select **Import app**.



3. On the form, fill in the fields.

Import from source control fields

Field	Description
Network protocol	Https or ssh credential type that enables secure channel data exchange.

Field	Description
URL	The URL to the Git repository where the application files reside. **Note:** If the Git repo URL for SSH provided by your Git server does not work, check with your Git server owner or provider for the correct URL. There may be additional specifications such as scheme protocol prefixes, port numbers, and so on, required for your Git repo URL to function.
Branch	The repository branch to work on within the application. **Note:** The default branch is named after your instance. If you do not choose a name, the branch defaults to **master**.
MID Server Name	Select an existing MID Server to link to a Git repository stored behind your corporate firewall. **Note:** Use a separate MID Server to prevent conflicts with Discovery activities. See MID Server for more information.
Default email	The committer email address is defined by the sys_user record if available. But if a committer's sys_user record email field is empty, the system generates an alternate email (username@instancename.service-now.com). You can also enter a default email address and change it later. To use that default email address in all cases, select the check box.
Credential	Select the credential for your Git repository. For more information, see Getting started with Credentials. **Note:** If you select the ssh network protocol, enter a valid credential of the SSH Private Key type. If you select the https protocol, enter a valid credential of the Basic Auth Credentials type.

Note:
All application developers on the instance share the credential used to link a Git repository to an application.

4. Select **Import app**.
   The system compares the checksum in the `checksum.txt` file to current checksum. When the checksum values match, the integration skips validation and imports the application. When the checksum values do not match, the integration first validates and sanitizes the application files before importing them.
5. Select **Select Application**.
   App Engine Studio displays the application as a new choice in the Switch Applications window.

What to do next

- Review the upgrade logs for any sanitization applied to application files during the import.
- Select the imported application to edit it.

Related topics

MID Server

Getting started with credentials

Work with changes in Git

Developers using App Engine Studio (AES) can pull and commit changes in their Git repository.

Note:
If you're using source control to collaborate with other developers, only changes that have been checked in are available to other developers. For example, if an admin creates a new flow for an app that's linked to Git, the new flow won't be available in the app for other AES users until the admin checks the flow into Git.

Pull changes from a repository

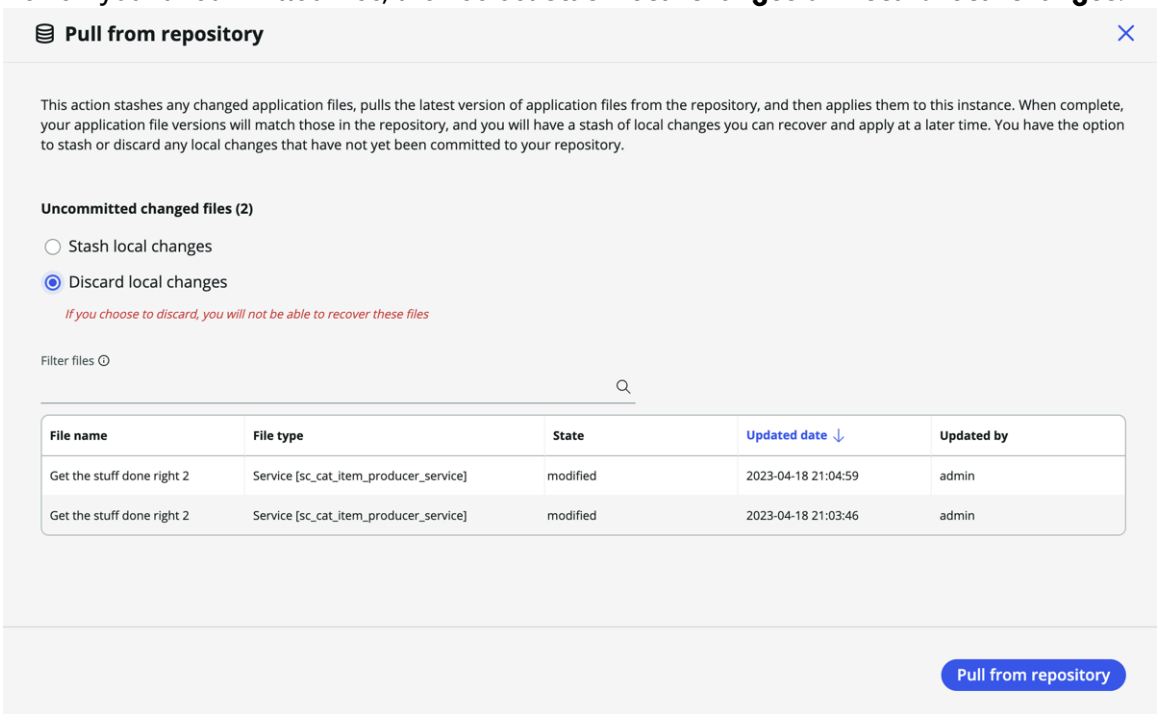
App developers using App Engine Studio (AES) can pull changes from a linked Git repository to apply remote changes to the local instance.

Before you begin

- Role required: admin
- Link an application or application-customization to source control

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select **Source control** > **Pull from repository**.
4. Review your uncommitted files, then select **Stash local changes** or **Discard local changes**.



5. Select **Pull from repository**.
   The following operations occur:

- The system fetches the most recent changes from the remote repository.
- The system applies the remote changes to the instance.
- The system identifies any change conflicts requiring resolution.
  If there are conflicts, the system displays the **Resolve Conflicts** window.

Delta loading is enabled by default in sys properties so your data isn't removed. You can disable this feature if you want data automatically deleted.

What to do next
Resolve any change conflicts.

Commit changes to a repository

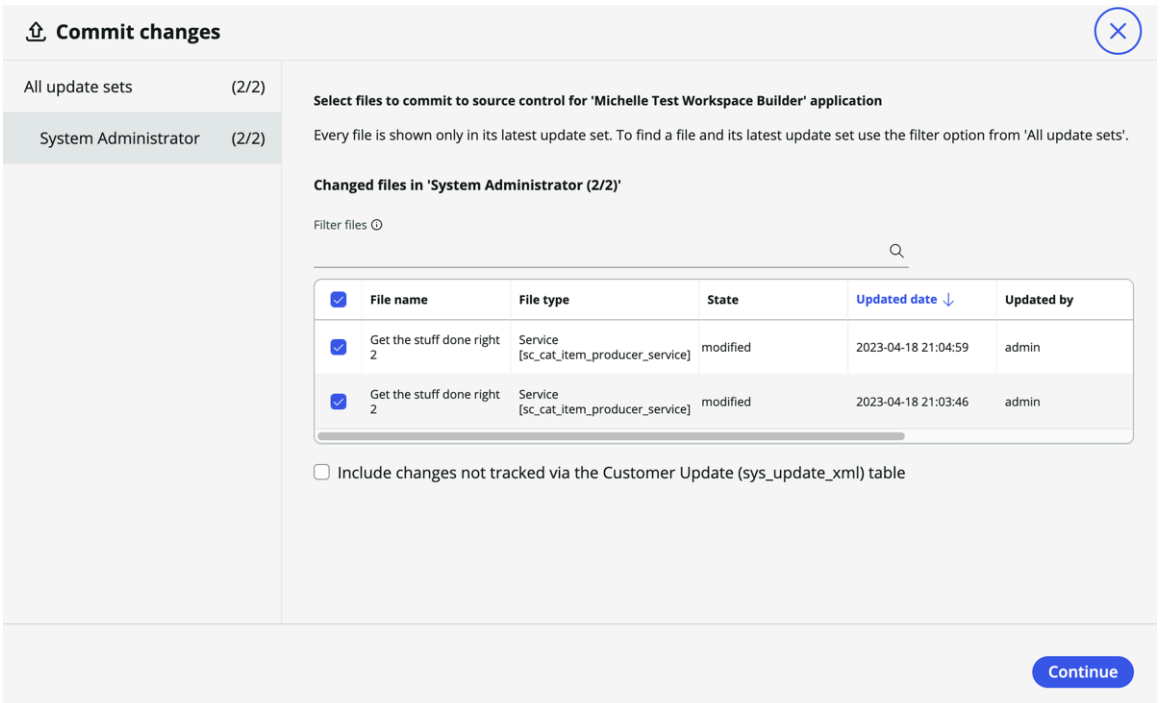
Commit changes made in your application in App Engine Studio (AES) to a linked Git repository. You can either select a few changes to commit, or commit all changes on the instance at once.

Before you begin

- Role required: admin
- Link an application or application-customization to source control

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select **Source control** > **Commit changes**.
   The system displays the **Select files to commit to source control** window. The file changes from all the updates sets display. By default, the file changes from the current update set display.



4. Select the file changes you wish to commit.
5. To include untracked changes, select the **Include changes not tracked via the Customer Update [sys_update_xml] table** check box.

- The default for this check box is set via the **glide.sourcecontrol.default_commit_mode** property.
  - Property can be set to **include_untracked** or **exclude_untracked**.
  - The **include_untracked** mode commits the updates to the application that do not generate sys_update_xml records, as well as any user-selected updates.
  - The **exclude_untracked** mode commits only updates selected by the user in the **Select files to commit to source control** dialog.
- The base system setting for the property is **exclude_untracked**.
- To hide the check box and use the value of the **glide.sourcecontrol.default_commit_mode** property, create the **sn_devstudio.vcs.allow_commit_mode_selection** property and set it to false.
- Checking this check box may incur a performance penalty.

Note:

Commits always occur in **include_untracked** mode in the following cases:

- Linking to source control for the first time. (To learn more, see Link an application or application-customization to source control.)
- Publishing an application that's linked to source control from App Engine Studio.
- Selective commit mode is disabled.

6. Select **Continue**.
7. In **Commit comment**, enter a comment for the changes.
8. Select **Commit Files**.
   The following operations occur:
   - The system identifies all local changes.
   - The system commits all local changes to the remote repository.

Note:
For list of known files that don't have customer update records and are untracked, see Customer Updates table.

Create versions and branches in Git

App developers using App Engine Studio (AES) can create versions and branches in their Git repositories.

Create a tag to link to a particular application version

Create a tag in the repository to link to a particular app version in App Engine Studio (AES).

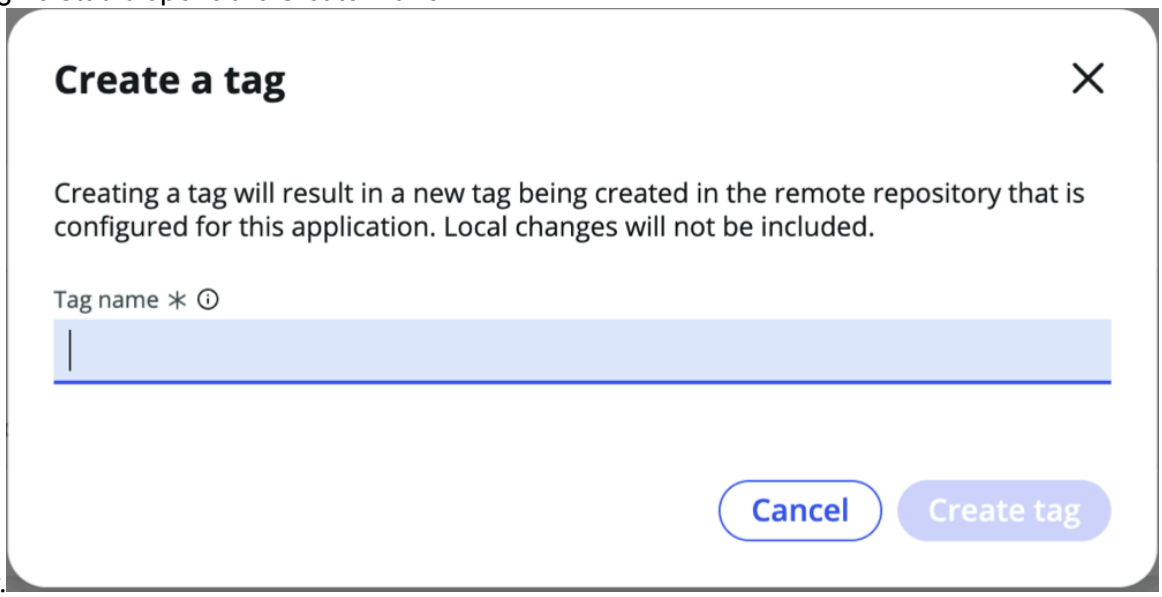
Before you begin

- Role required: admin or sn_group_creator.app_creator
- Link an application or application-customization to source control

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.

3. Select **Source control** > **Create tag**.
   App Engine Studio opens the Create Branch window.



4. Enter the **Tag Name**.
5. Select **Create Tag**.
   App Engine Studio creates the tag.
6. Select **Close**.

What to do next

Commit changes to the new branch.

Switch branches

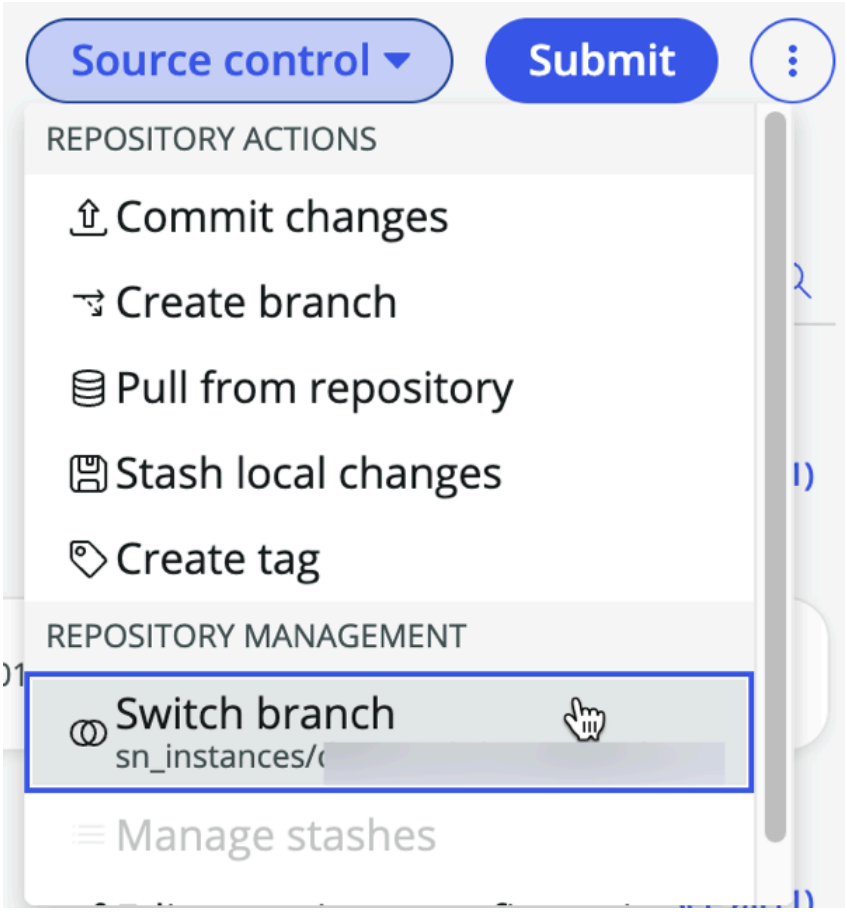
Application developers can switch to a different repository branch to work on another version of the application using App Engine Studio (AES).

Before you begin

- Role required: admin
- Git repository with one or more available branches.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select **Source control** > **Switch branch**.



The system displays the Switch Branch window.

4. **Optional:** If there any local changes on the instance, you can save or discard them.

   Note:
   Use caution when discarding local changes. Since all application developers share repository credentials, there is no way to discard just one set of user changes. Note you cannot later restore discarded changes.

5. Select the branch you want to switch to.
6. Select **Switch Branch**.
   App Engine Studio updates the local application to match the branch version from repository.

Create a branch

Application developers can create a branch to work on a new version of an existing app in App Engine Studio (AES).

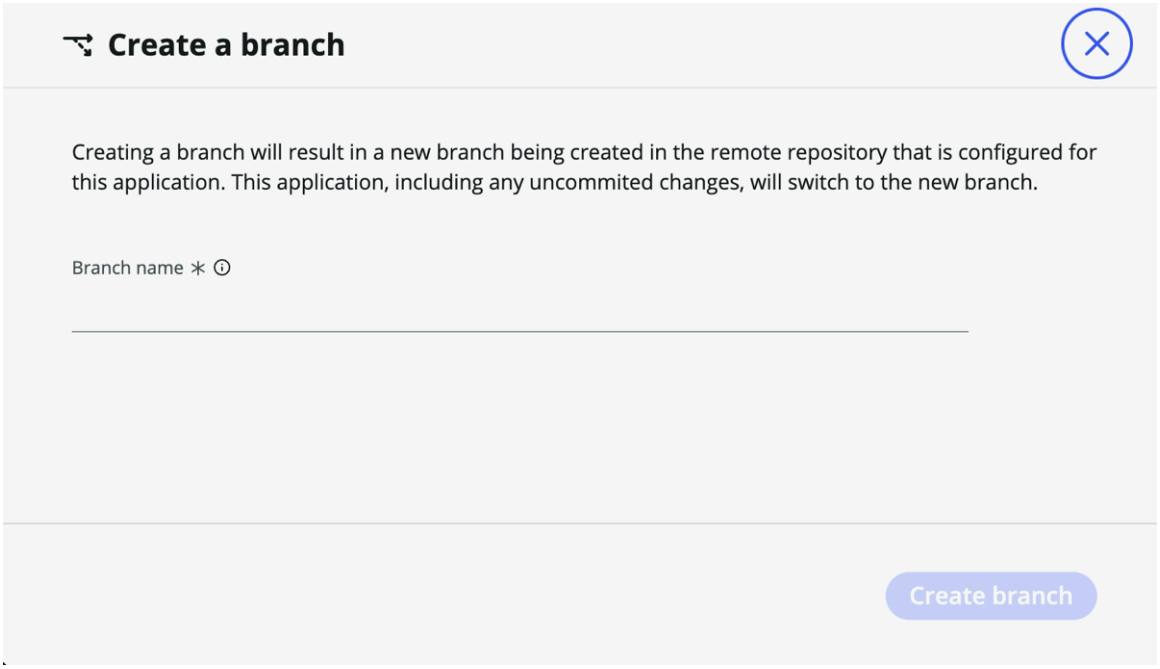
Before you begin

- Role required: admin or sn_group_creator.app_creator
- Link an application or application-customization to source control

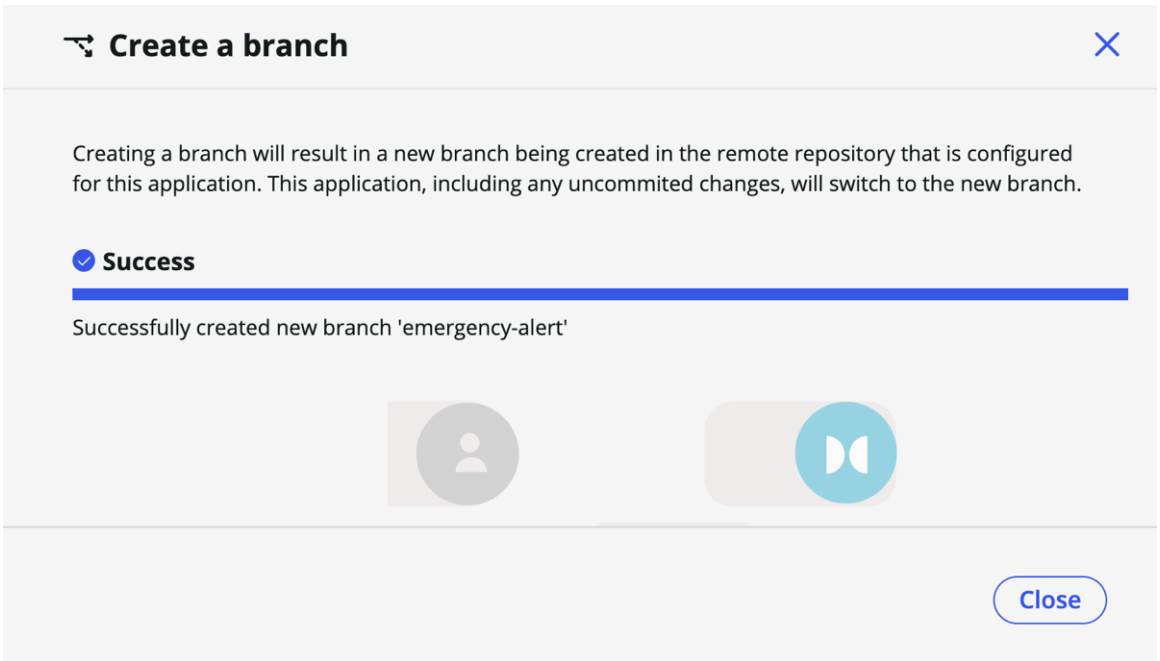
Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select **Source control** > **Create branch**.

App Engine Studio opens the Create Branch window.



4. Enter the **Branch Name**.
5. **Optional:** To create a branch from a tag, select the **Create from Tag** drop-down list and select an existing tag.
6. Select **Create Branch**.
   App Engine Studio creates the branch.



7. Select **Close**.

What to do next

Commit changes to the new branch.

Set the default branch

Set a default branch when you want to use a branch other than main for new changes or for your main App Engine Studio (AES) development repository.

Before you begin

- Role required: admin
- Link an application or application-customization to source control

Procedure

1. Follow the steps to Add a system property.
2. Add the **glide.source_control.default_branch_name** property, and specify the default branch name of the Git source control repository to work from (pull requests, code commits, etc.). Application developers' work is managed from and saved into the default branch if not otherwise specified. If not changed, this value defaults to sn_instances/<instance_name>.

Stash changes

Developers can remove and save changes locally to apply them later, and manage stashed changes from App Engine Studio (AES).

Stash local changes

App developers can remove and save changes locally to apply them later in App Engine Studio (AES).

Before you begin

- Role required: admin
- Link an application to source control
- Change one or more application files

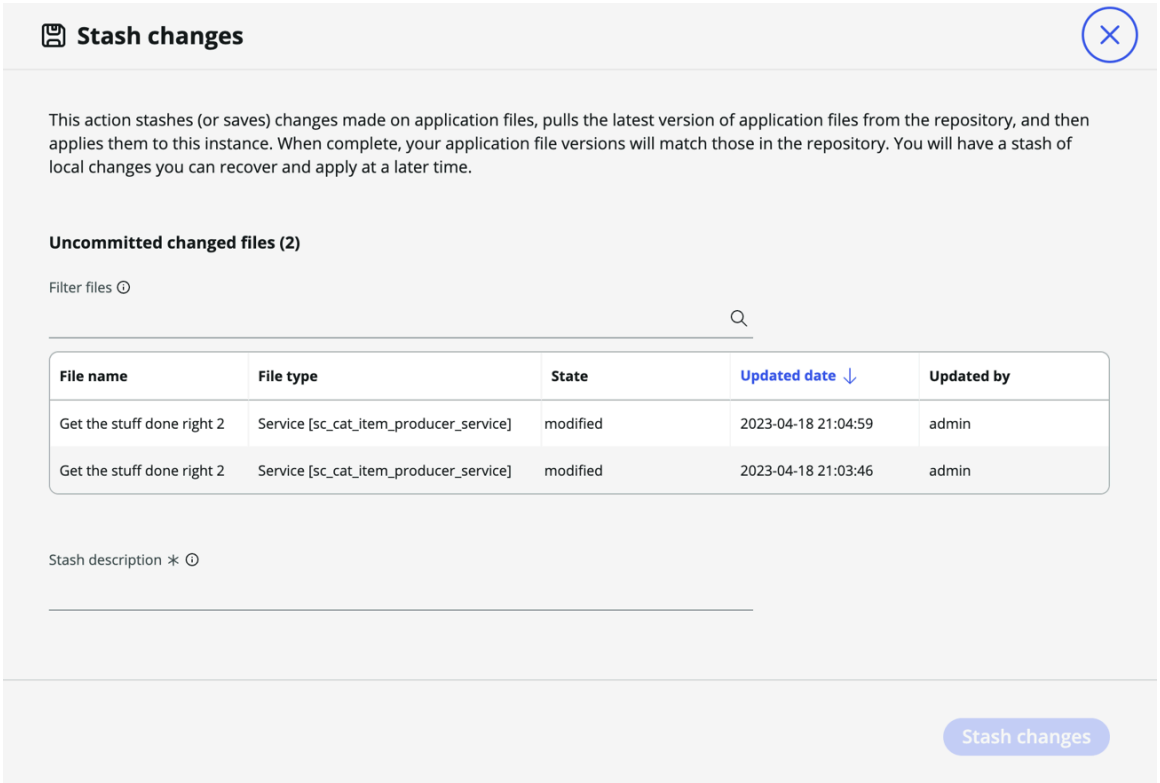
About this task

Stashing changes removes them from the current application and saves them for a developer to later apply or delete.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select **Source control** > **Stash local changes**.
   The system displays a list of locally changed files.
4. Enter your description.
5. Select **Stash Changes**.

The system saves the current changes and displays a success message.



What to do next

- Close dialog
- Manage stashes

To learn more, see Getting started with credentials

Manage stashes

App developers can apply or delete stashed changes from App Engine Studio (AES).

Before you begin

- Role required: admin
- Link an application to source control.
- Stash one or more application file changes.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Select **Source control** > **Manage stashes**.
   The system displays a list of locally stashed changes.
4. Select the action next to the stash you want to manage.

Option	Description
Apply	Commits the stashed changes to the application and checks for conflicts.
Delete	Removes the stashed changes.

Resolve conflicts

App developers can select the app file version to use when applying remote or stashed changes in App Engine Studio (AES).

Before you begin

- Role required: admin
- Link an application to source control
- Apply a stashed change

About this task

Conflicts occur when there are multiple change versions of the same application file: one set of changes in the remote or stashed version and another set of changes in the local version. App Engine Studio requires developers to resolve conflicts before applying remote or stashed changes.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. Pull from a repository or stash local changes.
   If the system identifies a conflict, it displays the Resolve Conflicts dialog.
4. Select how to resolve the conflict.

Option	Description
Select an action	Apply or discard all stashed changes. Go to step 3.
Manually merge changes	Individually select which changes to apply. Go to step 6.

5. If you want to apply or discard all stashed changes, select an **Action**.

Option	Description
Take Stashed Changes	Applies the application file version from the stashed changes.
Discard Stashed Changes	Applies the application file version from the most recent pull from the repository.

6. Select **Apply Stashed Changes**.
   The system applies the selected changes.
7. Select **Close Dialog**.
8. If you want to merge the conflicting changes, select **Manually Apply**.
   The system displays a list of version differences by field.

9. Select the field values you want the merged application file to have.
10. Select **Save Merge**.
    The system applies the selected changes.

View commit history

App developers can view the commit history of apps linked to a source control repository in App Engine Studio (AES).

Before you begin

- Role required: admin
- An existing link to a Git repository

Procedure

1. Navigate to **All** > **Source Control** > **View History** .
   The system displays the History window.

2. Select the commit sort order type.

Option	Description
Date	Sort by commit date.
Committer	Sort by user name.

3. Select the sort order direction.

Option	Description
Descending	Sort dates from the most recent to oldest date. Sort user names reverse-alphabetically from Z to A.
Ascending	Sort dates from the oldest to most recent date. Sort user names alphabetically from A to Z.

   The system sorts commits by the selected sort order.

4. Select a commit.
   The system displays the commit details for the selected commit.
5. Review the commit details.

Commit Details

Field	Description
Committer	The user who committed the change.
Date	The date-time stamp of the commit.
SHA-1	The secure hash value identifying this commit in the repository.
Message	The commit message associated with this commit.

Field	Description
Files	The list of application files changed in this commit.

6. Close the History window.

Move application files in a Git repository

Move application files linked to source control to any folder of the repository when working in App Engine Studio (AES). Allow application developers to store supporting content such as automated tests in the same repository as the applications they support.

Before you begin

- Link an application or application-customization to source control
- Role required: Source control credentials with write access

About this task

Linking an application to source control generates a properties text file called `sn_source_control.properties` at the root level of the repository. The properties file specifies the folder containing the application files. The integration tracks changes to these application files by generating a `checksum.txt` file. When the checksum matches, the integration skips the validation and sanitization process. When the checksum does not match, the integration validates and sanitizes the application files as part of the source control operation. The integration ignores all repository content outside the application path.

Note:
You can set system properties **glide.source_control.checksum_required** to enable optional checksum validations and sanitizations and **glide.source_control.checksum_quick_install** to bypass sanitization steps on checksum matches. See Available system properties for more information.

Procedure

1. Login to the source control repository linked to the application.
2. Create the destination folder where you will move the application files.

   Example
   For example, create the folders `src/app`.
3. Move the folder containing your application files to the destination folder.

   Example
   For example, move the folder `demo_my_app` to `src/app`.
4. Navigate to the root level of the repository.
5. Open the `sn_source_control.properties` text file in a text editor.
6. For the `path=` property, enter the folder path where you moved the application files.

   Example
   For example, enter `path=src/app`.
7. Save the properties file.

What to do next

Login to your instance and perform Source control operations in App Engine Studio from App Engine Studio.

Publish your app

When you have completed your app in App Engine Studio (AES), you need to submit it for approval and publishing. The process you use depends on whether you are linked to source control.

Submit your app for approval and publishing

Start the process of getting the application you built in App Engine Studio (AES) published by submitting it for administrator review.

Before you begin

Role required: delegated developer with permissions (**Submit for deployment**, **Publish to app repo**, or **Publish to app store**, depending on the desired action)

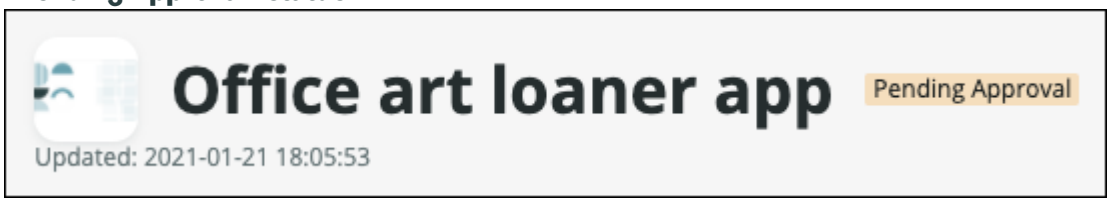
Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. From the app home, select **Submit**.
4. On the dialog box that appears, review the submission details and then select **Submit**.

What to do next

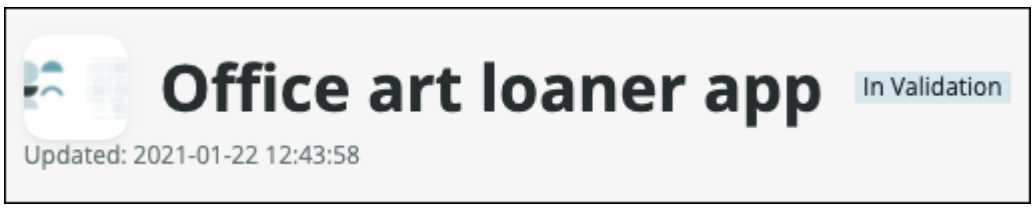
The administrator reviews the submitted application and checks for potential issues. You can check the status of your application from the application home in App Engine Studio.

"Pending Approval" status



After a reviewer begins testing the application, the status changes to `In Validation`.

"In Validation" status



An administrator may provide you with test accounts for different roles in your application. Log in with each of these accounts to check that the application works as expected.

If the reviewer rejects the application, the status changes to `Rejected`. You may also receive an email that includes the reviewer's comments. Use the feedback to improve your application, and then submit the application again.

"Rejected" status

If the application passes testing, the administrator publishes the application and the status changes to `Published`. The administrator on-boards the team that requested the application. For example, if you specified a certain security role for the application, the administrator assigns that role to the relevant users.

"Published" status

Related topics

Delegate developers using AES

Publish an app from App Engine Studio when linked to source control

You can publish a custom application from App Engine Studio (AES) when linked to source control.

Before you begin

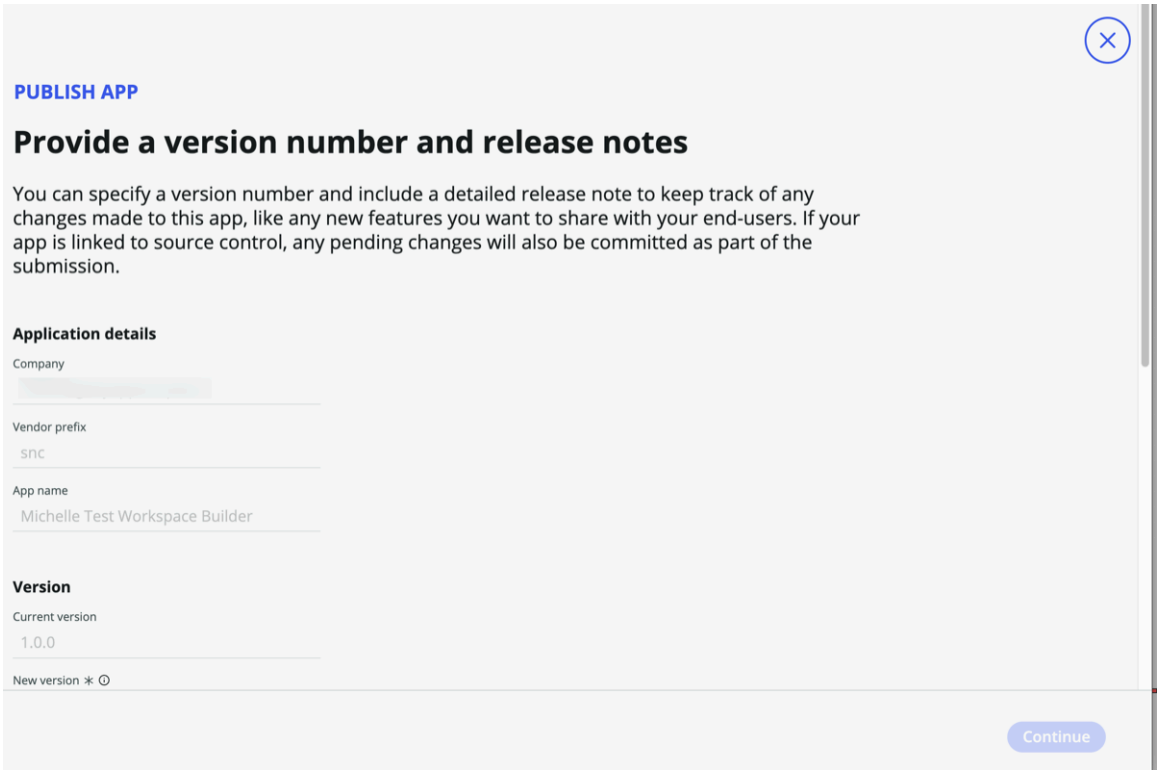
Role required: admin

About this task

When you publish an application from App Engine Studio that is linked to source control, there is a different outcome than when you publish via the sys_app or sys_store_app **Publish** related link.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. From the app home, select **Publish**.
   App Engine Studio displays the publish app fields.



4. Enter the following field values.

Provide a version number and release notes

Field	Description
Version	View your current app version and provide a new version number for your application. **Note:** To change the version number, you must have one of the following deployment permissions: **Submit for deployment**, **Publish to app repo**, or **Publish to app store**
Release notes	Provide a brief description of the changes made to your application.
Publish options	Select if you want the application to be published to your application repository or the ServiceNow Store.

Note:
All application developers on the instance share the credential used to link a Git repository to an application.

5. Select **Continue**.
6. The current state of the application is committed to source control, including any untracked or uncommitted changes.
The value of the **glide.sourcecontrol.default_commit_mode** property is ignored.
This occurs because when the application is published, all the untracked and uncommitted changes are also published. Therefore, the state of the application in the Git repository matches what is published. See the Commit changes to a repository topic for more information about the **glide.sourcecontrol.default_commit_mode** property.

7. A source control tag is created for the new version and the application is published. If needed, the sys_app record is updated with the new store correlation ID.

> **Note:**
> If your application is linked to source control and you publish a new version outside of App Engine Studio, a source control commit and tag are not created.

Managing app deployments using Pipelines and Deployments

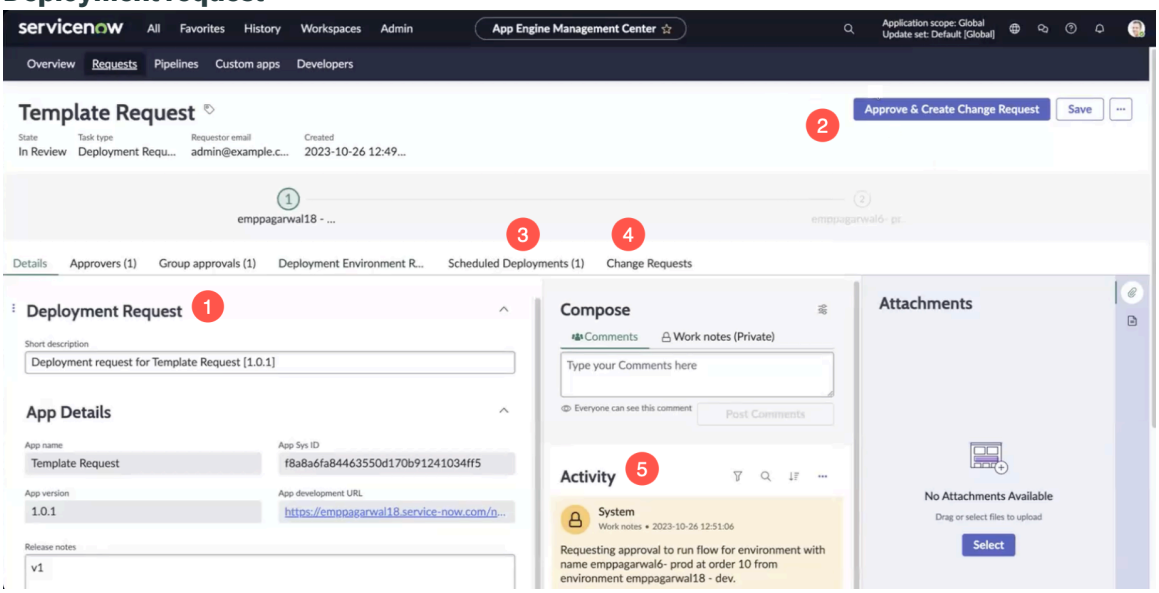
Review the applications developers build using App Engine Studio (AES) so you can deploy with confidence.

Deployment requests

After a developer submits an application in App Engine Studio, a deployment request is created. A deployment request is a record to track the review of submitted applications.

A reviewer can deploy the app to your test environment, reject it, or publish it, all using the App Engine Management Center. If you integrate an existing Change Management program with AEMC, and your app is ready to deploy to production, you can create a change request and deploy the app within a specified change window.

Deployment request



1. View details of the deployment request and the application being deployed.
2. Select **Approve & Create Change Request** when the app is ready for deployment. This action begins the Change Management processes you configure using Guided Setup.

> **Note:**
> If you don't have an existing Change Management program integrated with Pipelines and Deployments, select **Approve & Deploy App** to move the deployment to the next stage.

3. View and edit details of the deployment on the **Scheduled Deployments** tab.
4. View details of the change request on the **Change Request** tab. Depending on the role you have, you may be able to change some details of the request.
5. Watch the **Activity** stream for test results, change request progress, and other deployment details.

Note:
If you submit deployment requests without upgrading all instances in the pipeline, your existing pipeline continues to be used.

For more information on reviewing a deployment request, see Deployment Request form in the Pipelines and Deployments app.

Testing an application

Before you publish a submitted application, test it in a non-production instance. To begin testing, an admin must open the deployment request and select **Approve**. The pipeline record is read and determines the next state for the request.

The goal of testing the application is to ensure the viability of the production instance. When the app has transitioned to the Testing state, ServiceNow Automated Test Framework tests run if you have enabled the appropriate properties.

If the application doesn't pass testing, then you reject the deployment request.

Deploying to a test environment

When you deploy or install an application to a test environment, two jobs are automatically executed:

- Application Deployment Test Suite
- Scoped App Definitions instance scan

These tests can be useful to the administrator for diagnosing issues before an app is deployed. For more information about what goes on during an application deployment, see Pipelines and Deployments workflow version 24.1.2.

Jobs automatically run during deployment

Job	Description
Application Deployment Test Suite	A set of Automated Test Framework (ATF) tests. The suite consists of a single test called **Log**. When the test is run, scan results are logged to the Test Results [sys_atf_test_result] table. **Note:** The following two system properties must be enabled to run the Application Deployment Test Suite: • sn_atf.runner.enabled • sn_atf.schedule.enabled If they aren't enabled, only the instance scan runs. For more information, see Enable Automated Test Framework (ATF) properties.
Scoped App Definitions instance scan	Instance scans aid in diagnosing health issues on a non-production instance, and are

Jobs automatically run during deployment (continued)

Job	Description
	useful for addressing best practices. For more information, see Instance Scan. The instance scan checks all tables in the app that extend Task [task].

The admin can view the results of the scans for troubleshooting purposes by selecting the **Deployment Environment Results** tab in the deployment request.

Simply open the record of the instance. The results are split into three tabs:

- **ATF Results**
  - Test suite name
  - Test URL
  - Success and failure counts
  - Overall error count
- **Instance Scan Results**
  - Scan suite name
  - Scan URL
  - Finding count for the scan
- **Results (JSON)** includes unformatted JSON code of the ATF and instance scan results, as well as any errors identified during the scans. This JSON can also be found in the **Notes** related list.

Publishing an application

If an application passes testing, open the deployment request and deploy the application to your production environment. It is then available to all employees in your organization. For more information, see Managing deployments using pipelines in AEMC.

Guides for more information

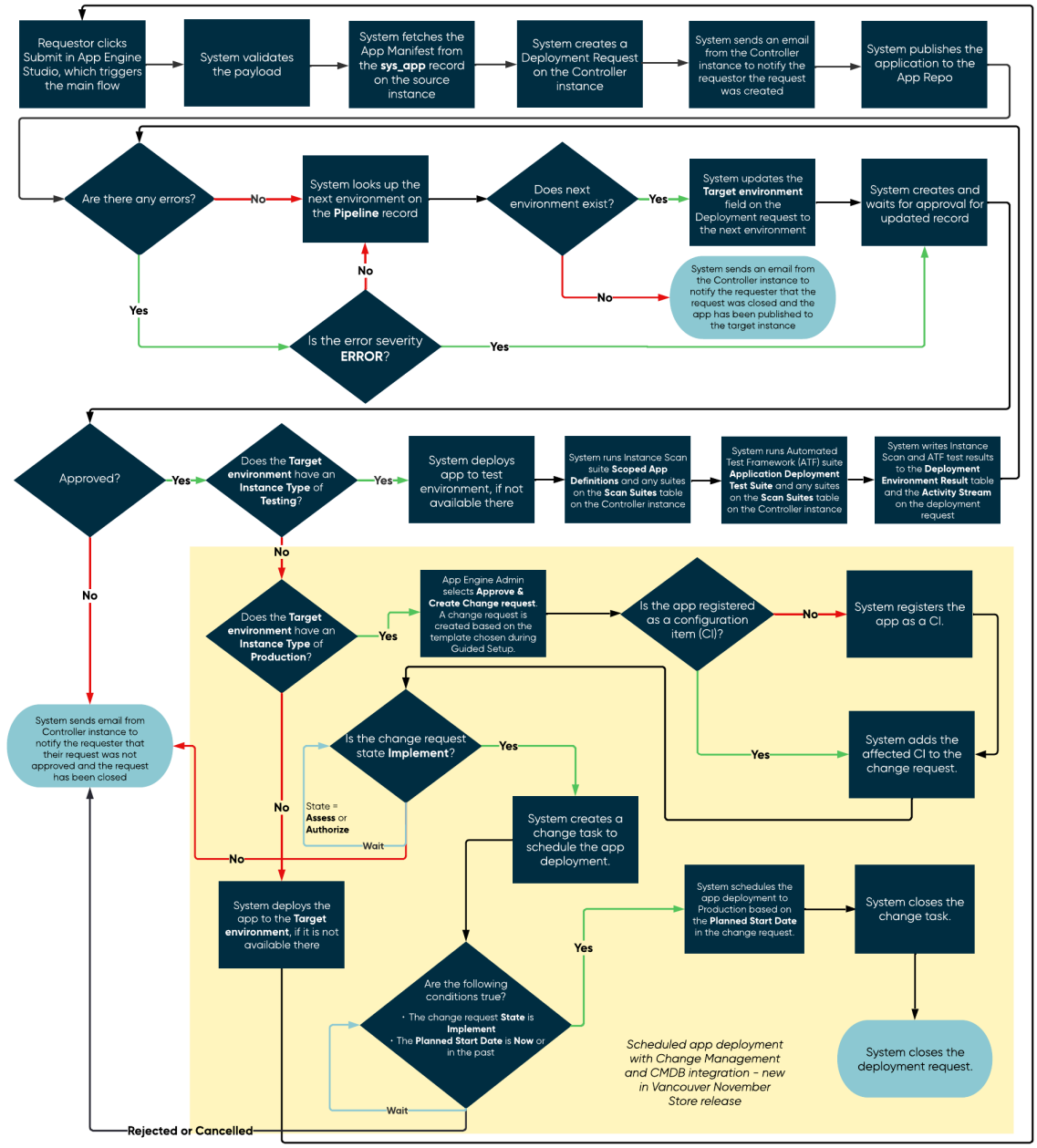
If you need more information, you can try these external guides sponsored by ServiceNow.

Learn more about Pipelines and Deployments	Additional ServiceNow resources
The Pipelines and Deployments app is used to deploy the apps you are creating using App Engine Studio between instances in a pre-defined order. Unlike previous versions, Pipelines and Deployments allows you to deploy your apps to an unlimited number of instances for app creation, testing, staging, and production.	 Promoting apps through the AES pipeline  Create apps fast with App Engine Studio

Pipelines and Deployments workflow version 24.1.2

As you manage requests for app deployment in App Engine Management Center (AEMC), use this workflow to understand how app deployments move through your pipelines in version 24.1.2, released in November 2023.

Pipelines and Deployments workflow



In this workflow:

1. The requester selects **Submit** in App Engine Studio, which triggers the main flow.
2. The system performs the following tasks behind the scenes:
   a. Validates the payload.
   b. Fetches the App Manifest from the **sys_app** record on the source instance.
   c. Creates a deployment request on the controller instance.

d. Sends an email from the controller instance to notify the requester that the request was created.

e. Publishes the application to the Application Repository.

3. The system performs different actions depending on if there are errors or not during publication.

a. If there are errors in the app publication, and the error severity is **Error**, then the system creates and waits for approval for the updated record.

b. If there are no errors, or if there are errors, but the error severity isn't **Error**, then the system looks up the next environment on the Pipeline record.

i. If the next environment doesn't exist, the system sends an email from the Controller instance to notify the requester that the request was closed and the app has been published to the target instance. This action ends the workflow.

ii. If the next environment does exist, the system updates the **Target environment** field on the deployment request to the next environment. Then, the system creates and waits for approval for the updated record.

4. The system performs different actions depending on if the new record was approved.

a. If the record isn't approved, the system sends an email from the Controller instance to notify the requester that the request wasn't approved and the request has been closed. This action ends the workflow.

b. If the record is approved, and if the **Target environment** is **Testing**, then the system performs the following actions:

i. Deploys app to test environment, if the app is not available there.

ii. Runs the Scoped App Definitions Instance Scan and any other suites on the Scan Suites [[scan_suites]] table on the Testing instance.

> **Note:**
> The Scan Suites table should be populated on the Controller instance.

iii. Runs the Application Deployment Test Suite Automated Test Framework (ATF) suite and any suites on the Scan Suites [scan_suites] table on the Testing instance.

iv. Writes Instance Scan and ATF test results to the Deployment Environment Result table and to the Activity Stream on the deployment request.

v. Returns the workflow to step 3, where it checks for errors.

c. If the record is approved, and the **Target environment** is **Production**, the app begins the scheduled deployment process with Change Management integration.

i. The App Engine Admin selects **Approve & Create Change request**. A change request is created based on the template chosen during Guided Setup.

ii. The system performs different actions depending on if the app is registered as a configuration item (CI).

1. If the app is not registered as a CI, the system registers the app as a CI and then adds the affected CI to the change request.

2. If the app is registered as a CI, the system adds the affected CI to the change request.

iii. The system performs different actions depending on if the change request is in the **Implement** state.

1. If the change request state is not **Implement**, and the state is not **Assess** or **Authorize**, the system sends an email from the Controller instance to notify the requester that their request was not approved and has been closed. This ends the workflow.
2. If the change request state is not **Implement**, and the state is **Assess** or **Authorize**, the system waits for the state to be **Implement**.
3. If the change request is in the **Implement** state, the system creates a change task to schedule the app deployment.

iv. If the change request state is **Implement** and the **Planned Start Date** is not **Now** or in the past, the system waits for those two conditions to be true

v. If the change request state is **Implement** and the **Planned Start Date** is **Now** or in the past, but the request is **Rejected** or **Cancelled**, the system sends an email from the Controller instance to notify the requester that their request was not approved and has been closed. This ends the workflow.

vi. If the change request state is **Implement** and the **Planned Start Date** is **Now** or in the past, the system schedules the app deployment to Production based on the **Planned Start Date** in the change request. The system closes the change task, and then closes the deployment request. This ends the workflow.

d. If the record is approved, and the **Target environment** isn't **Testing** or **Production**, then the system deploys the app to the target environment, if it isn't available there.

The workflow starts over when a requester selects **Submit** again in App Engine Studio.

Remove an app from App Engine Studio

If you have the appropriate permissions, you can remove an application from App Engine Studio (AES).

Before you begin

Role required: admin, delete application, or delegated developer with permissions. For more information, see Delegate developers using AES.

About this task

Delegated developers can delete applications based on their permissions. To request delete permissions, contact your administrator.

Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, select the Edit application properties icon ( ).
4. Select **Delete application**.
5. On the dialog box, enter `delete` and then select **Delete**.

Managing app development using the App Engine Management Center

Track and manage your App Engine Studio (AES) and Creator Studio requests, deployments, applications, and collaborative developers using the App Engine Management Center (AEMC) in your production instance. Additionally, AEMC allows admins to manage app development from intake through production.

AEMC uses Pipelines and Deployments to deploy applications through different instances. Pipelines and Deployments uses the Application Repository to manage these deployments. For more information, see ServiceNow application repository.

Each application can either be deployed using the Application Repository or System Update Sets. If you've used Update Sets in the past, but want to switch to using the Application Repository, you can do a one-time conversion to deploy the app using the Application Repository instead. All apps don't have to follow the same deployment. For more information, see Convert custom applications to upgrade from the application repository and System update sets.

Additional resources for App Engine Management Center

Learn more about App Engine Management Center	ServiceNow resources
Although you can use AEMC in any instance, for full functionality, you must install AEMC in the production instance. You must also set up the deployment pipelines with the controller instance in production. For more information, see Configure Pipelines and Deployments.	 Govern low-code app development at scale with App Engine Management Center  Creator toolbox | App Engine Management Center  Govern low-code app development at scale with App Management Center

Navigating AEMC

From the AEMC Overview screen, you can navigate to four additional views: Requests, Pipelines, Custom apps, and Developers by selecting icons in the primary navigation bar.

AEMC primary navigation bar

Page	Description
Overview	High-level view of all intake, app, collaboration, and deployment requests, with a graph showing new requests of each type over the last 90 days. You can also see a high-level view of your pipelines and application metrics.
Requests	Full list of each request type broken down by approval status.
Pipelines	High-level view of the number of open deployment requests in each instance and within each pipeline. Selecting a pipeline or an environment in a pipeline displays all active deployment requests.

AEMC primary navigation bar (continued)

Page	Description
	Note: If you view deployment requests in a production instance, all **Closed - Published** deployment requests for that environment are listed.
Custom apps	Charts and graphs illustrating the status of your active applications, and a list of all custom apps in development or production. You can view details for individual apps, including usage information, deployment history, and collaborators.
Developers	Charts and graphs illustrating total and active developers, as well as developers by department. You can view details for individual developers, including the apps for which they are collaborators and their request history.

Managing requests using AEMC

You can track and approve or reject Intake, App, Collaboration, and Deployment requests using the App Engine Management Center (AEMC).

To work more efficiently, you can also filter and search for requests.

Manage intake requests

View details of intake requests submitted by developers in App Engine Studio in the App Engine Management Center (AEMC) dashboard, and approve or reject them.

Before you begin

Before intake requests can be displayed in AEMC, you must confirm that the Application Intake application has been properly configured. For more information, see Configure Application Intake. For more information about how to submit intake requests, see Submit your idea for app development.

App requests submitted from Creator Studio restricted users are managed on the App tab on the Requests page. For more information, see Manage app requests from Creator Studio.

Role required: sn_app_eng_notify.app_engine_admin

Procedure

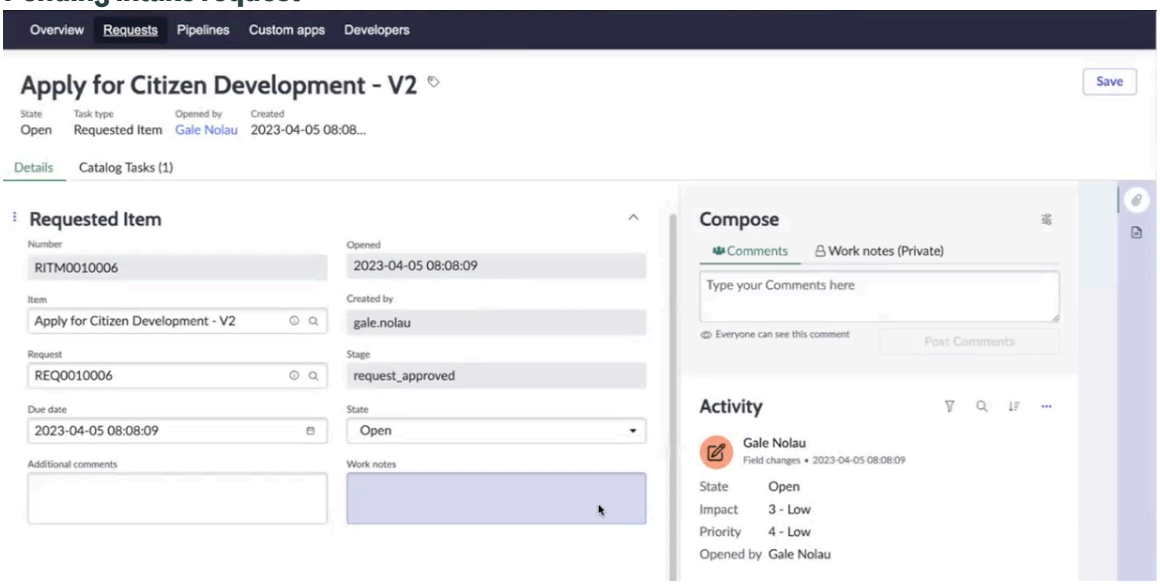
1. Navigate to **All** > **App Engine** > **Administration** > **App Engine Management Center**.

   The AEMC dashboard shows the number of pending intake requests.

   Note: To view a list of all intake requests, either select the number of requests or select the Requests page.

2. Select an intake request to review its details.

Pending intake request



3. Return to the **Details** tab and review the request.

> **Note:**
> The **Administration** section includes a **Development instance** field. Make sure that the development instance the developer intends to use has been selected. Also, if the user isn't defined on that instance, a message displays and you must manually add the user to that instance. For more information, see Create a user.

4. Use the **Permission type** list in the Administration section to select one of the following options for provisioning groups to this application.

5. Approve or reject the request.
   - If the request is approved, the user is added to the App Engine Studio User group and granted the App Engine Studio User role. The AES User role enables developers to begin creating their application in AES in the approved instance. The user also receives an email notification with a link to the provisioned instance. For more information about the AES User role, see Components installed with App Engine Studio.
   - If the request is rejected, a rejection email is sent to the user. The user can submit new requests. However, the rejected request can't be edited.

Manage app requests from Creator Studio

View and manage new app requests in App Engine Management Center (AEMC) that users with the Creator Studio Restricted role (sn_creator_studio_restricted_user) submit for apps they want to create.

Before you begin

Role required: sn_app_eng_notify.app_engine_admin

Procedure

1. Navigate to **All** > **App Engine** > **Administration** > **App Engine Management Center**.
2. View all app requests on the **Requests** page by selecting the **App** tab.
3. Select an app request to view its details.

4. Either approve or reject the request.
   - If the app is useful to your organization, approve the request.
   - If you see issues with the app request, reject the request. If an app request is rejected, the original requestor must put in a new request for the app with changes for reconsideration.

Manage collaboration requests

As an admin, view and approve or reject collaboration requests in the App Engine Management Center (AEMC) from developers who want assistance building their applications.

Before you begin

To view collaboration requests in AEMC, you must first set up Pipelines and Deployments in your production instance. Otherwise, requests are created on the instance where they originated (that is, the instance with App Engine Studio (AES) installed).

Role required: sn_app_eng_notify.app_engine_admin

About this task

Developers with collaboration permissions can access and develop certain parts of an application. Requests for collaboration may come from App Engine Studio or Creator Studio.

Requests from collaborators may be approved automatically depending on permissions or roles they already have.

- For AES collaboration requests, users who have the AES User role are approved automatically.
- Creator Studio users who have already created an app (or have had one created for them) have delegated development permissions, and are automatically approved to collaborate on other apps.
- Current delegated developers without the Creator Studio User or Creator Studio Restricted User roles are automatically approved to collaborate in Creator Studio.

For more information about managing collaboration and permissions, see Collaborate with other developers.

Procedure

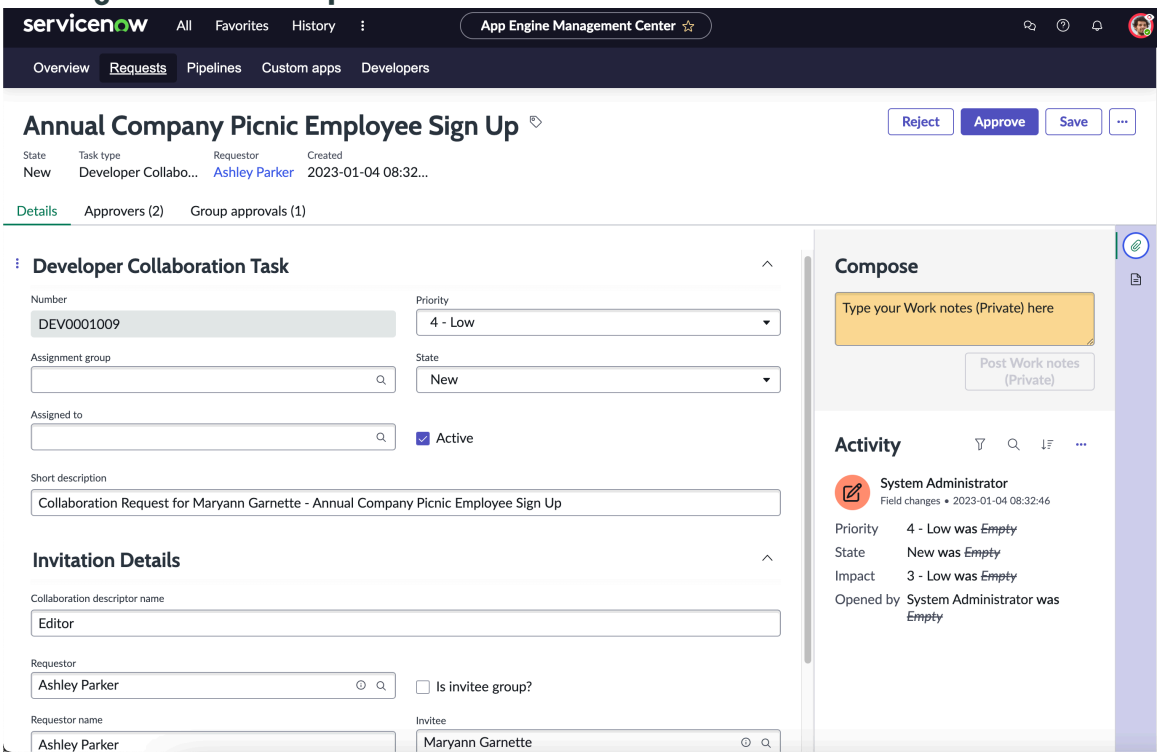
1. Navigate to **All** > **App Engine** > **Administration** > **App Engine Management Center**.

   The AEMC dashboard shows the number of pending collaboration requests.

   Note:
   To view a list of all collaboration requests, either select the number of requests or select the Requests page.

2. Select a collaboration request to review its details.

Pending collaboration request record



3. Return to the **Details** tab and review the request.
4. Approve or reject the request.
   - If the request is approved, the user invited to be a collaborator is added as a developer to the specified app. The collaborator can now work on the parts of the app defined in the Collaboration Descriptor, but cannot concurrently develop other apps in AES. For more information about the AES User role, see Components installed with App Engine Studio.
   - If the request is rejected, the request is closed, and the primary developer is notified by email.

Manage deployment requests

Review and approve or reject deployment requests in the App Engine Studio App Engine Management Center (AEMC) as applications move from development to production.

Before you begin

Role required: sn_app_eng_notify.app_engine_admin

About this task

To view deployment requests in AEMC, pipelines must be configured correctly. For details, see Configure Pipelines and Deployments.

Procedure

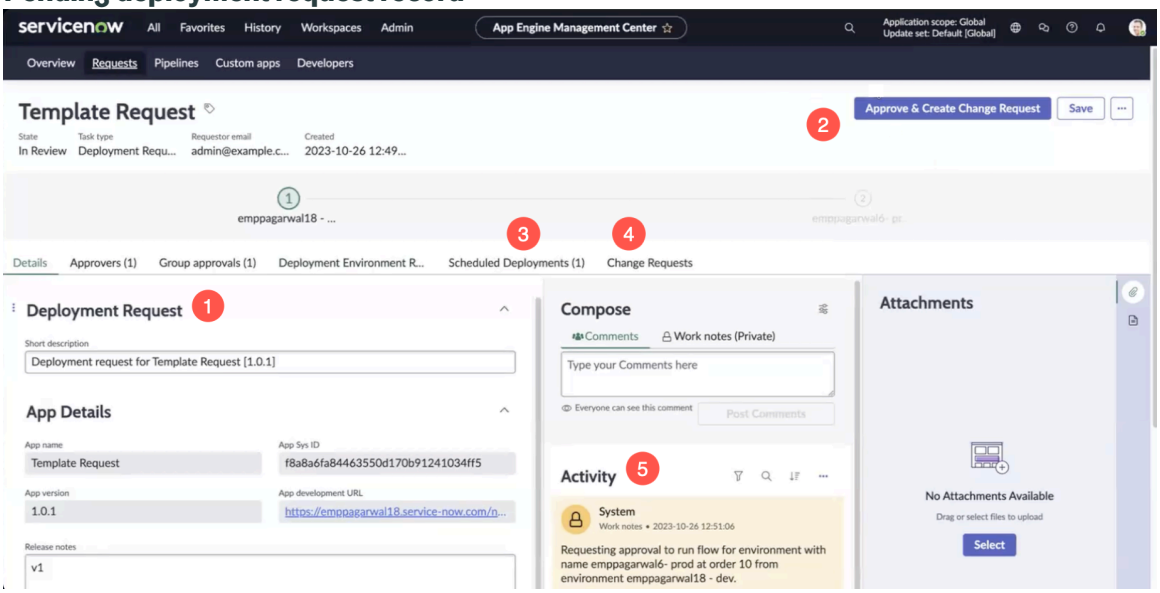
1. Navigate to **All** > **App Engine** > **Administration** > **App Engine Management Center**.

   The AEMC dashboard shows the number of deployment requests.

   Note:
   To view a list of all deployment requests, either select the number of requests or select the Requests page.

2. Select a deployment request to review its details.

Pending deployment request record



3. Review the details of the deployment request, including details about the application, the developer who requested deployment, and the deployment process.

   At the top of each deployment request, you can see the current position of the app in the pipeline.

> **Note:**
> If the request is in a testing environment, you can review the Automated Test Framework (ATF) test and instance scan results in the Activity stream or on the **Deployment Environment Results** tab.

4. Review the notes and application activity for the app manifest and any important details the developer may have added.
5. If a different App Engine Studio (AES) admin must review the deployment request, update the request assignee and select **Save**.
6. Approve or reject the request.
   - To move the application to the next deployment stage, select **Approve & Deploy app**.
   - If you have integrated your existing Change Management processes with AEMC, select **Approve & Create Change Request**. This action creates a change request, which can be seen and monitored on the **Change Requests** tab.
   - If the application is not ready or could potentially cause issues if it was moved on to the next deployment stage, select **Reject**. Add **Work notes** to identify the reasons for the rejection and any changes required to make the application ready for deployment.

> **Note:**
> Unless you are assigned a Change Management role, most of the information on the **Change Request** tab is read-only. You can monitor the change request and see as it changes states. However, if you have Change Management permissions, you may be able to update certain fields on the change request such as the planned start and end date or the state.

Filter and search for requests in AEMC

The App Engine Management Center (AEMC) application provides tools for locating requests. You can perform a global search for request records or filter the list of current requests to locate the ones you want to work on.

Filtering requests

On the Requests, Pipelines, Custom apps, and Developers pages, you can filter the list of requests by several different criteria.

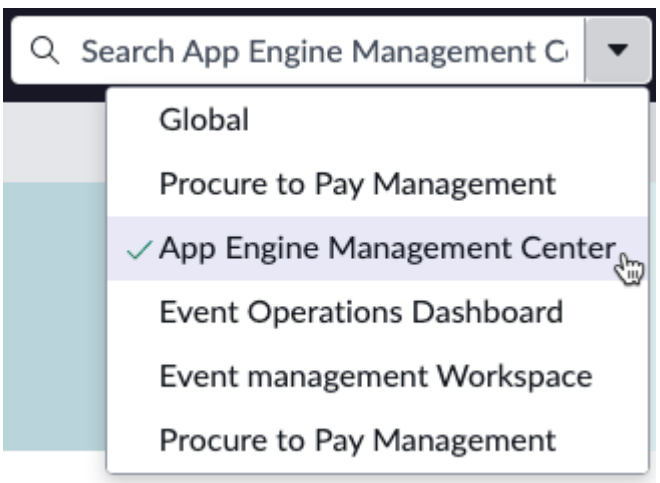
To create a filter, select the filter icon ( ), select **Advanced view**, and add criteria. Build a filter by adding conditions that contain a field, operator, and value(s). You can use an existing filter, save what you create, and limit who can use the filter. Build more complex filters by using the **or** and **and** connectors. Add as many filters as you need, and select **Update**.

When you filter on the Pipelines tab, only instances with requests that match the filtering criteria display information. For example, if your pipeline shows request data in three instances and you filter by an app name used in only one of those instances, only that instance returns results.

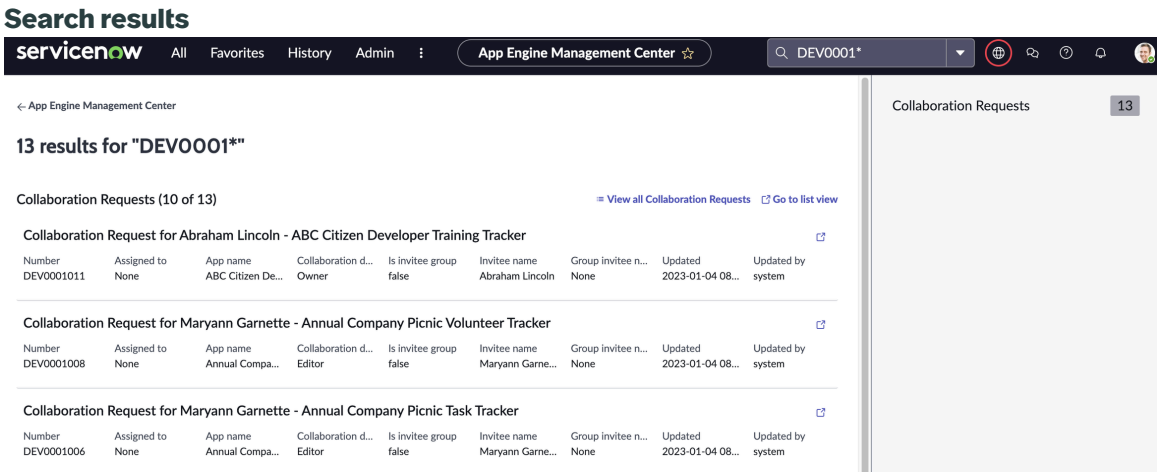
Searching for requests

You can use the **Search** field at the top of any AEMC screen to search for submitted requests. Simply type all or a portion of a request number using an asterisk wild card character, and select **View results**.

Search



For example, you can type DEV0001* and select **View results** (or simply press **Enter**), and a list of all collaboration requests that begin with DEV0001 are returned.

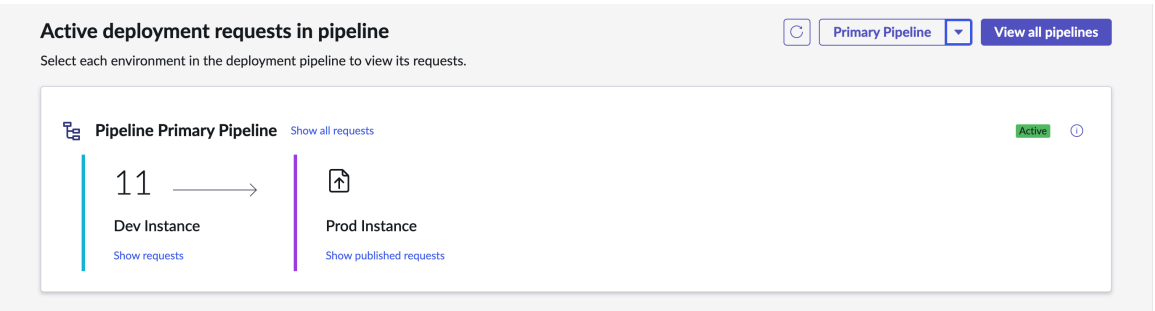


You can then select the record you want to review.

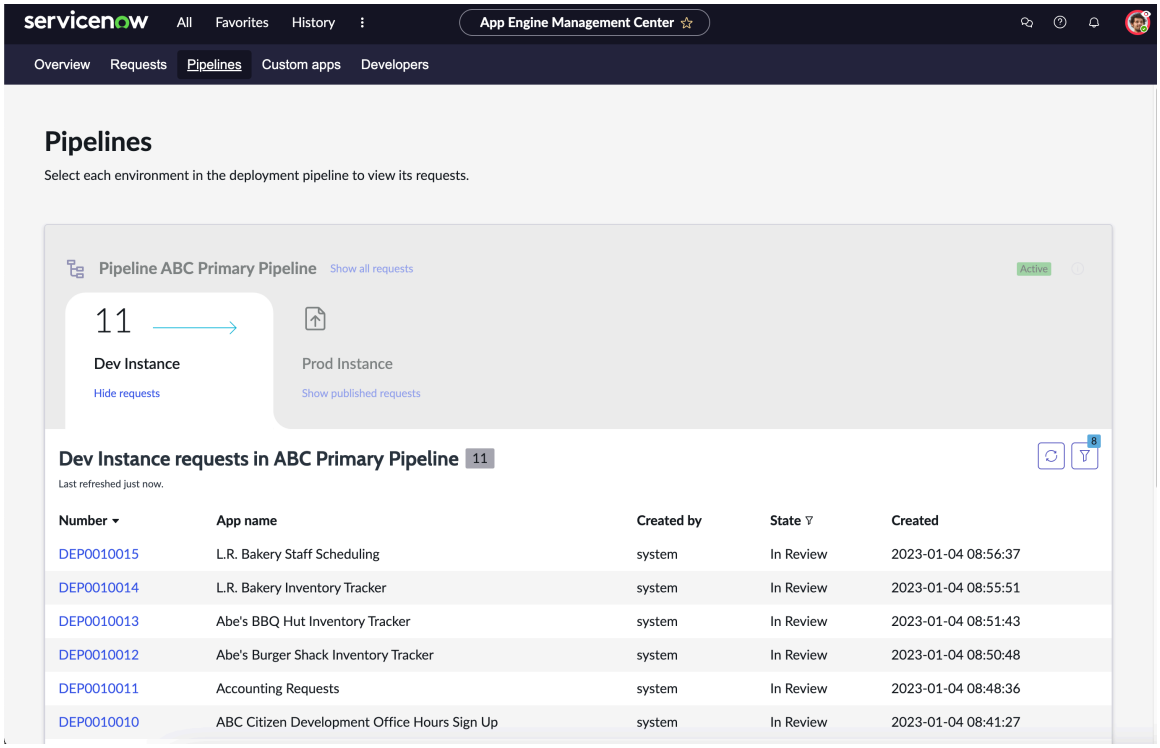
Managing deployments using pipelines in AEMC

Manage applications at all stages of deployment, view multiple pipelines, and approve or reject deployment requests using the App Engine Management Center (AEMC).

The Active deployment requests in the pipeline section of the AEMC Overview page shows the number of apps at each deployment stage within each of your pipelines. You can select **Show all requests**, **Show requests**, or **Show published requests** next to the corresponding sections to see an overview of each group of requests.



For a full picture of all your pipeline deployment requests, access the Pipelines page in AEMC. View all of your pipelines, quickly access each deployment request record, and filter each pipeline section to see only the requests that match your criteria.



Change Management integration

You can integrate an existing Change Management program with your app deployment processes to add oversight into your deployments and have apps deploy according to a scheduled Change window. For more information, see Manage deployment requests.

Schedule app deployments in AEMC

Schedule apps to deploy to production at a future time to load balance your systems using the App Engine Management Center (AEMC).

Before you begin

Role required: admin or app_engine_admin

About this task

Deploying apps through your pipelines requires a certain amount of processing power. Depending on the needs of your organization, you may want to schedule deployments to happen when less processing power is needed elsewhere or when it's generally more convenient.

Procedure

1. Navigate to **All** > **App Engine** > **Administration** > **App Engine Management Center**.
2. On the **Requests** page, select the deployment request for an application that is ready to go to your production environment.
3. Select **Deploy app**.
4. Either deploy your app now or set up a future deployment time.
   - To deploy now, select **Deploy**.
   - To deploy in the future, select **Schedule for later**.
5. **Optional:** If you scheduled the deployment for later, use the date and time picker to select when you want your app to deploy, select **OK**, and select **Deploy**.
   The Activity stream shows the deployment details and the version of the app that will be deployed at the selected date and time. The app is deployed within fifteen minutes of the

selected deployment time. If you want to change this time frame, update how often the deployment workflow checks for a scheduled deployment record by navigating to **All** > **System Definition** > **Scheduled Jobs** > **Scheduled Deployments**.

6. Select **Save**.

Update a deployment request in AEMC

Change the details of a deployment request in App Engine Management Center (AEMC).

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **App Engine** > **Administration** > **App Engine Management Center**.
2. On the **Requests** page, select the deployment request that you want to update.
3. On the **Scheduled Deployments** tab, select the deployment record you want to update.
4. In the **Scheduled Deployment Date** field, select an updated date and time for the app to deploy to production.
5. Select **Save**.

Cancel a scheduled deployment in AEMC

Cancel a scheduled app deployment in App Engine Management Center (AEMC) if you no longer need to deploy the app to production. Canceling the scheduled deployment request cancels the scheduled deployment and also the entire request itself.

Before you begin

Role required: admin

Procedure

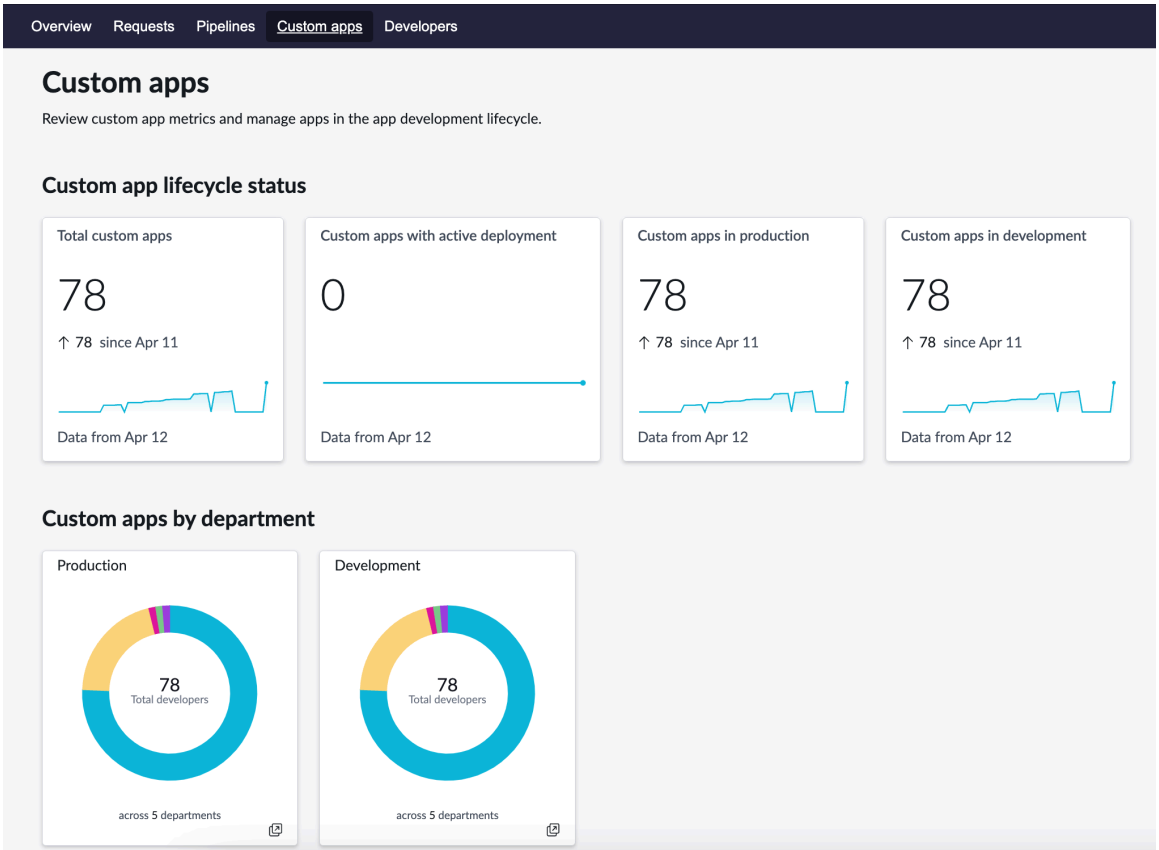
1. Navigate to **All** > **App Engine** > **Administration** > **App Engine Management Center**.
2. On the **Requests** page, select the deployment request that you want to update.
3. On the deployment request, select **Cancel deployment**.

Managing custom apps using AEMC

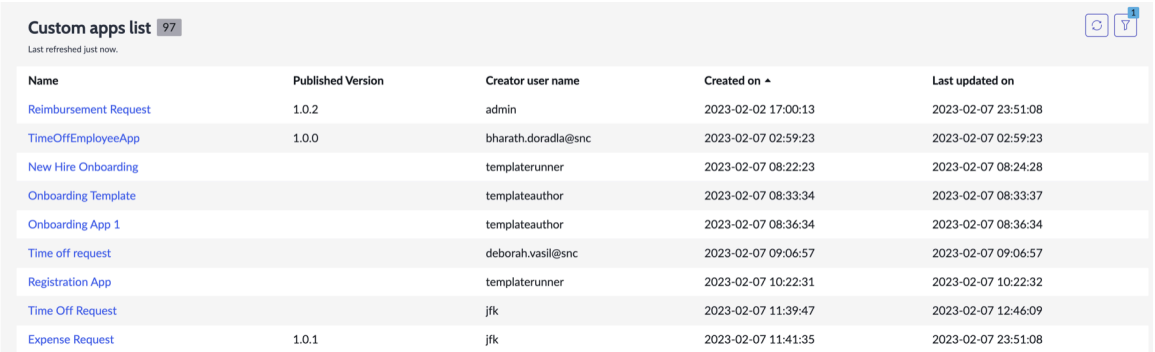
Review custom app metrics and manage apps through the development life cycle using the App Engine Management Center (AEMC).

The App Engine adoption metrics section of the AEMC Overview page shows how many apps you have in development and production environments. Access more information by selecting the large number on each Custom apps card.

The Custom apps page in AEMC shows more detailed metrics and a full list of your applications. In each tile in the Custom app lifecycle status section, you can see trending data for the last 90 days. You can show all apps in the Custom apps list or limit the list to only certain types of applications using the filter. Select a trend chart to filter the list to those criteria.



The Custom app page also lists all the applications individually, with a single record for each application. If the app has been published to production, the **Published Version** column shows the version number. If the **Published Version** column is blank, the app exists in development but has never been published to production.

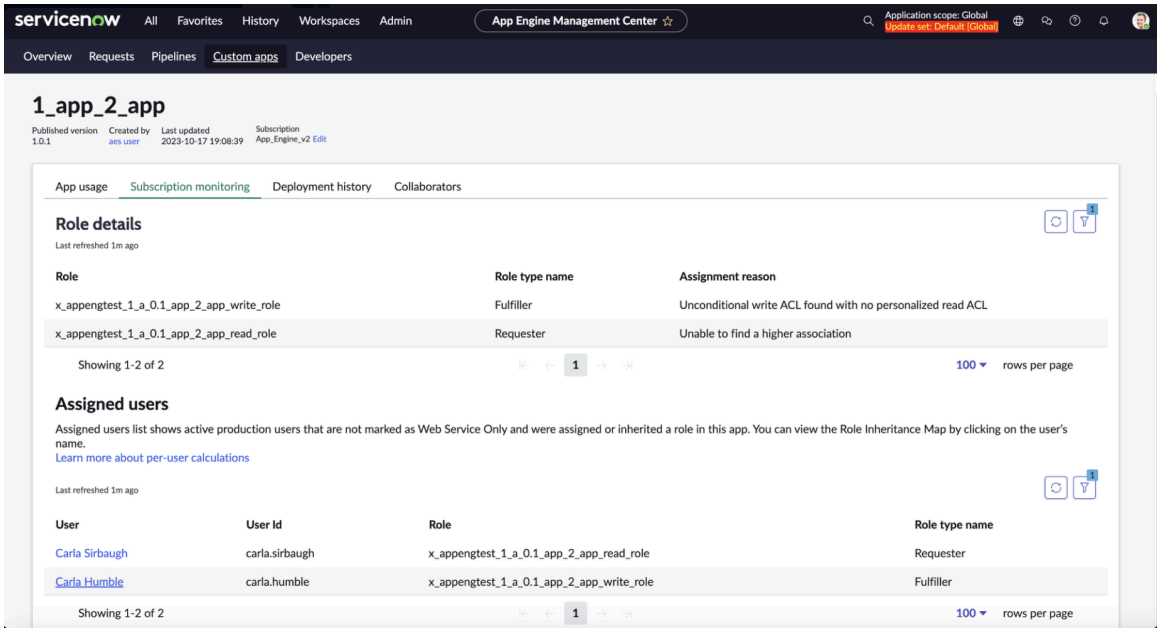


Custom apps list 97

Last refreshed just now.

Name	Published Version	Creator user name	Created on ▲	Last updated on
Reimbursement Request	1.0.2	admin	2023-02-02 17:00:13	2023-02-07 23:51:08
TimeOffEmployeeApp	1.0.0	bharath.doradla@snc	2023-02-07 02:59:23	2023-02-07 02:59:23
New Hire Onboarding		templaterunner	2023-02-07 08:22:23	2023-02-07 08:24:28
Onboarding Template		templateauthor	2023-02-07 08:33:34	2023-02-07 08:33:37
Onboarding App 1		templateauthor	2023-02-07 08:36:34	2023-02-07 08:36:34
Time off request		deborah.vasil@snc	2023-02-07 09:06:57	2023-02-07 09:06:57
Registration App		templaterunner	2023-02-07 10:22:31	2023-02-07 10:22:32
Time Off Request		jfk	2023-02-07 11:39:47	2023-02-07 12:46:09
Expense Request	1.0.1	jfk	2023-02-07 11:41:35	2023-02-07 23:51:08

View the app's usage data, subscription monitoring details, deployment history, and collaborators, if any, by selecting an application name from the list.



Select **Open in App Engine Studio** to see the contents of the app and access more information. The App Engine Studio application opens in a new browser window.

App usage

See a high-level view of the usage data for apps on production instances by selecting the **App usage** tab. The data includes the user count, insert count, and update count aggregated for the current month.

- User count: The number of users who have accessed the application.
- Insert count: The number of new records on tables in the application.
- Update count: The number of records in the application that have been updated.

Knowing usage data for your applications has several benefits:

- Discover whether your app is being used. Are people using the app? Are they creating records? If not, consider updating or deleting the app.
- Investigate and fix problems with the app. Knowing when or how data was changed can help identify where something went wrong.
- Determine whether you must load balance the processing power of your instance by seeing when there might be consistently higher traffic in your app.

In the User Experience Analytics dashboards section, see detailed data such as active users, sessions, and page views by selecting an experience. For more information about how to understand and use this data, see User Experience Analytics.

Subscription monitoring

Understand the roles used in your custom application and their implications for licensing and subscriptions. The data in this section includes details about each role and the users assigned to each role.

Knowing subscription and licensing data information for each of your applications has several benefits:

- Discover what roles are active in your custom application.
- Discover which users are assigned each role and the permission levels they have.
- See the cost implications of each license type and where it's used in your apps.

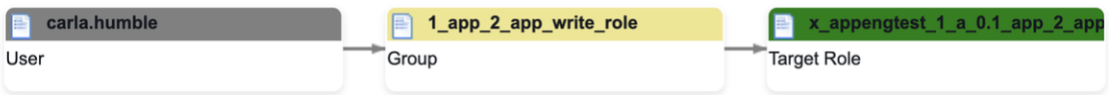
System administrators can see which subscription the app is mapped to and have an option to change the subscription. Several factors affect what you see in this section.

- If the app is mapped to an App Engine subscription, the **Subscription Monitoring** tab is visible.
- If the app is mapped to a different subscription such as ITSM, the **Subscription Monitoring** tab is not visible.
- If the app is mapped to App Engine, is switched to another subscription, and then is changed back to App Engine, the **Subscription Monitoring** tab is visible again.

App Engine admins do not have permissions to change subscriptions. However, if an app is in the production environment, and there is a **Subscription Monitoring** tab, the App Engine admin can infer that the app is mapped to an App Engine subscription. If the **Subscription Monitoring** tab is not there, that app isn't mapped to App Engine.

You can select a user's name in the **Assigned Users** section to see a role inheritance map. This map shows how this user inherited the role, whether from a parent role, a group assignment, or another inheritance. This information can help you take action if someone has a role you believe they shouldn't, and you want to change it.

Role inheritance map



App deployments

View the list of deployments your app has gone through on the **Deployment history** tab.

Knowing deployment history information has several benefits:

- If an app is deployed to production and there’s a defect, you can quickly see when the app was deployed and track down the problem.
- If an app is deployed multiple times, you can quickly track what's changed over multiple deployments.
- If an app is deployed to production, you can quickly see who approved the app deployment.

App collaborators

View the collaborators for the app by selecting the **Collaborators** tab. All collaborators are stored on the Development instance.

If you must take quick action in the application to fix an issue or defect, you can quickly see who has permissions to make changes in an app and what those permissions are.

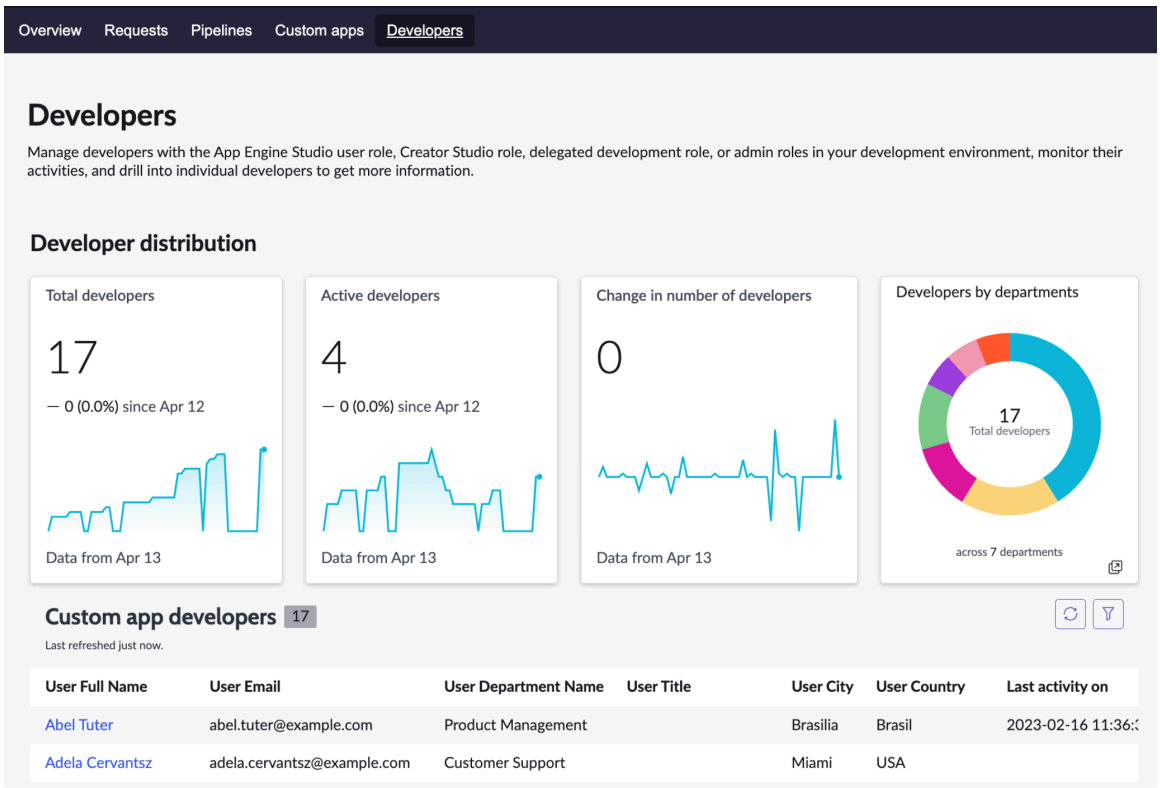
Managing developers using AEMC

View details about developers working on apps in App Engine Studio and Creator Studio using App Engine Management Center (AEMC).

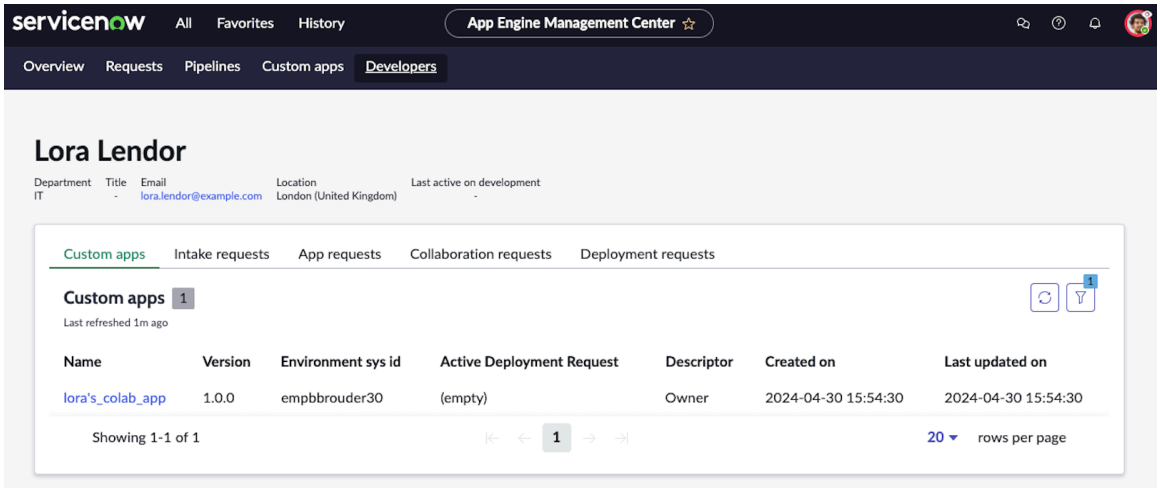
The App Engine adoption metrics section of the AEMC Overview page shows how many developers are working in App Engine Studio (AES) and their departments. Access more information by selecting the **Developers** page.

The Developers page in AEMC shows all developers with the AES Users role, AES User Limited role, Creator Studio Users role, Creator Studio Restricted Users role, or a delegated development role in your development environment. On this page, you can monitor their activities and select individual developers to get more information.

Note:
To remove a developer from the developer tab, you can either revoke all their permissions in Creator Studio, App Engine Studio, Delegated Development, and Admin roles, or deactivate their account on the development instance.



Select a developer record to see more information about the custom apps they have created and their intake requests, app requests, collaboration requests, and deployment requests. You can filter the content on each tab to find specific results, or select an app from this page to see its usage, deployment history, and collaborators. Select any of the displayed records to view the history or to approve, reject, or delete the requests.



App Engine Studio reference

Reference topics provide additional information about the lists and forms that you use to configure and administer App Engine Studio (AES).

AES glossary

Learn about terms and concepts that are unique to App Engine Studio (AES).

Application template

Provides predefined data, experience, logic and automation, and security to support a certain use case. For example, the Travel Request template provides application content for submitting and approving employee travel requests.

For more information on the available templates, see Available templates.

Logic and automation

Automate all the work in your application by adding logic and automation. For example, you can build a flow that sends a notification to the admin when someone makes a request.

Data

Information that is stored in your application. For example, employee phone numbers or office locations. You configure application data using tables.

Deployment request

Ticket to track the review of submitted applications. From the deployment request form, a reviewer can deploy the application to different environments, accept or reject an application, and send feedback to a developer.

Environment

Instance that you use for the developing, testing, or launching an application. To set up App Engine Studio, you must define environments in each of the instances that you're using as environments.

Experience

Graphical interface that your users interact with. For example, you can create a portal where users find information, submit requests, or complete business tasks. For more information on the available application experiences, see Add an application experience.

Pipeline

Configuration for deploying an application to different environments.

Security

Roles and access controls to limit who can use your application.

AES properties

System properties control system behavior. The properties in this section are specific to the App Engine Studio (AES) application. You can access system properties for AES by navigating to **All** > **App Engine Studio** > **Configuration** > **Properties**.

sn_app_eng_studio.show_servicenow_studio_banner

Controls whether the try ServiceNow Studio banner appears on the App Engine Studio home page. Set to true to show the banner or set to false to hide the banner.

- Type: true | false
- Default value: true

sn_app_eng_studio.aes_admin_contact

Defines the contact email for the App Engine Studio administrator. The system sends deployment request notifications to each email address listed in this property. For more information, see Add users to the App Engine Admin group.

- Type: string
- Default value: none

sn_app_eng_studio.illustration_supported_content_types

Defines the supported content types for Taxonomy Category illustrations. This property is read-only.

- Type: string
- Default value: image/svg+xml

sn_app_intake.instance_can_provision_users

Determines if the current instance can support the provisioning of users. The users on this instance will be provisioned to another instance. For more information on provisioning users, see Configure Application Intake.

- Type: true | false
- Default value: false

Deployment Request form in the Pipelines and Deployments app

The Deployment Request form is accessed from within the App Engine Studio (AES) Pipelines and Deployments app.

To view the Deployment Request form, navigate to **All** > **App Engine** > **Pipelines and Deployments** > **Deployment Requests** and open a record.

Deployment Request form

Field	Description
Number	Number to track the deployment request.
Assignment group	Group to which the deployment request is assigned.
State	State of the deployment request. All states are automatically updated by the pipeline flows:

Deployment Request form (continued)

Field	Description
	New The submitted request is new. **In Review** The submitted request is ready for review. **Closed - Published** The submitted request has been deployed to the production instance. **Closed - Rejected** The submitted request is not ready to publish. This state is enabled when you select **Reject** on the associated approval record at any time during the deployment process. The system then prompts you to include the reason why the application is rejected. These comments are then sent via email to the address included in the **Requestor Email** field on the **Requestor Details** related list. **Closed - Failed** This state is enabled when any of the pipeline flows or sub-flows fail for any reason. **Canceled** The submitted request has been canceled.
Assigned to	The user in the assignment group to whom this request is assigned.
Short description	A brief description for the request.
Release notes	Text that appears on the ServiceNow Store for this application after it is published.
App Details	
App name	Name of the submitted application.
App version	Version of the submitted application.
App Sys ID	System ID of the submitted application.
App Development URL	Link to app in App Engine Studio.
Requestor Details	
Requestor First/Last Name	First and last name of the user who requested to the app deployment.
Requestor User ID	Identifier of the user who requested to the app deployment.
Requestor Email	Email address of the user who requested to the app deployment.
Deployment Details	
Originating Environment	The instance of the originating environment.
Pipeline	Pipeline being used for deployment.

Deployment Request form (continued)

Field	Description
Current Environment	The target environment for the app.
Notes	
Additional comments (Customer visible)	Any additional comments to be shared with the customer, including the app manifest, which lists the number of each type of file in the app, and any ATF or instance scan results are shown.
Work notes	Internal notes.

When you have saved the form, additional tabs for approval-related information display. The AES admin can use these tabs to approve the request.

Supported features and metadata in custom templates

Custom templates in App Engine Studio (AES) support most features and metadata, including, but not limited to tables, forms, roles, ACLs, record producers, simple workspaces, flows, and actions.

Custom templates can be created in two ways, either from scratch or using an existing scoped application as a starting point.

When you create a custom template, the pre-scan checks to make sure that everything in the application is supported. There are three possible outcomes.

- All items are supported: The custom template is created. All the objects are available in the custom template and in apps created from the custom template.
- Some items aren't supported: You can continue to create the template but a few items are not available in the custom template. There’s no impact on the custom template or the apps created from the template.
- Some items are denied: You encounter an error message and cannot continue further until the errors are fixed. The author must change the app, if it was created from an existing app or the template, if it was created from scratch.

Note:
Creating an AES template using an application that contains a workspace is not supported.

For more information on the template creation process, see Build a custom template.

Supported metadata

Label	Name
Access Control	sys_security_acl
Access Roles	sys_security_acl_role
Action Assignment	sys_declarative_action_assignment
Action Category	sys_hub_category
Action Definition	sys_declarative_action_definition
Action Exclusion	sys_workspace_declarative_action_exclusion

Label	Name
Action Inputs	sys_hub_action_input
Action Instance	sys_hub_action_instance
Action item	sys_sg_write_back_action_item
Action Model Field	sys_declarative_action_model_field
Action Outputs	sys_hub_action_output
Action parameters mapping	sys_sg_action_param_map
Action Payload Definition	sys_declarative_action_payload_definition
Action Payload Field	sys_declarative_action_payload_field
Action Payload Mapping	sys_declarative_action_payload_mapping
Action Type	sys_hub_action_type_definition
Activity Definition	sys_pd_activity_definition
Activity Stream Screen	sys_sg_activity_stream_screen
Analytics preview	sys_sg_chart
Analytics section	sys_sg_chart_section
Application Menu	sys_app_application
Application Menu	sys_ui_application
Application Search Sources	m2m_search_context_config_search_source
Audience	sys_ux_applicability
Basic Auth Configuration	sys_auth_profile_basic
Builder Template	sys_uib_template
Business Rule	sys_script
Card	sys_sg_view_config
Card element	sys_sg_view_config_element
Card element attribute	sys_sg_view_config_element_attribute
Card template	sys_sg_view_template
Card template element	sys_sg_view_template_slot
Card template element attribute	sys_sg_view_template_slot_attribute
Catalog Client Scripts	catalog_script_client
Catalog Item	sc_cat_item
Catalog Item Available for	sc_cat_item_user_criteria_mtom
Catalog Item Category	sc_cat_item_category
Catalog Item Not Available for	sc_cat_item_user_criteria_no_mtom
Catalog Items Catalog	sc_cat_item_catalog
Catalog UI Policy	catalog_ui_policy
Catalog UI Policy Action	catalog_ui_policy_action

Label	Name
Catalog Variable Set	io_set_item
Category Property	sys_properties_category_m2m
Chart Colors	sys_report_chart_color
Chart screen	sys_sg_chart_screen
Choice	sys_choice
Choice Set	sys_choice_set
Client Script	sys_script_client
Color Definition	sys_report_color
Complex Object	sys_complex_object
Composite Data Broker	sys_ux_data_broker_composite
Contained Role	sys_user_role_contains
Cross scope privilege	sys_scope_privilege
Data Broker Scriptlet	sys_ux_data_broker_scriptlet
Data Broker Server Script	sys_ux_data_broker_transform
Data Item	sys_sg_data_item
Data Parameter	sys_sg_item_parameter
Data Policy	sys_data_policy2
Data Policy Rule	sys_data_policy_rule
Decision	sys_decision_question
Decision Condition	sn_decision_table_decision_condition
Decision Input	sys_decision_input
Decision Table	sys_decision
Decision Table Multiple Result	sys_decision_multi_result
Details screen	sys_sg_details_screen
Dictionary Entry	sys_dictionary
Dictionary Entry Override	sys_dictionary_override
Dynamic Filter Options	sys_filter_option_dynamic
Email Address Filter	sys_email_address_filter
Email Address Filter Domain	sys_email_address_filt_domain
Email Client Configuration	sys_email_client_configuration
Email Client From Address	sys_email_client_from_address
Email Client Template	sys_email_client_template
Email Filter	sys_email_filter
Email Layout	sys_email_layout
Email Reply Separator	sys_email_reply_separator

Label	Name
Email Script	sys_script_email
Email Template	sysevent_email_template
Embedded Help Role Priority	sys_embedded_help_role
Empty State	sys_sg_empty_state
EVAM Data Filter	sys_ux_composite_data_filter
EVAM Data Source Filter	sys_ux_composite_datasource_filter
EVAM Datasource	sys_ux_composite_datasource
EVAM Datasource M2M	sys_ux_composite_data_m2m_datasource
EVAM Definition	sys_ux_composite_data
EVAM View Config	sys_ux_composite_data_template_predicate
EVAM View Config Action Assignment M2M	sys_ux_predicate_m2m_action_assignment
EVAM View Config Bundle	sys_ux_composite_data_template_predicate_bundle
EVAM View Config Bundle M2M	sys_ux_composite_data_m2m_predicate_bundle
Event Registration	sysevent_register
Export Definition	sys_export_definition
Export Set	sys_export_set
Export Target	sys_export_target
Extended Flow Logic Inputs	sys_hub_flow_logic_ext_input
Extended Step Input Variable	sys_hub_step_ext_input
Extended Step Output Variable	sys_hub_step_ext_output
Extension Instance	sys_extension_instance
Extension Point	sys_extension_point
Facet	sys_search_facet
Field Label	sys_documentation
Field Map	sys_impex_entry
Filter	sys_filter
Filter Action	sys_email_filter_action
Flow	sys_hub_flow
Flow catalog variable	sys_flow_cat_variable
Flow Inputs	sys_hub_flow_input
Flow Logic Instance	sys_hub_flow_logic
Flow Snapshot	sys_hub_flow_snapshot
Flow Stages	sys_hub_flow_stage
Flow Variables	sys_hub_flow_variable
Form	sys_ui_form

Label	Name
Form Header	sys_aw_form_header
Form Section	sys_ui_section
Function	sys_sg_button
Function instance	sys_sg_button_instance
GraphQL Data Broker	sys_ux_data_broker_graphql
Highlighted Value	sys_highlighted_value
Highlighted Value Condition	sys_highlighted_value_condition
HTTP Headers	sys_rest_message_headers
HTTP Headers	sys_rest_message_fn_headers
HTTP Method	sys_rest_message_fn
HTTP Query Parameters	sys_rest_message_fn_param_defs
Icon	sys_sg_icon
Icon section destination launcher	sys_sg_navigation_section_destination_applet_launcher
Images	db_image
Import Export Map	sys_impex_map
Inbound Email Actions	sysevent_in_email_action
Indexes	sys_index
Launcher header title	sys_sg_applet_launcher_header
Launcher screen	sys_sg_applet_launcher
Launcher section mapping	sys_sg_applet_launcher_m2m_section
Launcher tab	sys_sg_applet_launcher_tab
Layout Applicability	sys_ux_applicability_m2m_layout
Legacy card	sys_sg_item_view
Legacy icon section	sys_sg_icon_section
List	sys_ui_list
List Applicability	sys_ux_applicability_m2m_list
List Control	sys_ui_list_control
List item config	sys_sg_master_item
List screen	sys_sg_list_screen
List stream	sys_sg_item_stream
M2m Action Assig Ux Action Config	sys_ux_m2m_action_assignment_action_config
M2m Workspace Header Ux Header Config	sys_ux_m2m_workspace_header_ux_header_config
Many to Many Definition	sys_m2m
Message	sys_ui_message
Mobile UI Rule	sys_sg_ui_rule

Label	Name
Mobile UI Rule Action	sys_sg_ui_rule_action
Module	sys_ui_module
Module	sys_app_module
Navigation Tab Map	sys_sg_navigation_tab_map
New Record Menu Item	sys_aw_new_menu_item
Notification	sys_notification
Notification	sysevent_email_action
Notification Category	sys_notification_category
Notification Common Content	sys_notification_common_content
Notification Route	notification_route
Notification Workspace Content	sys_notification_workspace_content
Number	sys_number
Playbook Activity Override	sys_playbook_activity_renderer
Playbook Activity UIs	sys_playbook_experience_activity_ui
Playbook Experience Action Assignment Map	sys_playbook_experience_action_assignment_map
Playbook Experience Record Generator	sys_playbook_experience_record_generator
Playbook Experiences	sys_playbook_experience
Process Definition	sys_pd_process_definition
Process Definition Swim Lane Images	sys_pd_swim_lane_image
Process Flow Trigger Plan	sys_flow_trigger_plan
Public Pages	sys_public
Push Notification	sys_sg_push_notification
Push Notification Installation	sys_push_notif_app_install
Push Notification Message	sys_push_notif_msg
Push Notification Message Content	sys_push_notif_msg_content
Question Choice	question_choice
Quick actions menu	sys_sg_alp_quick_action_map
Recipient Qualifier	sys_recipient_qualifier
Record Producer	sc_cat_item_producer
Record screen	sys_sg_form_screen
Record screen segment	sys_sg_form_segment
Record section	sys_sg_item_section
Redirection destination fields	sys_sg_redirect_dest_field
Related List	sys_ui_related_list
Relationship	sys_relationship

Label	Name
Relationship Data Item	sys_sg_relationship_data_item
Report	sys_report
Report Users and Groups	sys_report_users_groups
REST Data Broker	sys_ux_data_broker_rest
REST Message	sys_rest_message
Result	sys_decision_multi_result_element
Role	sys_user_role
Route Configuration	sys_ux_dynamic_route_config
Route Mapping	sys_ux_dynamic_route_mapping
Scheduled Script Execution	sysauto_script
Screen Applicability	sys_ux_applicability_m2m_screen
Screen fields	sys_sg_screen_field
Screen parameters mapping	sys_sg_screen_param_map
Screen segment	sys_sg_item_stream_segment
Screen tab	sys_sg_applet_tab
Screen UI Policy	sys_sg_ui_policy
Script Action	sysevent_script_action
Script Include	sys_script_include
Scripted REST API	sys_ws_definition
Scripted REST Query Parameter	sys_ws_query_parameter
Scripted REST Query Parameter Association	sys_ws_query_parameter_map
Scripted REST Resource	sys_ws_operation
Search Application Configuration	sys_search_context_config
Search config	sys_sg_global_search
Search M2M item config	sys_sg_global_search_m2m_master_item
Search Source	sys_search_source
Secondary Values	sys_aw_form_header_secondary_values
Service Fulfillment Approval Step	sc_service_fulfillment_approval_step
Service Fulfillment Stage	sc_service_fulfillment_stage
Service Fulfillment Step	sc_service_fulfillment_step
Service Fulfillment Task Step	sc_service_fulfillment_task_step
Service Offering SLA	service_offering_sla
SLA Definition	contract_sla
SLA Timer Configuration	sla_timer_config
SLA timer configuration mapping	sla_timer_config_mapping

Label	Name
Step Instance	sys_hub_step_instance
Stream M2M item config	sys_sg_item_stream_m2m_master_item
Stream M2M segment	sys_sg_item_stream_m2m_segment
Style	sys_ui_style
Subflow Instance	sys_hub_sub_flow_instance
Subflow Instance Inputs	sys_hub_sub_flow_instance_inputs
Suggestion Reader and Suggestion Reader Group Relations	m2m_sys_suggestion_reader_sys_suggestion_reader_group
Suggestion Reader Group	sys_suggestion_reader_group
Sys Ux M2m Action Layout Item	sys_ux_m2m_action_layout_item
Sys Ux M2m Highlighted Value Config	sys_ux_m2m_highlighted_value_config
Sys Ux M2m Workspace View Rule Ux View Rule Config	sys_ux_m2m_workspace_view_rule_ux_view_rule_config
System Property	sys_properties
System Property Category	sys_properties_category
Table	sys_db_object
Task Relationship Type	task_rel_type
Task Relationships Allowed	task_rel_allowed
Template	sys_template
Transform Script	sys_transform_script
Trigger Definition	sys_pd_trigger_definition
Trigger Execution	sys_flow_trigger_auto_script
Trigger Instance	sys_hub_trigger_instance
UI Action	sys_ui_action
UI Action Layout	sys_aw_form_uiaction_layout
UI Action Role	sys_ui_action_role
UI Formatter	sys_ui_formatter
UI Page	sys_ui_page
UI parameter	sys_sg_ui_parameter
UI Policy	sys_ui_policy
UI Policy Action	sys_ui_policy_action
UI Policy Rule	sys_sg_ui_policy_rule
UI Script	sys_ui_script
UI View	sys_ui_view
UIB Screen Test Values	sys_uib_screen_test_values
UX Actions Configuration	sys_ux_action_config

Label	Name
UX Add-on Event Mapping	sys_ux_addon_event_mapping
UX App Configuration	sys_ux_app_config
UX App Route	sys_ux_app_route
UX App Theme	m2m_app_config_theme
UX Application	sys_ux_page_registry
UX Application Category M2M	sys_ux_registry_m2m_category
UX Asset	sys_ux_lib_asset
UX Client Script	sys_ux_client_script
UX Client Script Include	sys_ux_client_script_include
UX Custom Content Extension	sys_ux_custom_content_root_elem
UX Dashboard Route Permission	sys_ux_route_permission
UX Event	sys_ux_event
UX Form Actions	sys_ux_form_action
UX Form Actions Layout	sys_ux_form_action_layout
UX Form Actions Layout Group	sys_ux_form_action_layout_group
UX Form Actions Layout Item	sys_ux_form_action_layout_item
UX Header Configuration	sys_ux_header_config
UX Highlighted Value Configuration	sys_ux_highlighted_value_config
UX List	sys_ux_list
UX List Category	sys_ux_list_category
UX List Menu Configuration	sys_ux_list_menu_config
UX Macroponent Definition	sys_ux_macroponent
UX Page Element Permissions	sys_ux_page_element_m2m_role
UX Page Property	sys_ux_page_property
UX Ribbon Configuration	sys_ux_ribbon_config
UX Screen	sys_ux_screen
UX Screen Collection	sys_ux_screen_type
UX Screen Condition	sys_ux_screen_condition
UX Theme	sys_ux_theme
UX Theme Asset	sys_ux_theme_asset
UX Theme Assets	sys_ux_theme_m2m_asset
UX View Rules Configuration	sys_ux_view_rules_configuration
Variable	item_option_new
Variable Set	item_option_new_set
Variable Substitutions	sys_rest_message_fn_parameters

Label	Name
View Rule	sysrule_view_workspace
View Rule	sysrule_view
View Template	sys_ux_composite_data_template
Workspace	sys_aw_master_config
Workspace Global Search Configuration	sys_aw_global_search_config
Workspace List	sys_aw_list
Workspace List Category	sys_aw_list_category
Workspace Module	sys_aw_module

Skipped features and metadata

The template creation process only skips items that should not impact the template or apps created from the template.

Label	Name
Activity Designer	wf_element_activity
Activity Variables	wf_activity_variable
Content Link	content_link
Content Themes	content_theme
Content Type	content_type
Database View	sys_db_view
Detail Pages	content_type_detail
Detailed Content	content_block_detail
Disallows "create from scratch" templates to be re-templatized if there are running executions	
Dynamic Content	content_block_programmatic
Flash Movie	content_block_flash
Flow Template Mapping	sys_app_flow_template_mapping
IFrames	content_block_iframe
Input Detail	sys_app_template_input_detail
Lists of Content	content_block_lists
Login Rule	content_page_rule
Menu Items	menu_item
Menu Sections	menu_section
Meta Tags	content_page_meta
Navigation Menu	content_block_menu
Pages	content_page

Label	Name
Payload	sys_app_scan_payload
Payload Loader Rule	sys_app_payload_loader_rule
Payload Rule	sys_app_payload_unloader_rule
Processor	sys_processor
Site	content_site
Spoke Configuration	sys_app_template_spoke_configuration
Stage Default	wf_stage_default
Static Content	content_block_static
Style Sheet	content_theme_css
Subtemplate	sys_app_subtemplate
Template	sys_app_template
Template Dependency	sys_app_template_dependency
Template Dependency Definition	sys_app_template_dependency_definition
Template Details	sys_app_template_detail
Template Function Type	sys_app_template_function_type
Template Input Variable	sys_app_template_input_var
Template Manifest	sys_app_template_manifest
Template Output Variable	sys_app_template_output_var
Template Page	sys_app_template_page
Template Page Configuration	sys_app_template_page_configuration
Template Wizard	sys_app_template_wizard
Variable	sys_app_scan_variable
Variable Type	sys_app_scan_variable_type
View Field	sys_db_view_table_field
View Table	sys_db_view_table
Workflow	wf_workflow
Workflow Activity Definition	wf_activity_definition
Workflow Instance	wf_workflow_instance
Workflow SC Variable	wf_variable
Workflow Schedule	wf_workflow_schedule
Workflow UI Policy	wf_ui_policy

ServiceNow CLI

The ServiceNow CLI is a command-line interface that lets you perform instance operations from your local system. You can extend the CLI to include new commands that meet your application's needs.

Benefits

The ServiceNow CLI lets you:

- Perform basic CRUD operations on records in your instance.
- Develop custom components and deploy them to your instance to personalize a UI.
- Create custom commands that enable you to manage custom applications from the command line.
- Use the ServiceNow CLI in scripts to simplify setup tasks and operational activities.

Architecture

Commands are stored in a table on the instance you are connected to. When the ServiceNow CLI connects to the instance, it receives all the available commands supported by that instance.

Commands map to a REST endpoint that executes asynchronously. For more information, see Create a custom command in ServiceNow CLI.

Activating ServiceNow CLI

Install ServiceNow CLI by requesting it from the ServiceNow Store. Visit the ServiceNow Store website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Configuration file

The ServiceNow CLI stores profile information in a `config.json` file which, by default, is stored in your home directory at the following path:

- Linux and Mac: `~/.snc/config.json`
- Windows: `%USERPROFILE%\.snc\config.json`

The CLI uses this file to determine what information to use to connect to an instance, and what settings to use to generate output. By default, the ServiceNow CLI uses the settings found in the default profile to connect to an instance. To use alternate settings, you can create and reference additional named profiles. For more information, see Configuring and managing your ServiceNow CLI connection profiles.

The following example shows a#configuration#file with a default profile and a named profile. Each profile can use different credentials and specify different hosts and output formats.

```
{
   "profiles":{
      "default":{
         "host":"https://myinstance.service-now.com",
         "loginmethod":"basic",
         "username":"admin",
         "output":"json",
         "hostversion":"Paris",
         "appversion":"1.0"
         },
      "user1":{
         "host":"https://otherinstance.service-now.com",
         "loginmethod":"basic",
         "username":"user1",
         "output":"yaml",
         "hostversion":"Paris",
         "appversion":"1.0"
         }
      }
}
```

Note:
Sensitive credential information is only stored in the OS keychain, not in the configuration file.

Command structure

ServiceNow CLI commands follow this structure:

1. The base call to the `snc` program.
2. The top-level command group followed by any child command groups.
3. The command that specifies which operation to perform.
4. General CLI arguments required by the operation. You can specify arguments in any order.

```
$ snc <command-group> <command> [arguments]
```

Arguments can take various types of input values, such as numbers, strings, and JSON objects. The types supported depend on the command you specify.

Argument values

Many argument values in the ServiceNow CLI are simple string or numeric values, such as the table and table name in the following example.

```
$ snc record create --table incident --data "{short_description:
 'New Incident'}"
```

You can surround strings that do not contain any space characters with quotation marks or not. However, you must use quotation marks around strings that include one or more space characters.

Output formats

The ServiceNow CLI supports four output formats:

- `json`: The output is formatted as JSON. This is the default.

```
{
   "default": {
      "appversion": "1.0.8",
      "host": "https://myinstance.service-now.com",
      "hostversion": "Paris",
      "loginmethod": "basic",
      "output": "json",
      "username": "admin"
   },
   "user1": {
      "appversion": "1.0.8",
      "host": "https://otherinstance.service-now.com",
      "hostversion": "Paris",
      "loginmethod": "basic",
      "output": "yaml",
      "username": "admin"
   }
}
```

- `yaml`: The output is formatted as YAML. Use YAML to handle the output with services and tools that emit or consume YAML-formatted strings.

```
default:
   appversion: 1.0.8
   host: https://myinstance.service-now.com
   hostversion: Paris
   loginmethod: basic
   output: json
   username: admin
user1:
   appversion: 1.0.8
   host: https://otherinstance.service-now.com
   hostversion: Paris
   loginmethod: basic
   output: yaml
   username: admin
```

- `text`: The output is formatted as multiple lines of tab-separated string values. Use this output with traditional UNIX text tools such as grep, sed, and awk, and the text processing performed by PowerShell.

```
default   https://myinstance.service-now.com   Paris   1.0.8
 basic   admin   json
user1   https://otherinstance.service-now.com   Paris   1.0.8
 basic   admin   yaml
```

- `table`: The output is formatted as a table which presents the information in a human-readable format.

```
NAME   HOST   HOST   VERSION   APP   VERSION   LOGIN   METHOD
 USERNAME   OUTPUT
-----------------------------------------------------------------
---------------
```

```
default   myinstance   Paris   1.0.8   basic   admin   json
user1   otherinstance   Paris   1.0.8   basic   admin   yaml
```

- `none`: The CLI does not print the output to the console. Success, error, and progress messages still display.

You can specify command output in two ways:

Use the `output` option in a named profile in the configuration file

The following example sets the default output format to text.

```
{
    "profiles":{
       "default":{
          "output":"text"
       }
}
```

Use the `--output` argument on the command line

The following example sets the output of a single command to JSON. This option on the command overrides any currently set value in the configuration file.

```
$ snc record query --table incident --query
 'active=true' --output json
```

Install the ServiceNow CLI

Install the ServiceNow CLI on a Mac, Windows, or Linux machine.

The latest version of the ServiceNow CLI is 1.1.0. To find your installed version and see if you must update, run `snc version`.

Installing the ServiceNow CLI requires:

- Admin rights to install software on your machine.
- Software to extract or unzip the downloaded package.

The sn_cli_metadata plugin needs to be installed on your instance to support full CLI functionality.

Role required: admin

Available for the following operating systems:

- Mac
- Windows
- Linux

Install the ServiceNow CLI on Mac

Install ServiceNow CLI on a Mac OS using the installer.

Before you begin

You must have an Apple-supported version of 64-bit Mac OS.
Role required: admin

Procedure

1. In your browser, download the installer bundle from the ServiceNow Store.
2. Extract the OS-specific installers.
3. Open the `snc-1.0.0-osx-installer` file.
4. Follow the on-screen instructions.
   a. When prompted, ensure the **Add to PATH** option is selected.
   This option requires root account access.
5. To verify that the installation succeeded, use the following commands.

```
$ which snc

/usr/local/bin/snc
```

```
$ snc version

{
 "extensions": {},
 "snc": "1.0.0"
}
```

Install the ServiceNow CLI on Windows

Install ServiceNow CLI on a Windows OS using the installer.

Before you begin

You must have a 64-bit version of Windows 10 or later.
Role required: admin

Procedure

1. In your browser, download the installer bundle from the ServiceNow Store.
2. Extract the OS-specific installers.
3. Open the `snc-1.0.0-windows-x64-installer.exe` file.
4. Follow the on-screen instructions.
   By default, the CLI installs to `C:\Program Files\ServiceNow CLI`.
5. To confirm the installation, check the version in the command line using the following command.

```
C:\>snc version

{
 "extensions": {},
 "snc": "1.0.0"
}
```

   If Windows is unable to find the program, close and reopen the command prompt window to refresh the path.

Install the ServiceNow CLI on Linux

Install ServiceNow CLI on a Linux machine using the installer.

Before you begin

You must have a 64-bit version of a recent distribution of CentOS or Ubuntu.
Role required: admin

Procedure

1. In your browser, download the installer bundle from the ServiceNow Store.
2. Extract the OS-specific installers.
3. Open the `snc-1.0.0-linux-x64-installer.run` file.
4. Follow the on-screen instructions.
   a. When prompted, ensure the **Add to PATH** option is selected.
   This option requires root account access.
5. To verify that the installation succeeded, use the following commands.

```
$ which snc

~/ServiceNow CLI/bin/snc
```

```
$ snc version

{
 "extensions": {},
 "snc": "1.0.0"
}
```

Configuring and managing your ServiceNow CLI connection profiles

Create a connection profile to connect with your instance, view connection profiles, refresh your connection and available commands, or delete profiles you no longer need.

The ServiceNow CLI stores profile information in a `config.json` file which, by default, is stored in your home directory at the following path:

- Linux and Mac: `~/.snc/config.json`
- Windows: `%USERPROFILE%\.snc\config.json`

The CLI uses this file to determine what information to use to connect to an instance, and what settings to use to generate output. By default, the ServiceNow CLI uses the settings found in the default profile to connect to an instance. To use alternate settings, you can create and reference additional named profiles. For more information, see Configuring and managing your ServiceNow CLI connection profiles.

The following example shows a#configuration#file with a default profile and a named profile. Each profile can use different credentials and specify different hosts and output formats.

```
{
   "profiles":{
      "default":{
         "host":"https://myinstance.service-now.com",
         "loginmethod":"basic",
         "username":"admin",
         "output":"json",
         "hostversion":"Paris",
```

```
         "appversion":"1.0"
         },
      "user1":{
         "host":"https://otherinstance.service-now.com",
         "loginmethod":"basic",
         "username":"user1",
         "output":"yaml",
         "hostversion":"Paris",
         "appversion":"1.0"
         }
      }
}
```

Create a default profile

Create a connection profile that the ServiceNow CLI uses by default. You must create a default profile to set up the CLI's initial connection with an instance.

About this task

By default, the information in this profile is used when you run a command that does not explicitly specify a profile to use.

> **Note:**
> Sensitive credential information is only stored in the OS keychain, not in the configuration file.

Procedure

1. Open your system's command-line tool and execute this command.

   ```
   $ snc configure profile set
   ```

   The CLI prompts you for more information.

2. Enter the requested values.
   When prompting for information, the CLI displays current values in brackets []. To keep an existing value, press the Enter key.

Requested information	Description
Host	The host name of the instance to connect to. Supports both the full URL (https://my-instance.service-now.com) or just the host name (my-instance).
Login method	The login method to use to connect to the instance. Supports `Basic`, `OAuth`, and `OAuth + MFA`.
Username	The user name to use to connect to the instance.
Password	The password to use to connect to the instance.
Client id	The client ID to use to connect to the instance when the login method is OAuth or OAuth + MFA.
Client secret	The client secret to use to connect to the instance when the login method is OAuth or OAuth + MFA.
Authentication code	The authentication code to use to connect to the instance when the login method is OAuth + MFA.

Requested information	Description
Default output format	Specifies how to format the command results. Options are `json`, `yaml`, `text`, and `table`.

Example

```
$ snc configure profile set

Host:
https://myinstance.service-now.com

Login Method (Basic, OAuth, OAuth + MFA):
Basic

Username:
myusername

Password:
mypassword
```

3. Run any command using the values specified in your default profile by omitting the `--profile` attribute from your command.

   Example

   ```
   $ snc record create
   ```

   The command creates a record in the instance specified in the default profile with the specified connection options.

Create a named profile

Create a named connection profile to use with specific commands. This allows you to specify a different instance or connection protocol for a specific command.

Before you begin

A default profile must first exist to communicate with an instance.

About this task

Note:
Sensitive credential information is only stored in the OS keychain, not in the configuration file.

Procedure

1. Open your system's command-line tool and execute this command.
2. Specify the name of the named profile in the *`profile-name`* argument.

   ```
   $ snc configure profile set [--profile profile-name]
   ```

   The CLI prompts you for more information.
3. Enter the requested values.
   When prompting for information, the CLI displays current values in brackets []. To keep an existing value, press the Enter key.

Requested information	Description
Host	The host name of the instance to connect to. Supports both the full URL (https://my-instance.service-now.com) or just the host name (my-instance).
Login method	The login method to use to connect to the instance. Supports `Basic`, `OAuth`, and `OAuth + MFA`.
Username	The user name to use to connect to the instance.
Password	The password to use to connect to the instance.
Client id	The client ID to use to connect to the instance when the login method is OAuth or OAuth + MFA.
Client secret	The client secret to use to connect to the instance when the login method is OAuth or OAuth + MFA.
Authentication code	The authentication code to use to connect to the instance when the login method is OAuth + MFA.
Default output format	Specifies how to format the command results. Options are `json`, `yaml`, `text`, and `table`.

Example

```
$ snc configure profile set --profile user1

Host:
https://mytestinstance.service-now.com

Login Method (Basic ,OAuth ,OAuth + MFA):
Basic

Username:
myusername

Password:
mypassword
```

4. Run any command using the values specified in your named profile by adding the`#--profile# <profile-name>` attribute to your command.

Example

```
$ snc record create --profile user1
```

The command creates a record in the instance specified in the `user1` profile with the specified connection options.

View profiles

View all connection profiles set in the configuration file, or view information about a specific profile.

Before you begin

A default profile must first exist to communicate with an instance.

About this task
For each profile, the CLI displays host information, version details, user name, login method, and preferred output format. It does not display sensitive information such as passwords or client IDs.

Procedure

Open your system's command-line tool and execute this command.

Include the `[--profile profile-name]` argument to view the connection information for a single profile. Otherwise, the CLI lists information for all profiles.

```
$ snc configure profile list [--profile profile-name]
```

Example

This command displays all configured profiles.

```
$ snc configure profile list
```

The CLI provides the following output.

```
{
   "profiles":{
      "default":{
      "host":"https://myinstance.service-now.com",
      "loginmethod":"basic",
      "username":"admin",
      "output":"json",
      "hostversion":"Paris",
      "appversion":"1.0"
   }
}
```

Remove a profile

Remove a named connection profile that you no longer need from the configuration file.

Before you begin
A default profile must first exist to communicate with an instance.

About this task
You cannot remove the default connection profile using this command. To remove the default connection profile, edit the configuration file manually.

Procedure

1. Open your system's command-line tool and execute this command.
2. Specify the name of the named profile in the *`profile-name`* argument.

```
$ snc configure profile remove [--profile profile-name]
```

Result
The CLI removes the specified profile from the configuration file.

Refresh your connection

Update the available commands from the instance for the given profile. Refresh your connection after modifying any of the commands on the corresponding instance in order to keep the CLI up-to-date.

Before you begin

A default profile must first exist to communicate with an instance.

Procedure

Open your system's command-line tool and execute this command.

Include the `[--profile profile-name]` argument to refresh connection information from a named profile. If you do not include the argument, the CLI refreshes the default profile.

```
$ snc configure profile refresh [--profile profile-name]
```

Result

The CLI connects to the instance in the designated connection profile and updates any commands that may have changed.

Get help with ServiceNow CLI

See available commands, command options, and examples, or generate debug logging output.

Before you begin

- Install the ServiceNow CLI
- Configuring and managing your ServiceNow CLI connection profiles
- Role required: none

About this task

In addition to the debug logging output option, the CLI logs all command executions to a log file at the following locations:

- Linux and Mac: `~/.snc/.logs`.
- Windows:`%USERPROFILE%\.snc\.logs`.

Procedure

1. To see the available commands and arguments, open your system's command-line tool and enter the command you need help with, followed by the `--help` argument.

   Example

   The following example displays help for the available top-level commands.

   ```
   $ snc --help
   ```

   Example

   The following example displays the available profile-specific commands.

   ```
   $ snc configure profile --help
   ```

   The CLI displays help for the requested command divided into eight sections:

Name

The name of the command.

```
Name
     set
```

Description

The description of what the command does and its return values.

```
Description
     Configure connection profiles. This command
 is interactive and will prompt you for each
 configuration value.
```

Synopsis

The basic syntax for using the command and its options.

```
Synopsis
     snc configure profile set [arguments]
```

Available commands

The commands available under the current command group.

```
Available Commands
 list    : Lists the configured connection profiles.
 refresh : Updates the available commands from the
 instance for the given profile.
 remove  : Removes the specified connection profile.
 set     : Configures connection profiles in order to
 communicate with an instance.
```

Command groups

The command groups available under the current command group.

```
Command Groups
     profile : Set, view, and remove connection
 profiles.
```

Arguments

A description of each of the arguments the command accepts.

```
Arguments
     -p, --profile string : Use a specific connection
 profile when executing a command.
```

Global arguments

The global arguments that the command accepts.

```
Global Arguments
     -d, --debug Print logs to console.
     -h, --help Display detailed help information.
     -o, --output string Set the format for printing
 command output.
```

Examples

Examples of the requested command.

```
Examples
   Create a new profile to save as the default:
      $ snc configure profile set
         Host:
         Login method:
         Username:
         Password:
         Client id:
         Client secret:
```

2. To generate debug logging output, open your system's command-line tool and enter the command you want to debug, followed by the `--debug` argument.

Example

The following example generates debug logging output when trying to perform the operation.

```
$ snc record delete --table incident --sysid
 552c48888c033300964f4932b03eb092 --debug
```

The CLI generates debug logging output when executing the command.

Perform record operations using ServiceNow CLI

Create, read, update, delete, and query records in your instance using the ServiceNow CLI command-line tool.

- Install the ServiceNow CLI
- Configuring and managing your ServiceNow CLI connection profiles

Note:
To perform record operations, install the CLI Metadata application from the ServiceNow Store on the instance you want to connect to.

Create a record

Inserts a single record in a specified table.

Before you begin

Role required: sn_cli_metadata.cli_admin, sn_cli_metadata.cli_user, or admin

About this task

Note:
You cannot insert multiple records using this command.

Procedure

Open your system's command-line tool and execute this command.

```
$ snc record create [--table table --data data]
```

Pass in values for these arguments.

Parameter	Description
table	Required. Name of the table in which to save the record.
data	Required. Field name and the associated value for each field to define in the specified record in JSON string format.

Result

The system creates a record in the designated table and returns field-value pairs of the new record.

Example:

```
$ snc record create --table incident, --data
 "{'short_description': 'Unable to open file', 'impact':'3'}"
```

The system returns the record in JSON format.

```
{
   "result": {
      "active": "true",
      "activity_due": "",
      "additional_assignee_list": "",
      "approval": "not requested",
      "approval_history": "",
      "approval_set": "",
      "assigned_to": "",
      "assignment_group": "",
      "business_duration": "",
      "business_service": "",
      "business_stc": "",
      "calendar_duration": "",
      "calendar_stc": "",
      "caller_id": "",
      "category": "inquiry",
      "caused_by": "",
      "child_incidents": "0",
      "close_code": "",
      "close_notes": "",
      "closed_at": "",
      "closed_by": "",
      "cmdb_ci": "",
      "comments": "",
      "comments_and_work_notes": "",
      "company": "",
      "contact_type": "",
      "contract": "",
      "correlation_display": "",
      "correlation_id": "",
      "delivery_plan": "",
      "delivery_task": "",
      "description": "",
      "due_date": "",
      "escalation": "0",
      "expected_start": "",
      "follow_up": "",
      "group_list": "",
```

```
      "hold_reason": "",
      "impact": "3",
      "incident_state": "1",
      "knowledge": "false",
      "location": "",
      "made_sla": "true",
      "notify": "1",
      "number": "INC0010005",
      "opened_at": "2021-01-27 22:49:11",
      "opened_by": {
         "link":
 "https://my-instance.service-now.com/api/now/table/sys_user/681
6f79cc0a8016401c5a33be04be441",
         "value": "6816f79cc0a8016401c5a33be04be441"
      },
      "order": "",
      "parent": "",
      "parent_incident": "",
      "priority": "5",
      "problem_id": "",
      "reassignment_count": "0",
      "reopen_count": "0",
      "reopened_by": "",
      "reopened_time": "",
      "resolved_at": "",
      "resolved_by": "",
      "rfc": "",
      "route_reason": "",
      "service_offering": "",
      "severity": "3",
      "short_description": "Unable to open file",
      "sla_due": "",
      "state": "1",
      "subcategory": "",
      "sys_class_name": "incident",
      "sys_created_by": "admin",
      "sys_created_on": "2021-01-27 22:49:11",
      "sys_domain": {
         "link":
 "https://my-instance.service-now.com/api/now/table/sys_user_gro
up/global",
         "value": "global"
      },
      "sys_domain_path": "/",
      "sys_id": "6a6a4f5cdbcae850d7055268dc9619d4",
      "sys_mod_count": "0",
      "sys_tags": "",
      "sys_updated_by": "admin",
      "sys_updated_on": "2021-01-27 22:49:11",
      "task_effective_number": "INC0010005",
      "time_worked": "",
      "universal_request": "",
      "upon_approval": "proceed",
      "upon_reject": "cancel",
      "urgency": "3",
      "user_input": "",
      "watch_list": "",
```

```
      "work_end": "",
      "work_notes": "",
      "work_notes_list": "",
      "work_start": ""
   }
}
```

Delete a record

Deletes the specified record from the specified table.

Before you begin

Role required: sn_cli_metadata.cli_admin, sn_cli_metadata.cli_user, or admin

Procedure

Open your system's command-line tool and execute this command.

```
$ snc record delete [--table table --sysid sys_id]
```

Pass in values for these arguments.

Parameter	Description
table	Required. Name of the table in which to delete the record.
sysid	Required. Sys_id of the record to delete.

Example:

```
$ snc record delete --table incident --sysid
 552c48888c033300964f4932b03eb092
```

Get a record

Retrieves a single record based on the specified sys_id from the specified table.

Before you begin

Role required: sn_cli_metadata.cli_admin, sn_cli_metadata.cli_user, or admin

Procedure

Open your system's command-line tool and execute this command.

```
$ snc record get [--table table --sysid sys_id]
```

Pass in values for these arguments.

Parameter	Description
table	Required. Name of the table from which to retrieve the record.
sysid	Required. Sys_id of the record to retrieve.

Example:

```
$ snc record get --table incident --sysid
 552c48888c033300964f4932b03eb092
```

The system returns the record in JSON format.


```
{
   "result": {
      "active": "true",
      "activity_due": "",
      "additional_assignee_list": "",
      "approval": "not requested",
      "approval_history": "",
      "approval_set": "",
      "assigned_to": "",
      "assignment_group": "",
      "business_duration": "",
      "business_service": "",
      "business_stc": "",
      "calendar_duration": "",
      "calendar_stc": "",
      "caller_id": {
         "link":
 "https://my-instance.service-now.com/api/now/table/sys_user/005
d500b536073005e0addeeff7b12f4",
         "value": "005d500b536073005e0addeeff7b12f4"
      },
      "category": "inquiry",
      "caused_by": "",
      "child_incidents": "0",
      "close_code": "",
      "close_notes": "",
      "closed_at": "",
      "closed_by": "",
      "cmdb_ci": "",
      "comments": "",
      "comments_and_work_notes": "",
      "company": "",
      "contact_type": "",
      "contract": "",
      "correlation_display": "",
      "correlation_id": "",
      "delivery_plan": "",
      "delivery_task": "",
      "description": "",
      "due_date": "",
      "escalation": "0",
      "expected_start": "",
      "follow_up": "",
      "group_list": "",
      "hold_reason": "",
      "impact": "3",
      "incident_state": "1",
      "knowledge": "false",
      "location": "",
      "made_sla": "true",
```

```
      "notify": "1",
      "number": "INC0010112",
      "opened_at": "2019-07-29 18:48:43",
      "opened_by": {
         "link":
 "https://my-instance.service-now.com/api/now/table/sys_user/681
6f79cc0a8016401c5a33be04be441",
         "value": "6816f79cc0a8016401c5a33be04be441"
      },
      "order": "",
      "parent": "",
      "parent_incident": "",
      "priority": "5",
      "problem_id": "",
      "reassignment_count": "0",
      "reopen_count": "0",
      "reopened_by": "",
      "reopened_time": "",
      "resolved_at": "",
      "resolved_by": "",
      "rfc": "",
      "route_reason": "",
      "service_offering": "",
      "severity": "3",
      "short_description": "Assessment :  ATF Assessor",
      "sla_due": "",
      "state": "1",
      "subcategory": "",
      "sys_class_name": "incident",
      "sys_created_by": "admin",
      "sys_created_on": "2019-07-29 18:49:28",
      "sys_domain": {
         "link":
 "https://my-instance.service-now.com/api/now/table/sys_user_gro
up/global",
         "value": "global"
      },
      "sys_domain_path": "/",
      "sys_id": "552c48888c033300964f4932b03eb092",
      "sys_mod_count": "0",
      "sys_tags": "",
      "sys_updated_by": "admin",
      "sys_updated_on": "2019-07-29 18:49:28",
      "task_effective_number": "INC0010112",
      "time_worked": "",
      "universal_request": "",
      "upon_approval": "proceed",
      "upon_reject": "cancel",
      "urgency": "3",
      "user_input": "",
      "watch_list": "",
      "work_end": "",
      "work_notes": "",
      "work_notes_list": "",
      "work_start": ""
```

```
    }
}
```

Query records

Retrieves multiple records from a specified table.

Before you begin

Role required: sn_cli_metadata.cli_admin, sn_cli_metadata.cli_user, or admin

Procedure

Open your system's command-line tool and execute this command.

```
$ snc record query [--displayvalue displayValue --fields fields
 --limit limit --offset offset --query query --table table]
```

Pass in values for these arguments.

Parameter	Description
displayValue	Include `--displayvalue` to retrieve the display value from the database for reference and choice fields. Do not include this parameter to retrieve the actual values.
fields	Comma-separated list of field names to return from the database.
limit	Maximum number of records to return.
offset	Starting record index for which to begin retrieving records. Use this value to paginate record retrieval.
query	Required. Encoded query used to filter the result set in the following format: `--query '<column_name><operator><value>'`.
table	Required. Name of the table in which to query the records.

Example:

```
$ snc record query --displayvalue --fields
 short_description,state --query '123TEXTQUERY321=email' --table
 incident
```

The CLI returns any records that match the query.

```
{
   "result": [
      {
         "short_description": "Unable to connect to email",
         "state": "Closed"
      }
   ]
}
```

Update a record

Updates the specified record with the given data attributes.

Before you begin

Role required: sn_cli_metadata.cli_admin, sn_cli_metadata.cli_user, or admin

Procedure

Open your system's command-line tool and execute this command.

```
$ snc record update [--sysid sys_id --table table --data data]
```

Pass in values for these arguments.

Parameter	Description
table	Required. Name of the table in which to save the record.
sysid	Required. Sys_id of the record to update.
data	Required. Field name and the associated value for each field to define in the specified record in JSON string format.

Example:

```
$ snc record update --sysid 9ef81de2db0d6090d7055268dc961978
 --table incident --data "{'short_description': 'Email servers
 down', 'urgency':'1'}"
```

The system returns field-value pairs for the updated record.


```
{
   "result": {
      "active": "true",
      "activity_due": "",
      "additional_assignee_list": "",
      "approval": "not requested",
      "approval_history": "",
      "approval_set": "",
      "assigned_to": "",
      "assignment_group": {
         "link":
 "https://my-instance.service-now.com/api/now/table/sys_user_gro
up/287ebd7da9fe198100f92cc8d1d2154e",
         "value": "287ebd7da9fe198100f92cc8d1d2154e"
      },
      "business_duration": "",
      "business_service": "",
      "business_stc": "",
      "calendar_duration": "",
      "calendar_stc": "",
      "caller_id": "",
      "category": "inquiry",
      "caused_by": "",
      "child_incidents": "0",
      "close_code": "",
      "close_notes": "",
      "closed_at": "",
      "closed_by": "",
      "cmdb_ci": "",
      "comments": "",
      "comments_and_work_notes": "",
      "company": "",
```

```
      "contact_type": "",
      "contract": "",
      "correlation_display": "",
      "correlation_id": "",
      "delivery_plan": "",
      "delivery_task": "",
      "description": "",
      "due_date": "",
      "escalation": "0",
      "expected_start": "",
      "follow_up": "",
      "group_list": "",
      "hold_reason": "",
      "impact": "2",
      "incident_state": "1",
      "knowledge": "false",
      "location": "",
      "made_sla": "true",
      "notify": "1",
      "number": "INC0010003",
      "opened_at": "2020-12-15 21:17:05",
      "opened_by": {
         "link":
 "https://my-instance.service-now.com/api/now/table/sys_user/681
6f79cc0a8016401c5a33be04be441",
         "value": "6816f79cc0a8016401c5a33be04be441"
      },
      "order": "",
      "parent": "",
      "parent_incident": "",
      "priority": "2",
      "problem_id": "",
      "reassignment_count": "0",
      "reopen_count": "0",
      "reopened_by": "",
      "reopened_time": "",
      "resolved_at": "",
      "resolved_by": "",
      "rfc": "",
      "route_reason": "",
      "service_offering": "",
      "severity": "3",
      "short_description": "Email servers down",
      "sla_due": "",
      "state": "1",
      "subcategory": "",
      "sys_class_name": "incident",
      "sys_created_by": "admin",
      "sys_created_on": "2020-12-15 21:17:05",
      "sys_domain": {
         "link":
 "https://my-instance.service-now.com/api/now/table/sys_user_gro
up/global",
         "value": "global"
      },
      "sys_domain_path": "/",
      "sys_id": "9ef81de2db0d6090d7055268dc961978",
```

```
        "sys_mod_count": "1",
        "sys_tags": "",
        "sys_updated_by": "admin",
        "sys_updated_on": "2021-01-27 22:56:33",
        "task_effective_number": "INC0010003",
        "time_worked": "",
        "universal_request": "",
        "upon_approval": "proceed",
        "upon_reject": "cancel",
        "urgency": "1",
        "user_input": "",
        "watch_list": "",
        "work_end": "",
        "work_notes": "",
        "work_notes_list": "",
        "work_start": ""
    }
}
```


Create a custom command in ServiceNow CLI

Manage your custom application from the command line by creating custom commands in the ServiceNow CLI.

Before you begin

Role required: admin

About this task

A ServiceNow CLI command maps to a scripted REST endpoint in the End Point [sn_cli_metadata_end_point] table. You can define a scripted REST endpoint to perform a function in your custom application, or use any existing REST endpoint. Then map a CLI command to execute the REST call.

Procedure

1. Make a REST endpoint available to a ServiceNow CLI command.

   a. Navigate to **Command Line Interface (CLI)** > **End Points**.

   b. Select **New** and complete the form.

Field	Description
Resource Path	Required. Path to the endpoint on the instance to map a command to. Can be the path to an inbound REST API, or a scripted REST API. For example, `api/now/table/{table}/{sysid}`. For more information, see Available REST APIs and Scripted REST APIs.
HTTP Method	Required. HTTP method to use when the user runs the associated command.
Application	Read-only application scope for the endpoint.

2. **Optional:** Create a command group.
   Alternatively, you can add your new command to an existing command group.

a. Navigate to **Command Line Interface (CLI)** > **Command Groups**.

b. Select **New** and complete the form.

Field	Description
Name	Required. Name of the command group.
Parent Group	Parent command group.
Reference Group	Command group to reference. For example, you can create a new command group as an alias for an existing command group. When the user calls a referenced command using the new command group, the original command executes. This enables you to create a command group specific to your custom application that includes both new and existing commands.
Short Description	Required. Short description for the command group.
Description	Description of the command group used as help text when the user runs the `--help` command on the command group.
Application	Read-only application scope for the command group.
Active	When selected, the command group is active.

3. Create a command.

a. Navigate to **Command Line Interface (CLI)** > **Commands**.

b. Select **New** and complete the form.

Field	Description
Name	Required. Name of the command.
Command Group	Required. Command group that the command is a part of.
Reference Command	Command to reference. For example, you can create a new command as an alias for an existing command. When the user calls the new command, the original command executes. This enables you to create a command specific to your custom application that executes existing functionality. **Note:** A command cannot reference a command that references another command, or reference a descendant command, an ancestor command, or a callback command.
API Endpoint	Required. API call to execute when the user runs the command.
Short Description	Required. Short description of the command.
Application	Read-only application scope for the command.

Field	Description
Active	When selected, the command is active.
Is Callback Command	When true, designates the command as a callback command. Select this option to hide the command from the CLI client and prevent users from calling it from the command line. Use this field with the **Callback** section of a primary command. The callback command executes when the primary command is complete.
Help Text	
Description	Description of the command used as help text when the user runs the `--help` command.
Examples	Examples of the command used as help text when the user runs the `--help` command.
Expressions	
Success Expression	Expression used to evaluate the response from the server and determine if the command succeeded. For example, `result.code = 1`.
Failure Expression	Expression used to evaluate the response from the server and determine if the command failed.
Messages	
Success Message	Message displayed on the CLI client when the command is successful.
Progress Message	Message displayed on the CLI client when the command is in progress.
Failure Message	Message displayed on the CLI client when the command fails.
Callback	
Callback Expression	Expression used to determine whether to execute the callback command. For example, you can write an expression that checks on a long-running process. If the expression produces a certain result, the callback executes.
Callback Command	Command to execute when the **Callback Expression** is satisfied. Must be a command with the **Is Callback Command** field selected.
Callback Interval	The interval between callback command executions. Unit: Milliseconds Default: 1,000
Max Retries	Maximum number of times the callback command executes. Default: 10

c. In the Command Arguments related lists, create any command arguments needed for the command.

Create a command argument to allow users to set options when running a command.

Field	Description
Name	Required. Name of the command argument, for example `data`.
Short Name	Short dame of the command argument, for example `d`.
Data Type	Required. Type of data expected for the command argument. Options include: ▪ String: Allows users to enter a string as input. ▪ Integer: Allows users to enter a number as input. ▪ Boolean: Allows users to enter `true` or `false` as input. ▪ File Input: Allows users to upload a file as input. Map this argument to a Body **Parameter Type** in the API Endpoint Arguments related list in the next step. Users can pass plain text, YAML, JSON, or another file type accepted by the REST API. When the user submits a YAML file, the CLI client converts to file to JSON format by default. Use the **Skip Pre-processing** field to disable this behavior. ▪ Password: Allows users to securely enter a password as input. **Note:** For File Input data types, the file size limit is 10 MB by default. However, you change this limit using the *`glide.rest.scripted.max_inbound_content_length_mb`* system property.
Skip Pre-processing	Set this flag to prevent the CLI client from converting YAML files into JSON format before sending them in the request. When **Data Type** is set to **File Input** and the user submits a YAML file, the CLI client converts the file to JSON before executing the command by default. **Note:** Only applies when **Data Type** is set to **File Input**.
Default Value	Default value to use when the user does not pass a value.
Short Description	Required. Short description of the command argument. Used as help text when the user runs the `--help` command.
Application	Read-only application scope for the argument.
Command	Required. Command that the argument applies to.
Mandatory	When selected, the user must provide a value for the argument when running the associated command.
Prompt	Prompt to request information from the user. The CLI prompts for information when the user does not include a required argument in a command.
Visibility Expression	Expression used to determine whether the CLI should prompt for the argument. Typically used to display an argument based on the value provided to a previous argument.

Field	Description
Order	Order in which to prompt for the argument.

d. In the API Endpoint Arguments related lists, create any API endpoint arguments needed for the command.

Map command arguments from the Command Arguments related list to parameters in your REST endpoint.

Field	Description
Name	Required. Name of the argument.
Value	Required. The value from the command that you want to pass to the REST endpoint. You can pass a static value, or an expression such as `{flags.table}`. Use the `flags` global variable to access the command arguments.
Parameter Type	Required. The type of parameter in the REST endpoint that you want to pass the **Value** to. For example, if you select **Body**, the CLI passes the value of the **Value** field to the REST endpoint body. Options include: ▪ Body ▪ Header ▪ Path ▪ Query
Application	Read-only application scope for the argument.
Command	Required. Command that the argument applies to.

e. In the Return Values related lists, create any return values needed for the command.

Create return values to only return certain keys from the response.

Field	Description
Path Expression	Required. Expression representing the path to the key that you want to return.
Alias	Variable name to assign the return value to.
Application	Read-only application scope for the command.
Command	Required. Command that the return value applies to.

Result

When the user runs the ServiceNow CLI command, the system executes the associated REST API call and returns the result to the ServiceNow CLI.

Manage ServiceNow CLI extensions

Add extensions to the ServiceNow CLI to load additional functionality and commands, update existing extensions, or remove extensions you no longer need.

Procedure

1. Find available extensions.

   a. Open your system's command-line tool and execute this command.

   ```
   $ snc extension list-available -o table
   ```

   The system lists available extensions and associated details.

   ```
   NAME       DESCRIPTION
    VERSION            INSTALLED
   ------------------------------------------------------------------
   -------------------------
   ui-component        Build and deploy components
       19.0.0-alpha.15     false
   ```

2. Install an extension.

   a. Open your system's command-line tool and execute this command.

   ```
   $ snc extension add --name <extension-name>
   ```

3. **Optional:** Update an extension.

   a. Open your system's command-line tool and execute this command.

   ```
   $ snc extension update --name <extension-name>
   ```

4. **Optional:** Remove an extension.

   a. Open your system's command-line tool and execute this command.

   ```
   $ snc extension remove --name <extension-name>
   ```

ServiceNow CLI available commands

Commands and command arguments available to the base system ServiceNow CLI.

Global command arguments

Use command arguments to set options for any CLI commands.

--help

Provides help information for the specified command, which includes the description, supported arguments, and examples. For more information, see Get help with ServiceNow CLI.

```
$ snc --help
```

--debug

provides debug logging output when executing a command. For more information, see Get help with ServiceNow CLI.

```
$ snc record delete --table incident --sysid
 552c48888c033300964f4932b03eb092 --debug
```

--profile

Specifies the named profile to use for a command. For more information, see Configuring and managing your ServiceNow CLI connection profiles.

```
$ snc configure profile set --profile <profilename>
```

--output

Specifies the output format to use for a command. The ServiceNow CLI supports the following output formats.

- `json`: The output is formatted as JSON. This is the default.

```
{
   "default": {
      "appversion": "1.0.8",
      "host": "https://myinstance.service-now.com",
      "hostversion": "Paris",
      "loginmethod": "basic",
      "output": "json",
      "username": "admin"
   },
   "user1": {
      "appversion": "1.0.8",
      "host": "https://otherinstance.service-now.com",
      "hostversion": "Paris",
      "loginmethod": "basic",
      "output": "yaml",
      "username": "admin"
   }
}
```

- `yaml`: The output is formatted as YAML. Use YAML to handle the output with services and tools that emit or consume YAML-formatted strings.

```
default:
   appversion: 1.0.8
   host: https://myinstance.service-now.com
   hostversion: Paris
   loginmethod: basic
   output: json
   username: admin
user1:
   appversion: 1.0.8
   host: https://otherinstance.service-now.com
   hostversion: Paris
   loginmethod: basic
   output: yaml
   username: admin
```

- `text`: The output is formatted as multiple lines of tab-separated string values. Use this output with traditional UNIX text tools such as grep, sed, and awk, and the text processing performed by PowerShell.

```
default   https://myinstance.service-now.com   Paris
 1.0.8   basic   admin   json
user1   https://otherinstance.service-now.com   Paris
  1.0.8   basic   admin   yaml
```

- `table`: The output is formatted as a table which presents the information in a human-readable format.

```
NAME   HOST   HOST   VERSION   APP   VERSION   LOGIN
 METHOD   USERNAME   OUTPUT
-------------------------------------------------------
-------------------------
default   myinstance   Paris   1.0.8   basic   admin
 json
user1   otherinstance   Paris   1.0.8   basic   admin
  yaml
```

- `none`: The CLI does not print the output to the console. Success, error, and progress messages still display.

```
$ snc record query --table incident --query
 'active=true' --output json
```

--no-interactive

Prevents the CLI from prompting the user for argument values. If the user does not pass a value for a required argument, the system uses the default value. If no default value is defined, the system throws an error.

--no-verbose

The command executes silently without messages. Use this argument in automated testing scenarios.

Configure profile

Create a connection profile to connect with your instance, view connection profiles, refresh your connection and available commands, or delete profiles you no longer need. For more information about configuring your profile, see Configuring and managing your ServiceNow CLI connection profiles.

Set up a default profile

Create a connection profile that the ServiceNow CLI uses by default. You must create a default profile to set up the CLI's initial connection with an instance.

```
$ snc configure profile set
```

The system prompts you for the following information:

Requested information	Description
Host	The host name of the instance to connect to. Supports both the full URL (https://my-instance.service-now.com) or just the host name (my-instance).

Requested information	Description
Login method	The login method to use to connect to the instance. Supports `Basic`, `OAuth`, and `OAuth + MFA`.
Username	The user name to use to connect to the instance.
Password	The password to use to connect to the instance.
Client id	The client ID to use to connect to the instance when the login method is OAuth or OAuth + MFA.
Client secret	The client secret to use to connect to the instance when the login method is OAuth or OAuth + MFA.
Authentication code	The authentication code to use to connect to the instance when the login method is OAuth + MFA.
Default output format	Specifies how to format the command results. Options are `json`, `yaml`, `text`, and `table`.

Set up a named profile

Create a named connection profile to use with specific commands. This allows you to specify a different instance or connection protocol for a specific command.

```
$ snc configure profile set [--profile profile-name]
```

The system prompts you for the following information:

Requested information	Description
Host	The host name of the instance to connect to. Supports both the full URL (https://my-instance.service-now.com) or just the host name (my-instance).
Login method	The login method to use to connect to the instance. Supports `Basic`, `OAuth`, and `OAuth + MFA`.
Username	The user name to use to connect to the instance.
Password	The password to use to connect to the instance.
Client id	The client ID to use to connect to the instance when the login method is OAuth or OAuth + MFA.
Client secret	The client secret to use to connect to the instance when the login method is OAuth or OAuth + MFA.
Authentication code	The authentication code to use to connect to the instance when the login method is OAuth + MFA.
Default output format	Specifies how to format the command results. Options are `json`, `yaml`, `text`, and `table`.

View profiles

View all connection profiles set in the configuration file, or view information about a specific profile.

```
$ snc configure profile list [--profile profile-name]
```

Remove a profile

Remove a named connection profile that you no longer need from the configuration file.

```
$ snc configure profile remove [--profile profile-name]
```

Refresh a profile

Update the available commands from the instance for the given profile. Refresh your connection after modifying any of the commands on the corresponding instance in order to keep the CLI up-to-date.

```
$ snc configure profile refresh [--profile
 profile-name]
```

Perform record operations

Create, read, update, delete, and query records in your instance using the ServiceNow CLI command-line tool. For more information about performing record operations, see Perform record operations using ServiceNow CLI.

Create a record

Inserts a single record in a specified table.

```
$ snc record create [--table table --data data]
```

Pass in values for these arguments.

Parameter	Description
table	Required. Name of the table in which to save the record.
data	Required. Field name and the associated value for each field to define in the specified record in JSON string format.

Delete a record

Deletes the specified record from the specified table.

```
$ snc record delete [--table table --sysid sys_id]
```

Pass in values for these arguments.

Parameter	Description
table	Required. Name of the table in which to delete the record.
sysid	Required. Sys_id of the record to delete.

Get a record

Retrieves a single record based on the specified sys_id from the specified table.

```
$ snc record get [--table table --sysid sys_id]
```

Pass in values for these arguments.

Parameter	Description
table	Required. Name of the table from which to retrieve the record.
sysid	Required. Sys_id of the record to retrieve.

Query records

Retrieves multiple records from a specified table.

```
$ snc record query [--displayvalue displayValue
 --fields fields --limit limit --offset offset --query
 query --table table]
```

Pass in values for these arguments.

Parameter	Description
displayValue	Include `--displayvalue` to retrieve the display value from the database for reference and choice fields. Do not include this parameter to retrieve the actual values.
fields	Comma-separated list of field names to return from the database.
limit	Maximum number of records to return.
offset	Starting record index for which to begin retrieving records. Use this value to paginate record retrieval.
query	Required. Encoded query used to filter the result set in the following format: `--query '<column_name><operator><value>'`.
table	Required. Name of the table in which to query the records.

Update a record

Updates the specified record with the given data attributes.

```
$ snc record update [--sysid sys_id --table table
 --data data]
```

Pass in values for these arguments.

Parameter	Description
table	Required. Name of the table in which to save the record.
sysid	Required. Sys_id of the record to update.
data	Required. Field name and the associated value for each field to define in the specified record in JSON string format.

Work with extensions

Add extensions to the ServiceNow CLI to load additional functionality and commands, update existing extensions, or remove extensions you no longer need. For more information, see Manage ServiceNow CLI extensions.

Find available extensions

```
$ snc extension list-available -o table
```

Install an extension

```
$ snc extension add --name <extension-name>
```

Update an extension

```
$ snc extension update --name <extension-name>
```

Remove an extension

```
$ snc extension remove --name <extension-name>
```

Use the ui-component extension

Add the ui-component extension

Add the ui-component extension to the ServiceNow CLI.

```
$ snc extension add --name ui-component
```

Set up your project

Create the component project and the set of files required to develop a component. You can connect to your instance and create an application scope for your component, or you can reserve a scope to verify later.

```
$ snc ui-component project [--name name --description
 description --scope scope --offline]
```

Pass in values for these arguments.

Name	Description
name	Required. The project name. Must be a valid and unique npm package name.
description	The project description to be available in the npm registry and the plugins list in your instance.
scope	Suggested application scope to assign to this project and its components. If provided, the instance validates the name. Use the namespace identifier guidelines for application development on the instance. For more information, see Application scope. Maximum: 18 characters. Case: snake case. Default: `x_customerprefix_componentname`, where: • `customerprefix` is the value in the *`glide.appcreator.company.code`* system property on your instance. • `componentname` is the value provided in the component's `name` parameter when you created the project.

Name	Description
	Alternatively, you can add a value to the `scopeName` parameter in the `now-ui.json` file. For more information, see Change a component's application scope.
offline	When true, creates and scaffolds a component while disconnected from your instance. Skips validation of the given scope name. Default: `false`.

Run the development server

Add your component code and test it using a local development server.

```
$ snc ui-component develop [--entry entry --open --port
 port --host host]
```

Pass in values for these arguments.

Name	Description
entry	Path to the test module in your component project. Default: `example/index.js`.
open	Opens the default browser and navigates to the test page. Default: false.
port	Port where the development server runs. Default: 8081.
host	Host address to use if you want your local development server to be accessible externally by others. Typically set to `0.0.0.0`

Deploy a component to an instance

Deploy your component to display in your instance as an application plugin.

```
$ snc ui-component deploy [--open --force]
```

Pass in values for these arguments.

Name	Description
open	When true, opens the default browser and navigates to UI Builder in your instance. Default: false
force	Deploys component changes and overwrites any existing component records. Default: false.

Commands installed with CMDB Application CLI and API

Commands and command groups available to the ServiceNow CLI when the app-cmdb-api-cli plugin is installed.

Request apps on the Store

Visit the ServiceNow Store website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

To script critical operations which support automation across the enterprise, you can leverage APIs or run command line operations that the CMDB Application CLI and API store app provide instead of using the user interface. The CMDB Application CLI and API store app provides a robust framework which consolidates all the APIs that are related to application services and the command lines that let you access the interface to those APIs.

CMDB Application CLI and API commands enable the following tasks:

- Registering and creating an application service and establishing upstream relationships
- Getting details of a given application service and its upstream relationships
- Connecting higher level constructs such as business applications and business service offerings
- Populating an application service with a given population type
- Changing the state of an application service

For the REST API solution, see SG Services API.

Convert Application Service

Converts a manual or empty type application service to a calculated application service. During conversion, the application service record moves into the [cmdb_ci_service_calculated] table with the newly assigned class.

Command group:

- Parent group: service-graph
- Child group: app-service

Roles required

app_service_admin

If using a service mapping related service, the user must have the required roles for that service mapping related service.

Command structure for Mac OS

```
snc service-graph app-service convert --data '{JSON}'
```

Command structure for Windows OS

```
snc service-graph app-service convert --data "{JSON}"
```

Arguments

This command passes a JSON object using the *data* parameter.

The following properties for identifying a CI take precedence as follows:

1. sys_id – If sys_id, the system only uses the sys_id and ignores any additional values.
2. number – If provided without the sys_id, the system only uses the number and ignores any additional values.
3. <IRE field name> – The system only uses these values if the sys_id or number are not provided.

JSON object properties

Name	Description
<IRE field name>	One or more IRE fields identifying the application service. For example, name or version. Data type: String
levels	Number of levels to include in the conversion. Data type: String
number	Unique number that identifies the application service. Data type: String
sys_id	Sys_id of the application service. Data type: String

Example on Mac OS

```
snc service-graph app-service
 convert --data '{"name":"Test
 Register","environment":"Test","version":"1.0","levels
":8}'
```

Example on Windows OS

```
snc service-graph app-service
 convert --data "{"name":"Test
 Register","environment":"Test","version":"1.0","levels
":8}"
```

Return value

```
{
  "result": {
  "status": "success"
  }
}
```

Create Application Service Relationship

Constructs upstream relations such as business applications, business service offerings, and other application services. Running this command creates a relationship, taking input with a single parent and a corresponding child object.

Command group:

- Parent group: service-graph
- Child group: app-service

Roles required

app_service_admin

If using a service mapping related service, the user must have the required roles for that service mapping related service.

Command structure for Mac OS

```
snc service-graph app-service create-relationship
 --data '{JSON}'
```

Command structure for Windows OS

```
snc service-graph app-service create-relationship
 --data "{JSON}"
```

Arguments

This command passes a JSON object using the *data* parameter.

The following properties for identifying a CI take precedence as follows:

1. sys_id – If sys_id, the system only uses the sys_id and ignores any additional values.
2. number – If provided without the sys_id, the system only uses the number and ignores any additional values.
3. <IRE field name> – The system only uses these values if the sys_id or number are not provided.

JSON object properties

Name	Description
child	Information identifying the child application service with which to create a relationship. The child is located in the Service Instance [cmdb_ci_service_auto] table. A dynamic CI group can be added as a child but cannot be parent. Data type: Object `"child": { "<IRE field name>": "String", "number": "String", "sys_id": "String" }`
child.<IRE field name>	One or more IRE fields identifying the child application service. For example, name or version. Data type: String

JSON object properties (continued)

Name	Description
child.number	Unique number that identifies the child application service. Data type: String
child.sys_id	Sys_id of the child application service listed in the Service Instance [cmdb_ci_service_auto]. Data type: String
parent	Details identifying the parent application service with which to create a relationship. Data type: Object `"parent": {` `  "<IRE field name>": "String",` `  "number": "String",` `  "sys_id": "String",` `  "class_name": "String"` `}`
parent.<IRE field name>	One or more IRE fields identifying the application service. For example, name or version. Data type: String
parent.number	Unique number that identifies the application service. Data type: String
parent.sys_id	Sys_id of the application service listed in the Service Instance [cmdb_ci_service_auto]. Data type: String
parent.class_name	Name of the class that contains the application service. The parent class name should be from one of the following tables: • cmdb_ci_service_auto • cmdb_ci_service_discovered • cmdb_ci_service_by_tags • cmdb_ci_service_calculated • service_offering • cmdb_ci_business_app Default: cmdb_ci_service_auto Data type: String

Example on Mac OS

```
snc service-graph app-service
 create-relationship --data
 '{"child":{"name":"wdfsdf","environment":"Test","versi
on":"1.0"},"parent":{"sys_id":"abcdefg","name":"busin
ess App1","class_name":"service_offering"}}'
```

Example on Windows OS

```
snc service-graph app-service
 create-relationship --data
 "{"child":{"name":"wdfsdf","environment":"Test","versi
on":"1.0"},"parent":{"sys_id":"abcdefg","name":"busin
ess App1","class_name":"service_offering"}}"
```

Return value

```
{
  "result": {
  "status": "success"
  }
}
```

Delete Application Service

Deletes an application service.

Command group:

- Parent group: service-graph
- Child group: app-service

Roles required

app_service_admin

If using a service mapping related service, the user must have the required roles for that service mapping related service.

Command structure for Mac OS

```
snc service-graph app-service delete --data '{JSON}'
```

Command structure for Windows OS

```
snc service-graph app-service delete --data "{JSON}"
```

Arguments

This command passes a JSON object using the *data* parameter.

The following properties for identifying a CI take precedence as follows:

1. sys_id – If sys_id, the system only uses the sys_id and ignores any additional values.
2. number – If provided without the sys_id, the system only uses the number and ignores any additional values.
3. <IRE field name> – The system only uses these values if the sys_id or number are not provided.

JSON object properties

Name	Description
<IRE field name>	One or more IRE fields identifying the application service. For example, name or version. Data type: String
number	Unique number that identifies the application service. Data type: String
sys_id	Sys_id of the application service listed in the Application Service [cmdb_ci_service_auto] table. Data type: String

Example on Mac OS

```
snc service-graph app-service
 delete --data '{"name":"Test
 Register","environment":"Test","version":"1.0"}'
```

Example on Windows OS

```
snc service-graph app-service
 delete --data "{"name":"Test
 Register","environment":"Test","version":"1.0"}"
```

Return value

```
{
  "result": {
  "status": "success"
  }
}
```

Delete Application Service Relationship

Deletes an application service upstream relationship.

Command group:

- Parent group: service-graph
- Child group: app-service

Roles required

app_service_admin

If using a service mapping related service, the user must have the required roles for that service mapping related service.

Command structure for Mac OS

```
snc service-graph app-service delete-relationship
 --data '{JSON}'
```

Command structure for Windows OS

```
snc service-graph app-service delete-relationship
 --data "{JSON}"
```

Arguments

This command passes a JSON object using the *data* parameter.

The following properties for identifying a CI take precedence as follows:

1. sys_id – If sys_id, the system only uses the sys_id and ignores any additional values.
2. number – If provided without the sys_id, the system only uses the number and ignores any additional values.
3. <IRE field name> – The system only uses these values if the sys_id or number are not provided.

JSON object properties

Name	Description
child	Information describing the child relationship to be deleted from the service application. Data type: Object `"child": { "<IRE field name>": "String", "number": "String", "sys_id": "String" }`
child.<IRE field name>	One or more IRE fields identifying the child application service. For example, name or version. Data type: String
child.number	Unique number that identifies the child application service. Data type: String
child.sys_id	Sys_id of the child application service listed in the Service Instance [cmdb_ci_service_auto]. Data type: String
parent	Details identifying the parent application service from which to remove a relationship. Data type: Object `"parent": { "<IRE field name>": "String", "number": "String", "sys_id": "String", "class_name": "String" }`

JSON object properties (continued)

Name	Description
parent.<IRE field name>	One or more IRE fields identifying the application service. For example, name or version. Data type: String
parent.number	Unique number that identifies the application service. Data type: String
parent.sys_id	Sys_id of the application service listed in the Application Service [cmdb_ci_service_auto] table. Data type: String
parent.class_name	Name of the class that contains the application service. The parent class name should be from one of the following tables: • cmdb_ci_service_auto • cmdb_ci_service_discovered • cmdb_ci_service_by_tags • cmdb_ci_service_calculated • service_offering • cmdb_ci_business_app Default: cmdb_ci_service_auto Data type: String

Example on Mac OS

```
snc service-graph app-service delete-relationship
 --data '{"child":{"name":"Test
 Register","environment":"Test","version":"1.0"},"paren
t":{"name":"business Service
 Offering1","class_name":"service_offering"}}'
```

Example on Windows OS

```
snc service-graph app-service delete-relationship
 --data "{"child":{"name":"Test
 Register","environment":"Test","version":"1.0"},"paren
t":{"name":"business Service
 Offering1","class_name":"service_offering"}}"
```

Return value

```
{
  "result": {
  "status": "success"
```

```
    }
}
```

Find Application Service

Finds the details of a given application service and its upstream relationships.

Command group:

- Parent group: service-graph
- Child group: app-service

Roles required

app_service_admin – This role provides unlimited viewing of application services.

app_service_user – This role only provides viewing application services in Operational status.

If using a service mapping related service, the user must have the required roles for that service mapping related service.

Command structure for Mac OS

```
snc service-graph app-service find --data '{JSON}'
```

Command structure for Windows OS

```
snc service-graph app-service find --data "{JSON}"
```

Arguments

This command passes a JSON object using the *data* parameter.

The following properties for identifying a CI take precedence as follows:

1. sys_id – If sys_id, the system only uses the sys_id and ignores any additional values.
2. number – If provided without the sys_id, the system only uses the number and ignores any additional values.
3. <IRE field name> – The system only uses these values if the sys_id or number are not provided.

JSON object properties

Name	Description
<IRE field name>	One or more IRE fields identifying the application service. For example, name or version. Data type: String
number	Unique number that identifies the application service. Data type: String
sys_id	Sys_id of the application service listed in the Application Service [cmdb_ci_service_auto] table.

JSON object properties (continued)

Name	Description
	Data type: String

Example on Mac OS

```
snc service-graph app-service find --data '{"name" :
 "Test App Service1"}'
```

Example on Windows OS

```
snc service-graph app-service find --data "{"name" :
 "Test App Service1"}"
```

Return value


```
{
  "result": {
    "aliases": null,
    "asset": null,
    "asset_tag": null,
    "assigned": "",
    "assigned_to": null,
    "assignment_group": null,
    "attestation_score": null,
    "attested": "0",
    "attested_by": null,
    "attested_date": "",
    "attributes": null,
    "bucket": null,
    "business_contact": null,
    "business_need": null,
    "business_relation_manager": null,
    "business_unit": null,
    "busines_criticality": "4 - not critical",
    "can_print": "0",
    "category": null,
    "change_control": null,
    "checked_in": "",
    "checked_out": "",
    "checkout": null,
    "comments": null,
    "company": null,
    "compatibility_dependencies": null,
    "consumer_type": "internal",
    "correlation_id": null,
    "cost": null,
    "cost_cc": "USD",
    "cost_center": null,
    "delivery_date": "",
    "delivery_manager": null,
    "department": null,
    "discovery_source": "Manual Entry",
    "dns_domain": null,
    "due": "",
```

```
    "due_in": null,
    "duplicate_of": null,
    "end_date": "",
    "environment": null,
    "fault_count": "0",
    "first_discovered": "2021-07-19 20:09:48",
    "fqdn": null,
    "gl_account": null,
    "hide_from_dashboard": "0",
    "install_date": "",
    "install_status": "1",
    "invoice_number": null,
    "ip_address": null,
    "justification": null,
    "last_discovered": "2021-07-19 20:09:48",
    "last_review_date": "",
    "layer": null,
    "lease_id": null,
    "life_cycle_stage": null,
    "life_cycle_stage_status": null,
    "location": null,
    "mac_address": null,
    "maintenance_schedule": null,
    "managed_by": null,
    "managed_by_group": null,
    "manufacturer": null,
    "model_id": null,
    "model_number": null,
    "monitor": "0",
    "monitoring_requirements": null,
    "name": "Test App Service1",
    "number": "SNSVC0001014",
    "operational_status": "2",
    "order_date": "",
    "owned_by": null,
    "parent": null,
    "portfolio_status": "pipeline",
    "po_number": null,
    "prerequisites": null,
    "price_model": "per_unit",
    "price_unit": null,
    "published_ref": null,
    "purchase_date": "",
    "schedule": null,
    "serial_number": null,
    "service_classification": "Application Service",
    "service_level_requirement": null,
    "service_owner_delegate": null,
    "service_status": "requirements",
    "severity": null,
    "short_description": null,
    "skip_sync": "0",
    "sla": null,
    "spm_service_portfolio": null,
    "spm_taxonomy_node": null,
    "stakeholders": null,
    "start_date": "",
```

```
    "state": null,
    "subcategory": null,
    "supported_by": null,
    "support_group": null,
    "sys_class_name": "cmdb_ci_service_auto",
    "sys_class_path": "/!!/!7/!(",
    "sys_created_by": "admin",
    "sys_created_on": "2021-07-19 20:09:48",
    "sys_domain": "global",
    "sys_domain_path": "/",
    "sys_id": "a2f0618040697410f87713b656474255",
    "sys_mod_count": "0",
    "sys_updated_by": "admin",
    "sys_updated_on": "2021-07-19 20:09:48",
    "unit_description": null,
    "unverified": "0",
    "used_for": "Production",
    "user_group": null,
    "vendor": null,
    "version": null,
    "view_service": "61e1cb757f23220002d31ccebefa9120",
    "warranty_expiration": "",
    "relationships": [
      {
        "name": "Test Biz App1",
        "sys_id": "0250a94040697410f87713b656474250",
        "number": "APM0001001",
        "class_name": "cmdb_ci_business_app",
        "relationship": "Consumes::Consumed by"
      },
      {
        "name": "Tech Service Offering1",
        "sys_id": "98d0ed4040697410f87713b6564742ef",
        "number": "BSN0001005",
        "class_name": "service_offering",
        "relationship": "Contains::Contained by"
      }
    ]
  }
}
```


Populate Application Service

Populates an application service with a service population method.

Command group:

- Parent group: service-graph
- Child group: app-service

Roles required

app_service_admin

If using a service mapping related service, the user must have the required roles for that service mapping related service.

Command structure for Mac OS

```
snc service-graph app-service populate --data '{JSON}'
```

Command structure for Windows OS

```
snc service-graph app-service populate --data "{JSON}"
```

Arguments

This command passes a JSON object using the *data* parameter.

The following properties for identifying a CI take precedence as follows:

1. sys_id – If sys_id, the system only uses the sys_id and ignores any additional values.
2. number – If provided without the sys_id, the system only uses the number and ignores any additional values.
3. <IRE field name> – The system only uses these values if the sys_id or number are not provided.

JSON object properties

Name	Description
<IRE field name>	One or more IRE fields identifying the application service. For example, name or version. Data type: String
number	Unique number that identifies the application service. Data type: String
population_method	Required. Identifies the population method and its accompanying property to identify the content for population. Only one accompanying object is valid per type. Data type: Object
population_method.group_id	Group ID of the CMDB group configured with the cmdb_group population type. Data type: "String" `"population_method": { "group_id": "String", "type": "cmdb_group" }` Associated population type: cmdb_group
population_method.levels	Number of levels to use in building the service. If the level value is

JSON object properties (continued)

Name	Description
	not provided, the system checks the sys_property for the value. If svc.manual.convert.levels.default_value is not populated, a default value of 3 is used. Data type: Number `"population_method": { "levels": Number, "type": "dynamic_service" }` Associated population type: dynamic_service Default: 3 if no level value is set for the sys_property
population_method.service_candidate	Unique identifier of the service candidate. Data type: String `"population_method": { "service_candidate": "String", "type": "tag_based_service_family" }` Associated population type: tag_based_service_family
population_method.service_relations	List of objects containing hierarchy data for the CIs within the application service. All CIs form pairs with a parent and child CI. The top-level CI, referred to as the entry point of an application service, does not have a parent CI. Data type: Array `"population_method": { "service_relations":[ { "child": "String", "parent": "String" } ],`

JSON object properties (continued)

Name	Description
	<pre>"type": "service_hierarchy" }</pre> Associated population type: service_hierarchy
population_method.service_relations.child	Name of a child CI related to the CI. Data type: String
population_method.service_relations.parent	Name of a parent CI related to the CI. Data type: String
population_method.tags	List of objects containing tags to associate with the CI. This information is located in the Key Values [cmdb_key_value] table. Data type: Array <pre>"population_method": { "tags": [{ "tag": "String", "value": "String" }], "type": "tag_list" }</pre> Associated population type: tag_list
population_method.tags.tag	Tag name. Data type: String
population_method.tags.value	Tag value. Data type: String
population_method.type	Required. Population type to add to the application service. Data type: Object Valid values: • cmdb_group • service_hierarchy • dynamic_service

JSON object properties (continued)

Name	Description
	• tag_list • tag_based_service_family
sys_id	Sys_id of the application service listed in the Application Service [cmdb_ci_service_auto] table. Data type: String

Example on Mac OS

```
snc service-graph app-service
 populate --data '{"name":"Test
 Register","environment":"Test","version":"1.0","popula
tion_method":{"group_id":"String","type":"cmdb_group"
}}'
```

Example on Windows OS

```
snc service-graph app-service
 populate --data "{"name":"Test
 Register","environment":"Test","version":"1.0","popula
tion_method":{"group_id":"String","type":"cmdb_group"
}}"
```

Return value

```
{
  "result": {
  "status": "success"
  }
}
```

Register Application Service

Creates an application service, tags and constructs upstream relationships such as business applications, business service offerings, and other application services.

Command group:

- Parent group: service-graph
- Child group: app-service

Roles required

app_service_admin

If using a service mapping related service, the user must have the required roles for that service mapping related service.

Command structure for Mac OS

```
snc service-graph app-service register --data '{JSON}'
```

Command structure for Windows OS

```
snc service-graph app-service register --data "{JSON}"
```

Arguments

This command passes a JSON object using the *data* parameter.

The following properties for identifying a CI take precedence as follows:

1. sys_id – If sys_id, the system only uses the sys_id and ignores any additional values.
2. number – If provided without the sys_id, the system only uses the number and ignores any additional values.
3. <IRE field name> – The system only uses these values if the sys_id or number are not provided.

JSON object properties

Name	Description
<IRE field name>	One or more IRE fields identifying the application service. For example, name or version. Data type: String
number	Unique number that identifies the application service. Data type: String
relationships	Upstream relationships categorized by type. Data type: Object `"relationships": { "business_app": [Array], "business_service_offering": [Array], "parent_app_service": [Array], "technical_service_offering": [Array] }` Maximum number of relationships is 25.
relationships.business_app	List of objects representing Business Application relationship types. These values can be defined using one of the following items as key-value pairs. • <IRE field name> • number • sys_id

JSON object properties (continued)

Name	Description
	Data type: Array
relationships.business_service_offering	List of objects representing Business Service Offering relationship types. These values can be defined using the following items as key-value pairs. • <IRE field name> • number • sys_id Data type: Array
relationships.parent_app_service	List of objects representing Application Service relationship types. These values can be defined using the following items as key-value pairs. • <IRE field name> • number • sys_id Data type: Array
relationships.technical_service_offering	List of objects representing Technical Service Offering relationship types. These values can be defined using the following items as key-value pairs. • <IRE field name> • number • sys_id Data type: Array
sys_id	Sys_id of the application service listed in the Application Service [cmdb_ci_service_auto] table. Data type: String
tags	List of objects containing tag definitions as key-value pairs. `"tags": [ { "key": "String", "value": "String" }]`

JSON object properties (continued)

Name	Description
	Data type: Array
tags.key	Tag category name. Data type: String
tags.value	Tag value. Data type: String

Example on Mac OS

```
snc service-graph app-service
 register --data '{"name":"Test
 Register","environment":"Test","version":"1.0","number
":"
 SNSVC0001014","relationships":{"business_applicatio
n":[{"sys_id":"0250a94040697410f87713b656474250"},{"num
ber":"APM0001002"},{"name":"Test Biz
 App1"}],"business_service_offering":[{"sys_id":"ed32e9
8040697410f87713b656474259"}],"technical_service_offeri
ng":[{"sys_id":"80e12d8040697410f87713b65647421c"},{"nu
mber":"BSN0001005"},{"name":"Tech Service
 Offering2"}],"parent_app_service":[{"sys_id":"a2f06180
40697410f87713b656474255"}]},"tags":[{"key":"key1","val
ue":"value1"},{"key":"key2","value":"value2"}]}'
```

Example on Windows OS

```
snc service-graph app-service
 register --data "{"name":"Test
 Register","environment":"Test","version":"1.0","number
":"
 SNSVC0001014","relationships":{"business_applicatio
n":[{"sys_id":"0250a94040697410f87713b656474250"},{"num
ber":"APM0001002"},{"name":"Test Biz
 App1"}],"business_service_offering":[{"sys_id":"ed32e9
8040697410f87713b656474259"}],"technical_service_offeri
ng":[{"sys_id":"80e12d8040697410f87713b65647421c"},{"nu
mber":"BSN0001005"},{"name":"Tech Service
 Offering2"}],"parent_app_service":[{"sys_id":"a2f06180
40697410f87713b656474255"}]},"tags":[{"key":"key1","val
ue":"value1"},{"key":"key2","value":"value2"}]}"
```

Return value

```
{
  "result": {
    "app_service": {
      "sys_id": "99b2a54040697410f87713b6564742ad",
      "name": "Test Register",
      "number": "SNSVC0001014"
    },
    "message": "Service registered successfully",
```

```
    "status": "INSERT"
  }
}
```

Update Application Service

Updates an existing application service provided and creates tags for the given application service.

Command group:

- Parent group: service-graph
- Child group: app-service

Roles required

app_service_admin

If using a service mapping related service, the user must have the required roles for that service mapping related service.

Command structure for Mac OS

```
snc service-graph app-service update --data '{JSON}'
```

Command structure for Windows OS

```
snc service-graph app-service update --data "{JSON}"
```

Arguments

This command passes a JSON object using the *data* parameter.

The following properties for identifying a CI take precedence as follows:

1. sys_id – If sys_id, the system only uses the sys_id and ignores any additional values.
2. number – If provided without the sys_id, the system only uses the number and ignores any additional values.
3. <IRE field name> – The system only uses these values if the sys_id or number are not provided.

JSON object properties

Name	Description
<fields or tags to update>	Use key-value pairs to identify each field or tag to be updated. Only basic information can be updated, no upstream relationships can be updated. Data type: String
<IRE field name>	One or more IRE fields identifying the application service. For example, name or version. Data type: String
number	Unique number that identifies the application service.

JSON object properties (continued)

Name	Description
	Data type: String
sys_id	Sys_id of the application service listed in the Service Instance [cmdb_ci_service_auto]. Data type: String

Example on Mac OS

```
snc service-graph app-service update --data '{"name":
 "Test Register", "version": "2.0"}'
```

Example on Windows OS

```
snc service-graph app-service update --data "{"name":
 "Test Register", "version": "2.0"}"
```

Return value

```
{
  "result": {
    "sys_id": "99b2a5404069741 0f87713b6564742ad",
    "name": "Test Register",
    "number": "SNSVC0001014",
    "version": "2.0"
  }
}
```

Update Application Service State

Changes the application service lifecycle state to activate, deactivate, or retire.

Command group:

- Parent group: service-graph
- Child group: app-service

Roles required

app_service_admin

If using a service mapping related service, the user must have the required roles for that service mapping related service.

Command structure for Mac OS

```
snc service-graph app-service update-state --data
 '{JSON}'
```

Command structure for Windows OS

```
snc service-graph app-service update-state --data
 "{JSON}"
```

Arguments

This command passes a JSON object using the *data* parameter.

The following properties for identifying a CI take precedence as follows:

1. sys_id – If sys_id, the system only uses the sys_id and ignores any additional values.
2. number – If provided without the sys_id, the system only uses the number and ignores any additional values.
3. <IRE field name> – The system only uses these values if the sys_id or number are not provided.

JSON object properties

Name	Description
<IRE field name>	One or more IRE fields identifying the application service. For example, name or version. Data type: String
number	Unique number that identifies the application service. Data type: String
state	Required. Lifecycle state of the application service. These values are updated in the Service Instance [cmdb_ci_service_auto] table. Valid values: • ACTIVATE – Life cycle is operational and in use. ○ operational_status=Operational ○ life_cycle_stage=Operational ○ life_cycle_stage_status=In Use • DEACTIVATE – Life cycle is not operational and is in the design stage. ○ operational_status=Non-Operational ○ life_cycle_stage=Design ○ life_cycle_stage_status=Build • RETIRE – End of life. ○ operational_status=Retired ○ life_cycle_stage=End Of Life ○ life_cycle_stage_status=Retired Data type: String
sys_id	Sys_id of the application service listed in the Service Instance [cmdb_ci_service_auto]. Data type: String

Example on Mac OS

```
snc service-graph app-service
 update-state --data '{"name":"Test
 Register","environment":"Test","version":"1.0"}'
```

Example on Windows OS

```
snc service-graph app-service
 update-state --data "{"name":"Test
 Register","environment":"Test","version":"1.0"}"
```

Return value

```
{
  "result": {
  "status": "success"
  }
}
```

App Engine for ERP with the ServiceNow AI Platform

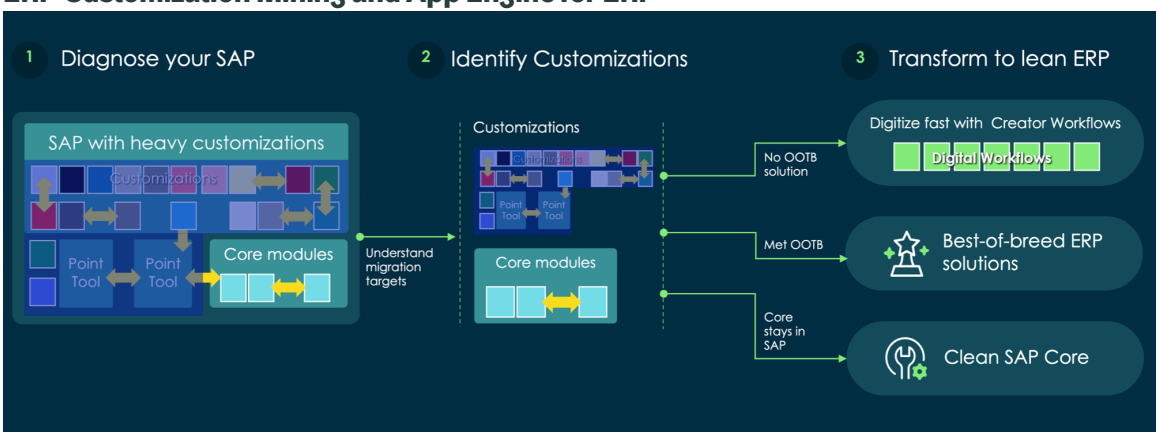
Read and update your legacy ERP (Enterprise Resource Planning) system using ERP Data Hub and identify customizations with ERP Customization Mining (ERP-CM). You can then use additional ServiceNow AI Platform apps, such as App Engine Studio (AES), to organize and replatform ERP data.

Replatforming is the process of scanning legacy ERP system code to find potential candidates to move onto your ServiceNow AI Platform instance as new apps. You can use data from the ERP system as a source for apps built on the ServiceNow AI Platform, improving performance, enhancing security, and reducing maintenance.

Combined benefits of integrating ERP Data Hub and ERP-CM to replatform apps

Legacy systems of records, such as SAP, can have old, complex custom code and data that require more time and effort to move to newer versions of the system of record. Use ERP Data Hub and ERP Customization Mining (ERP-CM) to scan the system of record to find and replace custom code with digitized workflows, resulting in a clean ERP core.

ERP Customization Mining and App Engine for ERP



The replatforming of legacy code enables innovation on top of the system of record without knowledge of the legacy system. Database administrators and developers are then relieved of time-consuming efforts to create database views or endpoints in the system of record, freeing them to work on other projects, such as migration.

Replatformed data is immediately available, mirrored in easy-to-manage tables and apps. Users no longer need to request information from database administrators, which can take weeks. Replatformed apps use the ERP system of record as the live data source.

Workflow for ERP Data Hub and ERP-CM

Using ERP Data Hub and ERP-CM together enables Solutions Integration consultants to implement the following workflow:

1. Have your administrator use the Connections and Credentials app to configure credentials to connect ERP Data Hub to the ERP system of record. For more information, see Connections and Credentials.
2. Create an ERP system in ERP Data Hub using the connection and credentials alias that you configured. For more information, see Create an ERP system in ERP Data Hub.
3. In ERP Data Hub, build your ERP systems, ERP models, and tables in a development instance.
   a. Create or clone an ERP model that scans the specified ERP module in the system of record for available tables and fields. Note the tables and fields in the ERP model for use in extraction and remote tables. For more information, see Clone an ERP model in ERP Data Hub.
   b. Add new tables, fields, and table joins to include additional data in the ERP model. For more information, see the following topics:
      - Managing how models read and update the ERP system
      - Add joins between ERP tables
   c. Create update operation using a BAPI (Business Application Programming Interface) to update data on the ERP system as needed.
   d. Build and customize remote tables to make them available for use as a data source, such as when building apps. Remote tables get their records from running an associated script against an external data source. For more information, see:
      - View and edit ERP remote table details with ERP Data Hub
      - Customize fields for an ERP remote table in ERP Data Hub
      - Query a remote table using ERP Data Hub
   e. Work with ETL extraction tables to regularly scan the system of record and extract data to a staging table. Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.
      - Create as many separate extraction tables as needed, such as one for each supported country. For more information, see Extracting and transforming data in ERP Data Hub.
      - You must first create the table transform map that connects the source table (on the system of record) to a Glide table on the ServiceNow AI Platform. For more information on creating table transform maps, see Create a transform map.
      - Extraction processes are configured in the ServiceNow app that uses them. For example, Workflow Studio.
      - After the extraction process is run, use import sets to map imported data into ServiceNow AI Platform tables. For more information, see Import sets.
4. Identify legacy ERP system customizations to modernize and replatform with ERP-CM.
5. Move the ERP Data Hub systems, ERP models, and tables to a production environment when they are ready. For more information, see Managing ERP development pipelines in ERP Data Hub.

a. Meet with a customer and agree to run an analysis with ERP Customization Mining on their ERP system of record.

b. Connect the customer ERP system of record to the ServiceNow instance using ERP Data Hub.

> **Note:**
> Most customers have their own instance.

ERP-CM uses the system connections configured in ERP Data Hub. For more information, see Working with ERP systems in ERP Data Hub.

c. Use ERP Data Hub to build ERP models from fields on the available remote tables as specified in previous workflow steps.

d. Run ERP Customization Mining to find candidates. Candidates are custom code in the system of record that you can replace with ServiceNow apps. For more information, see Browse an overview of candidates in ERP-CM.

e. Choose candidates to replatform. For more information, see Save potential candidates to replatform.

f. Use the candidate details in ERP-CM as a central place to enter comments and save attachments relating to the candidate. For more information, see View and work with candidate details in ERP-CM.

g. In the candidate details, identify any similar candidates that you could combine into a single replatformed app. For more information, see How ERP Customization Mining determines candidate score and potential.

h. Return to ERP Data Hub to continue building data models with remote tables and extraction tables. Ensure that all the necessary data is available in the ServiceNow AI Platform. For more information, see Building and managing ERP models to work with ERP data.

6. Use the ERP data that is now available as the data source when building apps on the ServiceNow AI Platform, such as:
   - App Engine Studio: For more information, see Create a data model for your application.
   - Flows in Workflow Studio: For more information, see Configuring flows.
   - Table Builder: For more information, see Data in Table Builder.
   - UI Builder: For more information, see Dynamically expose data in UI Builder pages (advanced feature).
   - Workflow Studio: For more information, see Configuring flows.
   - Workspace Builder: For more information, see Configure a record page for a workspace in Workspace Builder.
7. Measure and monitor the performance of the new app using applicable metrics and parameters with your preferred analytic tools.

Requirements for integrating ERP Data Hub and ERP-CM

1. Requirements for installing ERP Data Hub include the following:
   - A valid license
   - ServiceNow AI Platform plugins
   - MID server configuration

- Spoke integration
- SAP configuration

For more information, see Requirements for installing ERP Data Hub.

2. Requirements for installing ERP-CM include the following:
   - License and entitlement. For more information, see Install ERP Data Hub.
   - ServiceNow AI Platform plugins. For more information, see Install ERP Data Hub.
   - SAP configuration. For more information, see Configure SAP for ERP-CM.

Get started with ERP Data Hub and ERP-CM

Get started with ERP Data Hub and ERP-CM by completing these tasks:

1. Install and configure ERP Data Hub. For more information, see Configuring ERP Data Hub.
2. Install and configure ERP-CM. For more information, see Configuring ERP Customization Mining.
3. Install and configure any additional ServiceNow AI Platform apps and builders that consume ERP data, such as the following:
   - Installing App Engine Studio
   - Configuring Flows in Workflow Studio
   - Exploring Table Builder
   - Getting started with Playbooks
   - UI Builder quick start
   - Workflow Studio
   - Workspace Builder

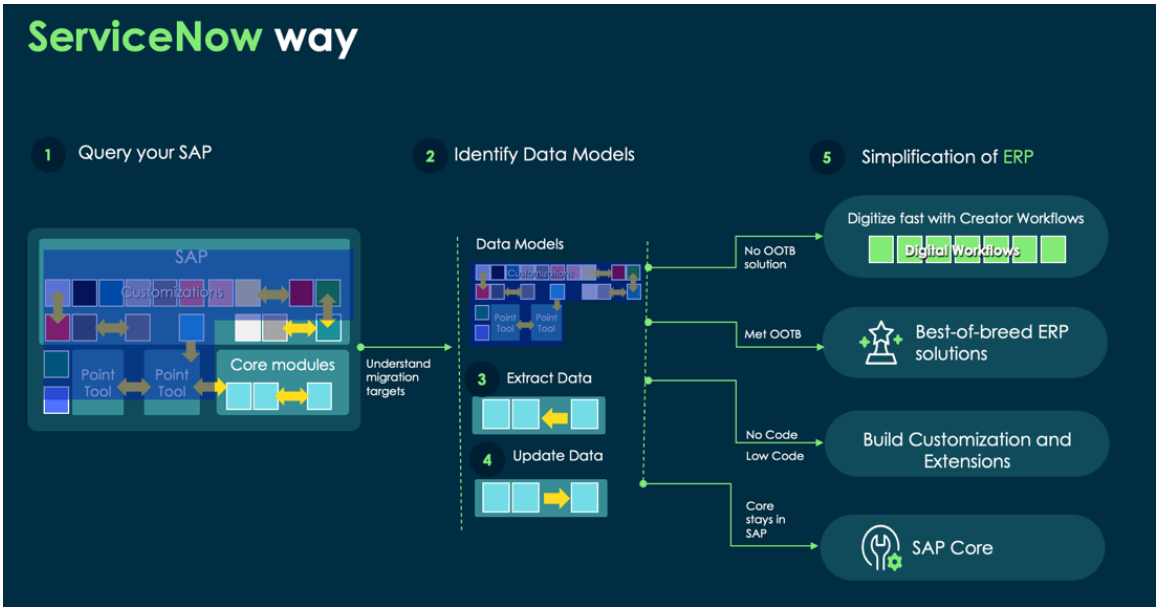
ERP Data Hub

ERP Data Hub (Enterprise Resource Planning) is a ServiceNow scoped app that enables you to retrieve and update ERP data from the system of record, such as SAP.

Legacy systems of records, such as SAP, can have old, complex custom code and data that require more time and effort to move to newer versions of the system of record. Use ERP Data Hub and ERP Customization Mining (ERP-CM) to scan the system of record to find and replace custom code with digitized workflows, resulting in a clean ERP core.

The replatforming of legacy code enables innovation on top of the system of record without knowledge of the legacy system. Database administrators and developers are then relieved of time-consuming efforts to create database views or endpoints in the system of record, freeing them to work on other projects, such as migration.

ERP Data Hub



Use ERP models as the data foundation for ERP apps. These models include ERP system data in remote tables and extraction tables, and enable you to read and update the ERP system. After you create ERP models, you can use the extracted and transformed data to build apps that access the ERP models, for example in App Engine Studio or Creator Studio, or use the ERP data in flows in Workflow Studio.

Note:
For information about new and updated features in the Xanadu release, see ERP Canvas release notes.

Explore

Learn about ERP Data Hub concepts and features.

Configure

Install and configure ERP Data Hub remote tables and connections.

Work

Work in ERP Data Hub to build ERP models using remote tables and extraction tables.

Reference

Get details about ERP Data Hub components, such tables, and terminology.

Learning resources for ERP Data Hub

Learn more about ERP Data Hub	ServiceNow resources
ERP Data Hub has a several training and learning resources for you to get started. (Some of these resources require logging in to the ServiceNow Learning site.)	Partner Essentials: Clean Core for App Engine ERP Clean Core with App Engine Overview ERP modernization: Working toward a clean core

Request ERP Data Hub on the store

Visit the ServiceNow Store website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Before you can use ERP Data Hub, you must first download the application from the ServiceNow Store. After you've completed the download, you may need to clear your local cache before ERP Data Hub appears on your instance.

Exploring ERP Data Hub

ERP Data Hub enables you to connect to the ERP (Enterprise Resource Planning) system to send updates and extract data to remote tables and extraction tables for use on the ServiceNow AI Platform.

Overview of ERP Data Hub

ERP Data Hub functions as a platform within ServiceNow, offering a unified data model for ERP systems. ERP Data Hub enables you to manage tables that contain both standard and custom fields grouped within ERP models. You can send updates to and extract data from ERP tables on the system of record and store it in a remote table or an extraction table, depending on the size of data sets and refresh needs.

- Remote tables get their records from running an associated script against an external data source.
- Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.

In addition to retrieving data, ERP Data Hub can also update the ERP system using a BAPI (Business Application Programming Interface) or OData and an HTTP connection. (OData option available starting in Xanadu Store Release 2.)

The unified data model of the ServiceNow AI Platform ensures seamless integration of ERP data into the ServiceNow AI Platform. ERP Data Hub streamlines ERP data management, making it accessible and actionable within the ServiceNow AI Platform and ServiceNow instances.

Note:
ERP Data Hub doesn't replicate data into the ServiceNow AI Platform. It mirrors data that lives in the ERP system of record, and remains protected there.

ERP Data Hub workflow

1. Have your administrator use the Connections and Credentials app to configure credentials to connect to the ERP system of record. For more information, see Configure the ERP Data Hub credentials and connection.
2. Create an ERP system in ERP Data Hub using the connection and credentials alias that you configured. For more information, see Create an ERP system in ERP Data Hub.

   Note:
   The rest of the workflow steps are in ERP Data Hub. Build your ERP systems, ERP models, and tables in a development instance, and then promote them to a production instance when you're ready. For more information, see Managing app deployments using Pipelines and Deployments.

3. Clone or create an ERP model that scans the specified ERP module in the system of record for available tables and fields. Note the tables and fields in the ERP model for use in extraction and remote tables, as well as for mapping parameters to read and update the system of record. For more information, see Clone an ERP model in ERP Data Hub.
4. Create read and update operations to connect to the ERP system by adding tables, mapping fields, and building table joins to include additional data in the ERP model. For more information, see the following topics:
   - Managing how models read and update the ERP system
   - Add joins between ERP tables
5. Work with remote tables to make them available for use as a data source, such as when building apps in App Engine Studio. Remote tables get their records from running an associated script against an external data source. For more information, see the following topics:
   - View and edit ERP remote table details with ERP Data Hub
   - Customize fields for an ERP remote table in ERP Data Hub
   - Query a remote table using ERP Data Hub
6. Work with ETL extraction tables to regularly scan the system of record and extract data to a staging table. Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.
   - Create as many separate extraction tables as needed, such as one for each supported country. For more information, see Extracting and transforming data in ERP Data Hub.
   - You must first create the table transform map that connects the source table (on the system of record) to a Glide table on the ServiceNow AI Platform. For more information on creating table transform maps, see Create a transform map.
   - Extraction processes are configured in the ServiceNow app that uses them, for example, Workflow Studio.
   - After the extraction process is run, use import sets to map imported data into ServiceNow AI Platform tables. For more information, see Import sets.

7. Build flows in Workflow Studio to specify details for when you query or update the ERP (Enterprise Resource Planning) system. For more information, see Building flows to read or update the ERP system.
8. Move the ERP systems, ERP models, tables, operations, and flows to a production environment when they're ready. For more information, see Managing ERP development pipelines in ERP Data Hub.
9. Use the ERP data as the data source when building apps on the ServiceNow AI Platform using:
   - App Engine Studio: For more information, see Create a data model for your application.
   - Flows in Workflow Studio: For more information, see Configuring flows.
   - Playbooks in Workflow Studio: For more information, see Getting started with Process Automation.
   - Table Builder: For more information, see Data in Table Builder.
   - UI Builder: For more information, see Dynamically expose data in UI Builder pages (advanced feature).
   - Workspace Builder: For more information, see Configure a record page for a workspace in Workspace Builder.

Benefits of ERP Data Hub

ERP Data Hub key benefits

Benefit	Feature	Role
Configure connections to the system of record	Working with ERP systems in ERP Data Hub	sn_erp_integration.erp_admin
Build ERP models to create read and update operations and organize mirrored ERP data	Building and managing ERP models to work with ERP data	sn_erp_integration.erp_admin
Work with and query remote tables to view ERP data on the system of record	Using ERP remote tables in ERP Data Hub	• To modify remote tables: sn_erp_integration.erp_admin • To read remote tables: sn_erp_integration.erp_user
Access standard remote tables	Standard remote tables for ERP Data Hub	• sn_erp_integration.erp_user • Additional, table-specific roles are required. For more information, see ERP Data Hub roles.
Configure extraction tables to regularly pull custom data from the ERP system	Extracting and transforming data in ERP Data Hub	sn_erp_integration.erp_admin

Additional resources for ERP Data Hub

Learn more about ERP Data Hub	ServiceNow resources
ERP Data Hub is a ServiceNow app that enables you to simplify the use of ERP data from the system of record, such as SAP.	App Engine for ERP Overview on ServiceNow University **Note:** You must log in to ServiceNow University to access this resource. ServiceNow Community site Video: Unlock the full potential of your ERP system

Identifying ERP candidates to replatform with ERP Data Hub and ERP-CM

ERP Data Hub enables you to connect to your ERP (Enterprise Resource Planning) system of record, and to organize its data.

Replatforming legacy ERP apps onto the ServiceNow AI Platform

Replatforming is the process of scanning legacy ERP system code to find potential candidates to move onto your ServiceNow AI Platform instance as new apps. You can use data from the ERP system as a source for apps built on the ServiceNow AI Platform, improving performance, enhancing security, and reducing maintenance.

Using ERP Data Hub, you can access standard fields for remote tables and ERP extraction tables, while ERP Customization Mining (ERP-CM) enables you to find good candidates for replatforming from the system of record to the ServiceNow AI Platform.

ERP-CM suggests candidates and possible next steps, such as updating remote tables and extraction tables to access ERP data. Remote tables send data to the ServiceNow instance as an attachment, which is then analyzed using AI/ML to identify similar candidates to replatform.

Using remote tables and extraction tables with ERP Data Hub and ERP-CM

You can use a combination of remote tables and extraction tables to retrieve data from your legacy ERP system.

- Remote tables get their records from running an associated script against an external data source.
- Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.

Remote tables describe the schema for the data that you want to retrieve from an external source, such as the system of record. Use remote tables to connect the ServiceNow AI Platform to third-party sources, or to another instance, so that you can retrieve external data and optionally cache it in the memory. You can view external data in lists or forms and process it with standard Glide scripts. You can also group, sort, aggregate, and filter the data just like you would for standard internal tables.

By using a remote table, you can retrieve the data from external sources or from another instance with REST or SOAP services. The external data lives in the memory in read-only mode, which makes the data temporary, or transient, within the ServiceNow AI Platform. You can then view and manipulate the external data without importing or storing it. For more information, see Remote tables.

Use an extraction table to work with large amounts of ERP data. ERP extraction tables regularly save data to a local transform table on the ServiceNow AI Platform, which you can then process and use as the data foundation of a replatformed app.

ERP Data Hub custom field example

ERP Data Hub helps you identify custom ERP (Enterprise Resource Planning) apps and fields in the system of record to access their data on the ServiceNow AI Platform. The ERP system can have both standard and custom fields that are accessed by ERP Data Hub.

Example of ERP custom data replatforming

In this example, a farmer grows grain to sell. The farmer has entries in a table for standard values, such as weight and sales price.

However, the sale price of grain depends on the moisture content of the grain. If it rains the day before grain is harvested, the farmer must adjust the sale price to reflect the moisture content. Thus, in addition to standard fields like **Date** and **Weight**, the legacy table that tracks the grain harvest on the ERP system must have a custom **Moisture %** field.

The farmer can use ERP Data Hub to connect to the legacy system of record and identify the table that contains the custom field. Then they can create an ERP model in ERP Data Hub with a remote table or an extraction table that contains the **Moisture %** field.

After they have the ERP model with custom data, they can use App Engine Studio or other ServiceNow products to quickly build an app that tracks their grain sales. The data that the grain sale app consumes still lives on the legacy system of record.

ERP Data Hub and security

In addition to role-based security and access control, ERP Data Hub protects personally identifiable data in other ways.

Personally identifiable data is secured with ERP Data Hub and ERP-CM in several ways.

- You can customize ERP models and remote tables to exclude personal data in a specified field, such as email address.
- All remote tables are secured using access control rules (ACLs). If you have a remote table that contains sensitive data, use ACLs to restrict that table from ServiceNow users. For more information, see ServiceNow® access control.

Configuring ERP Data Hub

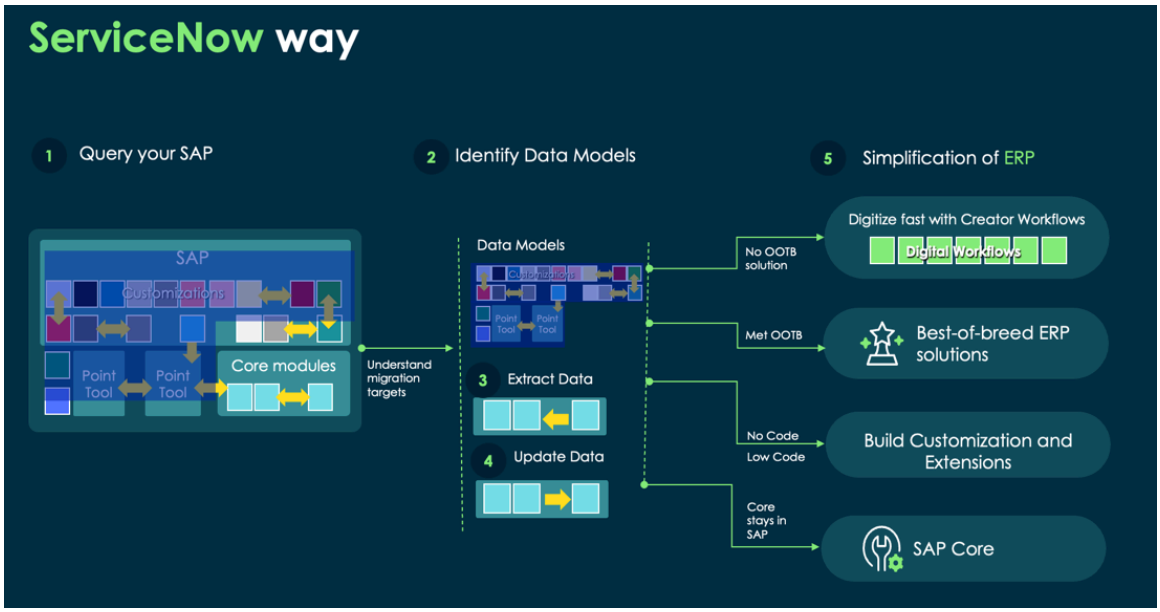
Install ERP Data Hub to configure connections to ERP (Enterprise Resource Planning) systems of record, as well as other ServiceNow products, such as ERP Customization Mining, Procurement for Field Service Management, and Process Mining.

ERP Data Hub uses basic authentication to connect a ServiceNow instance with an instance on the system of record (such as SAP).

After you configure a connection, you can read and update the system of record with ERP Data Hub using ERP models, and create remote tables and extraction tables.

Additionally, you can use ERP Customization Mining (ERP-CM) to identify legacy applications that are good candidates for replatforming, making their data immediately available on the ServiceNow AI Platform. For more information, see ERP Customization Mining (ERP-CM).

ERP Data Hub



Connecting to multiple instances

The number of ERP connections you can have per ServiceNow instance depends on your license. If you have the ERP-CM license, you get one connection per instance.

Requirements for installing ERP Data Hub

Before you install ERP Data Hub, you must complete several configurations, on both the ERP (Enterprise Resource Planning) system and on the ServiceNow AI Platform.

Licensing

You must have a license and get entitlement to ERP Data Hub for installation. For more information, see Licensing.

ServiceNow AI Platform plugins

The following plugins and servers are required for installing ERP Data Hub:

- Integration Hub plugin (for more information, see Request Integration Hub)
- Remote Tables plugin (for more information, see Activate the remote tables plugin)
- MID Server: The MID Server must be configured to install ERP Data Hub. See the following section for more information.

MID Server requirements

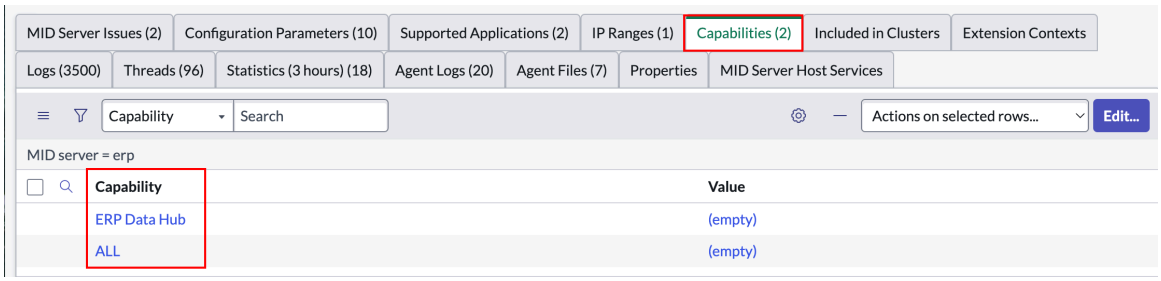
Communicating with the system of record through a MID Server requires a valid connection and credential alias.

Note:
The credentials you specify for the ERP Data Hub connection must match the service user credentials in the system of record.

ERP Data Hub and ERP-CM currently support ECC (minimum SAP Netweaver 7.31) and S/4HANA SAP systems.

Find your MID Server on the ServiceNow AI Platform in **All** > **MID Server** > **Servers**. Select it and add the following:

1. In the **Supported applications** tab, edit the list to include ERP Data Hub.
2. In the **Capabilities** tab, in addition to **ALL**, edit the list to include ERP Data Hub.



For more information, see MID Server.

Assign the mid_user user the sn_erp_integration.erp_mid_server role. The role enables the MID Server to use the ImportSet API to send data to the ServiceNow instance.

1. Navigate to **All** > **User Administration** > **Users**
2. Find and select the **mid_user**.
3. Select the **Roles** tab.
4. Select **Edit**.
5. Find and assign the **sn_erp_integration.erp_mid_server** role.

Required SAP authorizations in development system

For the credentials, use an SAP service type user account in your system of record that requires the following SAP authorization objects:

- S_RFC with Activity = 16 (Execute) for the following Function Modules:
  - RFCPING
  - BAPI_MONITOR_GETLIST
  - RFC_METADATA_GET

- RFC_GET_FUNCTION_INTERFACE
- RFC_READ_TABLE
- DDIF_FIELDINFO_GET

- S_TABU_NAM with Activity 03 (Display) for the following tables:
  - DD01L
  - DD02L
  - DD02T
  - DD03L
  - DD04L
  - DD04T
  - DD12L
  - DD17S
  - DD40L
- S_DEVELOP with Activity 03 (Display)

Required SAP authorizations in production and test system

For the credentials, use an SAP service type user account in your system of record that requires the following SAP authorization objects:

- S_RFC with Activity = 16 (Execute) for the following Function Modules:
  - RFCPING
  - BAPI_TRANSACTION_COMMIT
  - BAPI_TRANSACTION_ROLLBACK
  - RFC_METADATA_GET
  - RFC_GET_FUNCTION_INTERFACE
  - RFC_READ_TABLE
- S_TABU_NAM with Activity 03 (Display) for the tables you have used in models.

Spoke integration

For required spokes, contact the admin of your SAP ECC RFC account to obtain the following SAP proprietary JARs and other required files:

- Create a record called "SAP Jco Dll" and attach `sapjco3.dll`. Use this file if your MID Server is installed on a Windows server.
- Create a record called "SAP Jco so" and attach `libsapjco3.so`. Use this file if your MID Server is installed on a Linux server.
- Create a record called "SAP Eco Jar" and attach `sapjco3.jar`. Use this file for both Windows and Linux.

> **Note:**
> Create the records at **All** > **MID Server** > **JAR Files**. For more information about creating JAR files and adding attachments, see Synchronize a JAR file to MID Servers.

The MID Server restarts multiple times during this process.

For more information on spoke integration, see SAP ECC RFC Spoke.

SAP configuration

You must configure the JCO connector before you install ERP Data Hub. See the SAP documentation for more information.

You must add the DuckDB JDBC jar as a MID Server jar. For more information and to download, see DuckDB JDBC Driver 1.1.1 on the Maven web site.

Additionally, you need the following on your ERP system:

- SAP ECC (minimum SAP Netweaver 7.31)
- SAP S/4HANA (all versions supported)
- SAP Java Connector

Install ERP Data Hub

Install the ERP Data Hub (Enterprise Resource Planning) application (sn_erp_integration) if you have the admin role from the ServiceNow Store.

Before you begin

For a complete list of prerequisites for installing ERP Data Hub, including licensing information, see Requirements for installing ERP Data Hub.

Role required: admin

Procedure

1. Go to the ServiceNow Store.
2. In search type, `ERP Data Hub`.
3. Select the **ERP Data Hub** tile.
4. Select **Get**.
5. Log in using your ServiceNow credentials.
   Federal customers should select **Are you a federal customer?** and follow the instructions.
   If you don't have ServiceNow credentials and aren't a federal customer, select **Are you a customer who doesn't have ServiceNow ID?** and follow the instructions.
6. Check that you have entitlements (or licenses) to the product and dependent applications by viewing **Your Subscription Status**.
7. Select **Request Install**.
   When complete, confirm the installation of ERP Data Hub and select **Close** to return to the ServiceNow Store.

What to do next

After you install ERP Data Hub, you must enable the *sn_erp_integration.enableModelModification* property so users can customize ERP models. After you enable the *sn_erp_integration.enableModelModification* property, ERP Data Hub retrieves all tables and BAPIs (Business Application Programming Interface) to use when managing models. System properties are maintained in the System Property table [sys_properties], which you can access using the module navigator, or by directly typing `sys_properties.list` in the Navigator Filter.

Run Guided Setup for ERP Data Hub

Run the Guided Setup to configure ERP Data Hub.

Before you begin

You must first download and install ERP Data Hub from the ServiceNow Store. For more information, see Install ERP Data Hub.

Role required: sn_erp_integration.erp_admin

About this task

For more information on using Guided Setup, see Guided setup.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Guided Setup**.
2. Set up the MID Server in the MID Server setup section.
   a. Select **Create MID user** and create the user account on the ServiceNow instance that will connect to the MID Server.
   For example, you can create mid.user.
   b. Select **Download and install MID** and follow the instructions to download the appropriate MID Server installer archive for the operating system.
   c. Select **Configure MID server files** and follow the instructions to configure the required MID Server files, which are detailed in the Guided Setup.
   For more information, see Install ERP Data Hub.
3. Set up the connection and credentials.
   a. Select **Create credential record for the ERP system** and follow the steps.
   b. Select **Create a connection record for ERP Data Hub** and follow the steps.
   c. Select **Configure connection and capabilities** and complete the steps.
   Alternatively, you can configure without Guided Setup. For more information, see Configure the ERP Data Hub credentials and connection.
4. Create and validate the ERP system.
   a. Select **Create system** and follow the steps.
   b. Select **Validate system** and follow the steps.
   Alternatively, you can configure without Guided Setup. For more information, see Create an ERP system in ERP Data Hub.
5. Configure remote tables and extraction sources.
   a. Select **Configure remote tables** and follow the steps.
   Alternatively, you can configure without Guided Setup. For more information, see View and edit ERP remote table details with ERP Data Hub.
   b. Select **Configure extraction sources** and follow the steps.
   Alternatively, you can configure without Guided Setup. For more information, see Add a new ERP extraction table in ERP Data Hub.

Configure the ERP Data Hub credentials and connection

Connect ERP Data Hub to a system of record (such as SAP) directly or using a load balancer to enable access to the ERP (Enterprise Resource Planning) system. You must select an existing, configured connection when you set up an ERP system.

Before you begin

You must first create the alias for ERP Data Hub. For more information, see Create a Connection & Credential alias.

Role required: admin

About this task

ERP Data Hub and ERP-CM currently support ECC (minimum SAP Netweaver 7.31) and S/4HANA SAP systems.

Note:
The credentials you specify for the ERP Data Hub connection must match the service user credentials in the system of record.

Alternatively, you can run Guided Setup. For more information, see Run Guided Setup for ERP Data Hub.

Procedure

1. Navigate to **All** > **Connections & Credentials** > **Connection & Credential Aliases**.
2. Select the alias that you created for the new connection.
3. Create a connection and credential.
4. On the form, fill in the fields.
   You must specify a connection and login credential to be used simultaneously. That is, the connection you configure uses the defined login credentials for the connection.

   For a description of the field values, see ERP Data Hub connection and credentials field descriptions. If you're creating an HTTP connection, see Create an HTTP(s) connection for field and MID Server details.
5. Set up the ERP Data Hub connection.
   a. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
   b. Open the ERP systems list by selecting the systems icon () in the side panel.
   c. In the **Connection** column, select the **(empty)** cell.

      Important:
      You must select the empty row instead of creating a new row.

   d. In the search box that appears, select the search icon ().
   e. On the Connection & Credential Aliases dialog, select **sn_erp_integration.ERP_Integration**.
   f. Select the selection icon ( ).

ERP Data Hub roles

Administrators assign roles to give team members permission to configure or use ERP Data Hub.

Important:
When you assign ERP Data Hub (Enterprise Resource Planning) roles to a user, you must include the scope. For example, assign the `sn_erp_mining.erp_admin` role, not just `erp_admin`.

For more on assigning roles, see Assign a role to a user.

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

ERP Data Hub roles

Note:
All required roles are the same across development and production instances. However, only sn_erp_integration.erp_user is required to read data in a production instance.

ERP Data Hub roles

Role	Description	Additional access
sn_erp_integration.erp_admin	Grants the user access to updating the application setup.	Contains sn_erp_integration.erp_user.
sn_erp_integration.erp_data_pill	Grants the user read access to all the tables needed for making Remote Tables requests. This role can be combined with any single Remote Tables role. For example, combining sn_erp_integration.erp_data_pill and sn_erp_integration.sap_company_code_user enables the list of SAP Company Codes.	--
sn_erp_integration.erp_mid_server	Grants the user access to use the API to enable the MID Server to send attachments back to the ServiceNow instance.	--
sn_erp_integration.erp_user	Grants the user read access for all remote tables.	Contains sn_erp_integration.erp_data_pill and all Remote Tables roles. • sn_erp_integration.sap_company_code_user • sn_erp_integration.sap_country_user • sn_erp_integration.sap_currency_user • sn_erp_integration.sap_customer_invoice_us • sn_erp_integration.sap_distribution_channel • sn_erp_integration.sap_division_user • sn_erp_integration.sap_language_user

ERP Data Hub roles (continued)

Role	Description	Additional access
		• sn_erp_integration.sap_material_stock_user • sn_erp_integration.sap_purchase_document • sn_erp_integration.sap_purchasing_organiza • sn_erp_integration.sap_sales_customer_user • sn_erp_integration.sap_sales_delivery_user • sn_erp_integration.sap_sales_document_use • sn_erp_integration.sap_sales_organization_u • sn_erp_integration.sap_sales_revenue_recog • sn_erp_integration.sap_vendor_invoice_user • sn_erp_integration.sap_vendor_user

Additional ERP data model roles

If users need access to work with specific ERP data models, such as purchasing or invoices, assign them the following roles and access for ERP Data Hub.

ERP data model-specific roles for ERP Data Hub

Persona	Role	Access
ERP MID Server user	erp_mid_server	• sys_attachments • sn_erp_integration_queue
Customer invoice user	sap_customer_invoice_user	sn_erp_integration_st_sap_customer_invoice
Material stock user	sap_material_stock_user	sn_erp_integration_st_sap_material_stock
Purchase document user	sap_purchase_document_user	sn_erp_integration_st_sap_purchase_documer
Purchasing organization user	sap_purchasing_organization_user	sn_erp_integration_st_sap_purchasing_organiz
Sales customer user	sap_sales_customer_user	sn_erp_integration_st_sap_sales_customer
Sales delivery user	sap_sales_delivery_user	sn_erp_integration_st_sap_sales_delivery
Sales document user	sap_sales_document_user	sn_erp_integration_st_sap_sales_document
Sales organization user	sap_sales_organization_user	sn_erp_integration_st_sap_sales_organization
Sales revenue recognition user	sap_sales_revenue_recognition_user	sn_erp_integration_st_sap_sales_revenue_reco
Vendor invoice user	sap_vendor_invoice_user	sn_erp_integration_st_sap_vendor_invoice
Vendor user	sap_vendor_user	sn_erp_integration_st_sap_vendor

Working with ERP systems in ERP Data Hub

An ERP (Enterprise Resource Planning) system represents a connection to a section of your ERP system of record. For example, sales orders or vendor invoices.

ERP systems organize connections to the system of record

The system plays a crucial role in data synchronization, sharing, and collaboration, enabling seamless integration and operation between the ERP model and the connected ERP system.

ERP Data Hub provides a standard set of ERP models, such as SAP Material Stock and SAP Purchase Document. You can also build new models. For a list of standard ERP models, which you must clone to modify, see Standard ERP models and extraction tables for ERP Data Hub.

Configuring ERP systems and checking connections

ERP systems are configured by ServiceNow admins. The ERP system is set on the extraction table or remote table. ERP Data Hub supports connecting to multiple systems.

ERP Data Hub regularly scans all connected ERP systems for the latest heartbeat, which indicates whether a ping to the ERP system connection is currently successful.

Create an ERP system in ERP Data Hub

Configure an ERP (Enterprise Resource Planning) system in ERP Data Hub to organize your connections to the system of record.

Before you begin

Role required: sn_erp_integration.erp_admin

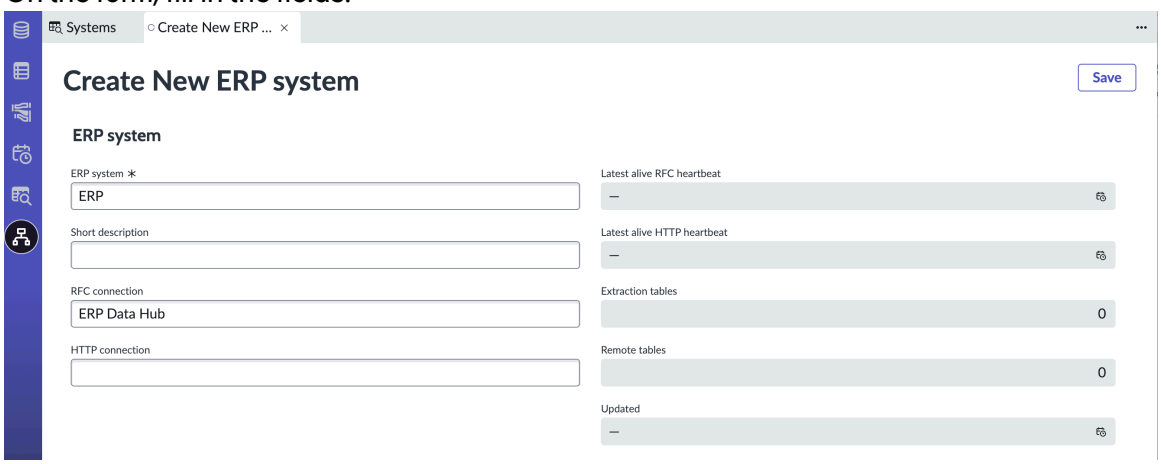
About this task

The ERP system is set on the extraction table or remote table. ERP Data Hub supports connecting to multiple systems.

Alternatively, you can run Guided Setup. For more information, see Run Guided Setup for ERP Data Hub.

Procedure

1. Navigate to **All** > **ERP Data Hub**.
2. Open the ERP systems list by selecting the systems icon ( ) in the side panel.
3. Select **New**.
4. On the form, fill in the fields.



Note:
The HTTP connection option is available starting in Xanadu Store Release 2. You must have an SAP system that is enabled to make an OData connection.

For a description of the field values, see ERP Data Hub new system field descriptions.

5. Select **Save**.

Result

After you create a system, you can view heartbeat and retrieval status on the ERP systems list page. For more information, see View a list of ERP Data Hub systems.

View a list of ERP Data Hub systems

Check the ERP (Enterprise Resource Planning) systems list in ERP Data Hub to view the heartbeats and retrieval status of your ERP systems.

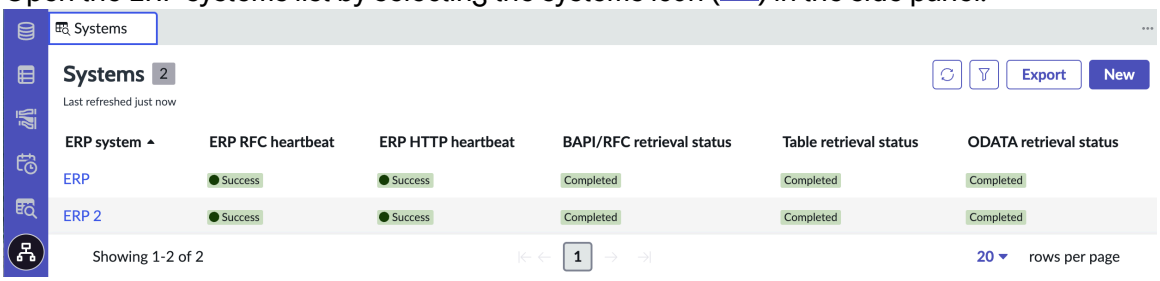
Before you begin

Role required: sn_erp_integration.erp_admin

Note:
Users with the sn_erp_integration.erp_admin role can view the list of ERP systems, but can't create a system. For more information, see Create an ERP system in ERP Data Hub.

Procedure

1. Navigate to **All** > **ERP Data Hub**.
2. Open the ERP systems list by selecting the systems icon ( ) in the side panel.



Note:
The HTTP connection option and ODATA option are available starting in Xanadu Store Release 2.

3. Check the information for each system to confirm that there are no connectivity issues.
   For a description of the field values, see ERP Data Hub system list field descriptions.

View ERP Data Hub system heartbeat information

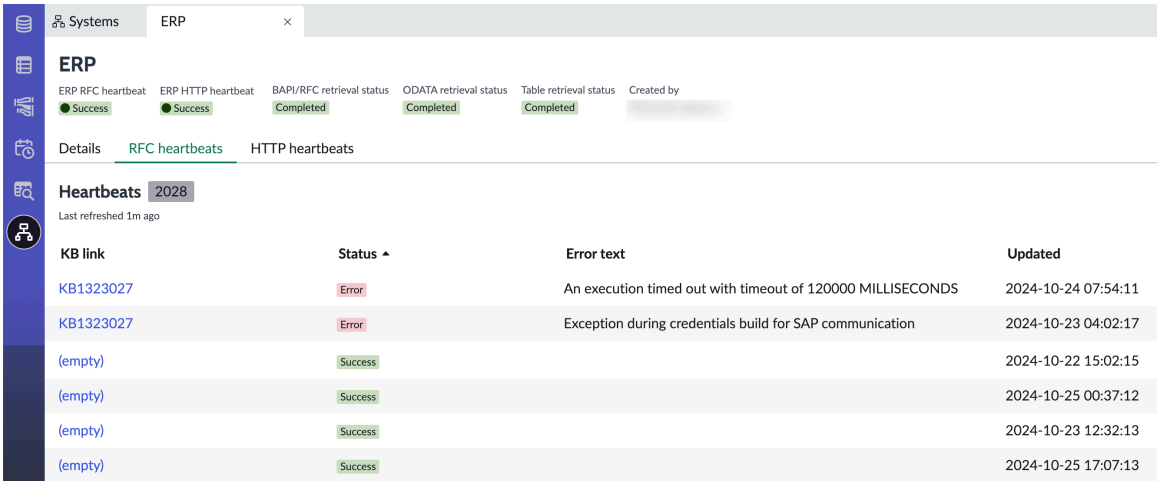
In ERP Data Hub, the heartbeat shows the status, date, and time of connections to the ERP system, along with error information.

Before you begin

Role required: admin and sn_erp_integration.erp_user

Procedure

1. Navigate to **All** > **ERP Data Hub**.
2. Open the ERP systems list by selecting the systems icon ( ) in the side panel.
3. Open a system.
4. Select the **RFC heartbeats** or **HTTP heartbeats** tab.



View information about the heartbeats, including updated date and time, and status. If there's an error, the error text is displayed and a link to a knowledge base article (if available) is provided. For more information, see ERP Data Hub new system field descriptions.

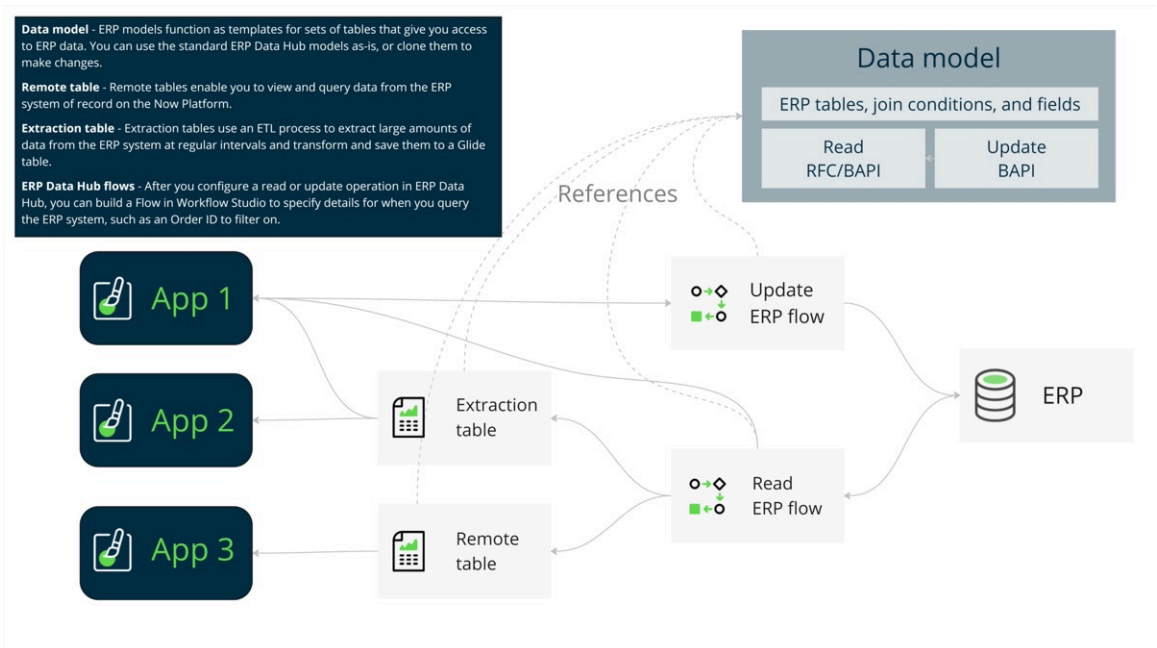
By default, all heartbeat information is kept for one week. To change that setting, go to **All** > **Data Management Policies**, open the sn_erp_integration_log_heartbeat policy, select the **Table Cleanup Rules** tab, select the **Tablename**, and edit the **Age in seconds**.

Using ERP models, extraction tables, and remote tables

Use ERP Data Hub to work with ERP (Enterprise Resource Planning) models, remote tables, and extraction tables to integrate ERP data from the system of record onto the ServiceNow AI Platform.

The workflow of ERP Data Hub generally follows its main sections, each of which you access by selecting the section icon in the contextual side panel.

Data models and ERP data



You can also build flows in Workflow Studio to use retrieved ERP data for processes or tasks outside of ERP Data Hub. For more information, see Building flows to read or update the ERP system.

Build an ERP model using remote tables and extraction tables to organize, load, and transform ERP data, as well as flows to query the ERP system.

Parts of ERP Data Hub

ERP Data Hub section	Description
ERP models	ERP models function as templates for sets of tables that give you access to ERP data. You can use the standard ERP Data Hub models as-is, or clone them to make changes. Manage models to map input and output data for reading and updating the ERP system using wither table read operations or BAPIs (Business Application Programming Interface). For more information, see Building and managing ERP models to work with ERP data.
Flows to query ERP data	Build a flow in Workflow Studio to specify details for querying the ERP system using the parameters specified in the model.
Remote tables	Remote tables enable you to view and query data from the ERP system of record on the ServiceNow AI Platform. For more information, see Using ERP remote tables in ERP Data Hub.
ERP extraction tables	Extraction tables use an ETL process to extract large amounts of data from the ERP system at regular intervals and transform and save them to a Glide table. For more information, see Extracting and transforming data in ERP Data Hub.

Building and managing ERP models to work with ERP data

ERP (Enterprise Resource Planning) models function as templates for sets of tables that give you access to ERP data. Use model management to create read and update operations that access the ERP system with specified inputs and outputs to map fields for use on the ServiceNow AI Platform.

ERP models represent data sets and create a staging area

An ERP model represents the logical structure and organization of data coming from the Enterprise Resource Planning system. ERP models define the entities, attributes, read/update operations, and table join relationships that capture and represent business processes and data elements in the ERP system. When you first open ERP Data Hub, you view a list of the ERP models for your instance.

ERP Data Hub provides a standard set of ERP models, such as SAP Material Stock and SAP Purchase Document. You can also build new models. For a list of standard ERP models, which you must clone to modify, see Standard ERP models and extraction tables for ERP Data Hub.

The ERP model serves as a blueprint for configuring, customizing, and integrating the ERP system to meet your business requirements. An ERP model functions as a staging area that contains all potential fields you can add to remote and extraction tables, and read and update operations. You can then use the tables and queried data as a data source on the ServiceNow AI Platform.

You can also build flows in Workflow Studio to use retrieved ERP data for processes or tasks outside of ERP Data Hub. For more information, see Building flows to read or update the ERP system.

You can view the list of existing ERP models or add a new one to create a custom data set. After you add a remote table, you can manage models to map input and output parameters, update the ERP system using BAPIs (Business Application Programming Interface), and create table joins. You can also create extraction tables that regularly pull large amounts of filtered data from the ERP system. Another option is to use a custom ERP action in Workflow Studio to use queried ERP data in other ServiceNow AI Platform processes.

You can create as many ERP models as needed, though you can't edit their field names.

How ERP models are structured

Each model is linked to ERP tables on the system of record, as well as remote tables and ERP extraction tables on the ServiceNow AI Platform. You can connect the same table to multiple, different ERP models.

ERP models are connected to an ERP system, one ERP system for each model. The connected ERP system:

- Enables access to field and table information.
- Helps coordinate data synchronization, sharing, and collaboration, enabling seamless integration and operation between the ERP model and the ERP system.

ERP models in ERP Data Hub encompass remote tables from the system of record, as well as APIs and ERP extraction tables and read/update operations, to create a holistic data set. For example, you can have one ERP model for sales orders and another for inventory.

Note:

After you install ERP Data Hub, you must enable the *sn_erp_integration.enableModelModification* property so users can customize ERP models. After you enable the *sn_erp_integration.enableModelModification* property, ERP Data Hub retrieves all tables and BAPIs (Business Application Programming Interface) to use when managing models. System properties are maintained in the System Property table [sys_properties], which you can access using the module navigator, or by directly typing `sys_properties.list` in the Navigator Filter.

Managing models to read and update ERP systems

After you create or clone an ERP model, you can specify how ERP Data Hub reads and writes to the ERP system using the ERP model manager page. When managing models, you have the option to use a BAPI, which is a remote procedure SAP function call that's similar to an API.

Each model can have only one read and one update operation defined.

For more information, see Managing how models read and update the ERP system.

View and edit the foundation of ERP models

Create a holistic data set by building ERP (Enterprise Resource Planning) models in ERP Data Hub, which encompasses remote tables and extraction tables from the ERP system, as well as read and update operations.

Before you begin

Your administrator must enable the *sn_erp_integration.enableModelModification* property for you to edit, customize, and clone ERP models and tables. After you enable the *sn_erp_integration.enableModelModification* property, ERP Data Hub retrieves all tables and BAPIs (Business Application Programming Interface) to use when managing models. The property must be configured for either a non-production or production state. System properties are maintained in the System Property table [sys_properties], which you can access using the module navigator, or directly typing `sys_properties.list` in the Navigator Filter.

Note:
You must enable the *sn_erp_integration.enableModelModification* property on the correct scope. Enabling the *sn_erp_integration.enableModelModification* on a production instance can create new metadata records when new models and fields are added in ERP Data Hub.

Role required: sn_erp_integration.erp_admin, sn_erp_integration.erp_user

About this task

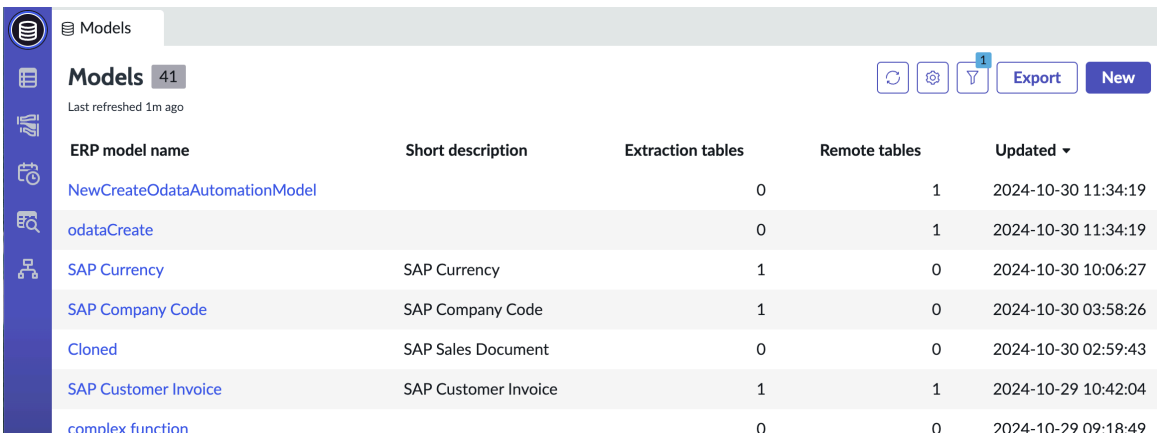
ERP Data Hub provides a standard set of ERP models, such as SAP Material Stock and SAP Purchase Document. You can also build new models. For a list of standard ERP models, which you must clone to modify, see Standard ERP models and extraction tables for ERP Data Hub.

An ERP model functions as a staging area that contains all potential fields you can add to remote and extraction tables, and read and update operations. You can then use the tables and queried data as a data source on the ServiceNow AI Platform.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP models page by selecting the ERP models icon ( ) in the side panel.

ERP Data Hub models



3. Review the list of ERP models.

ERP model columns

Column	Description
ERP model name	Name of the ERP model.
Short description	Brief description of what the ERP model represents.
ERP tables	Number of ERP tables on the system of record.
Extraction tables	Number of extraction tables on the ServiceNow AI Platform that are linked to the ERP model.
Remote tables	Number of remote tables on the ServiceNow AI Platform that are linked to the ERP model.
Updated	Date and time the model was last updated.

4. Select an ERP model to view and edit the details of the model.

ERP model Details tab

Field	Description
ERP model name	Name of the ERP model.
ERP module	Brief name of the ERP business area on the system of record.
ERP system	ERP system that represents a connection to a business section of your ERP system of record.
Application	Application scope the ERP model is linked to. For example, if you create a custom model, the model will be in that scope.
Short description	Brief description of what the ERP model represents.
Long text	Any longer description or information about the ERP data model.

5. **Optional:** View and work with the following on the **Details** tab of an ERP model:
   - Public comments and private work notes.
   - The Activity stream for the model.
   - File attachments.
6. Select **Save**.
7. View and confirm that the table entities included in the model by selecting the **Model entities** tab of the ERP model record.
   View details for an individual table by selecting the table name in the **Model entities** tab.

   ERP Data Hub automatically scans the linked ERP system to retrieve the latest entity data. However, you can select the refresh icon to update the data on demand.

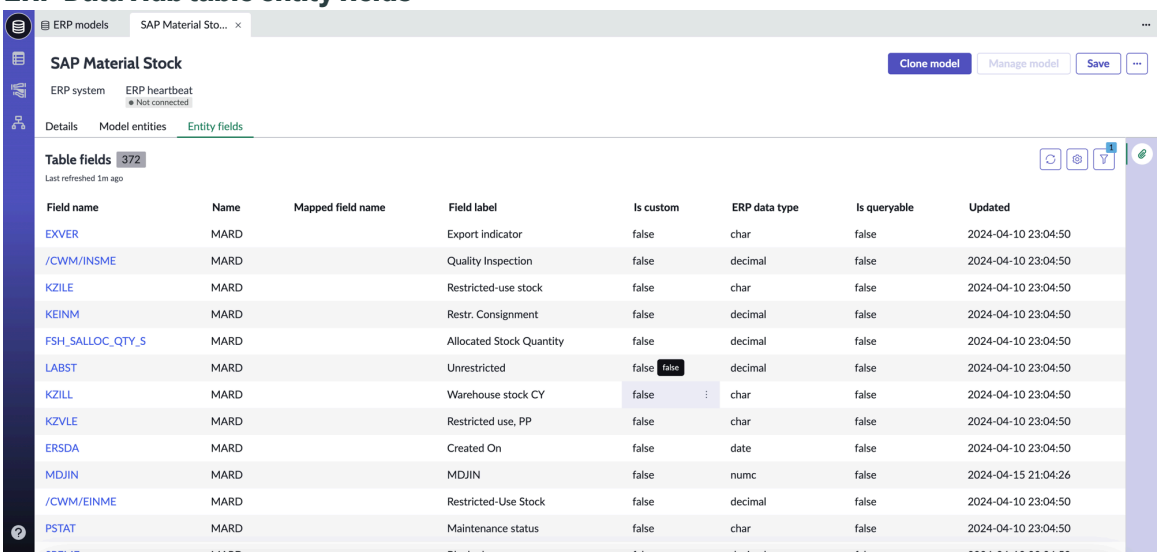
ERP Model entities tab details

Field	Description
Name	Name of the model table on the system of record.
Alias	Alias for the table. The alias refers to an alternative or substitute name for the table. An alias enables you to assign and customize a recognizable name for easier reference and identification. The alias can be a maximum of 40 characters, and contain a-z, A-Z, 0-9, and underscores.
Status	Status of the data synchronization, such as **Retrieved table data**.
Updated	Date and time the model entity was last updated.

8. **Optional:** Export the list of tables in the model by selecting the **Export** button. You can select the **File type**, such as **JSON** or **Excel**, and the **Delivery type**, such as **Download**.
9. View and confirm the table entity fields included in the ERP model by selecting the **Entity fields** tab of the ERP model record.

   For a description of the field values, see ERP Data Hub ERP model table field descriptions.

ERP Data Hub table entity fields



What to do next

After you have noted the available fields and tables, you can add new table entities to a model by managing the model. When you manage the model, you can also create read and update operations using table reads and BAPIs (Business Application Programming Interface). For more information, see Managing how models read and update the ERP system.

Clone an ERP model in ERP Data Hub

Clone a standard ERP (Enterprise Resource Planning) model that ships with ERP Data Hub. After you clone the model you can make modifications, for example, by adding new fields or tables.

Before you begin

Your administrator must enable the *sn_erp_integration.enableModelModification* property for you to edit, customize, and clone ERP models and tables. After you enable the *sn_erp_integration.enableModelModification* property, ERP Data Hub retrieves all tables and BAPIs (Business Application Programming Interface) to use when managing models. The property must be configured for either a non-production or production state. System properties are maintained in the System Property table [sys_properties], which you can access using the module navigator, or directly typing `sys_properties.list` in the Navigator Filter.

Note:
You must enable the *sn_erp_integration.enableModelModification* property on the correct scope. Enabling the *sn_erp_integration.enableModelModification* on a production instance can create new metadata records when new models and fields are added in ERP Data Hub.

Role required: sn_erp_integration.erp_admin, sn_erp_integration.erp_user

About this task

ERP Data Hub provides a standard set of ERP models, such as SAP Material Stock and SAP Purchase Document. You can also build new models. For a list of standard ERP models, which you must clone to modify, see Standard ERP models and extraction tables for ERP Data Hub.

Cloning ERP models to make customizations ensures that your changes don't break connections to other ServiceNow AI Platform applications.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP models page by selecting the ERP models icon ( ) in the side panel.
3. Select the name of ERP model that you want to clone.
4. Select the **Clone model** button.
5. In the **Clone this model** modal, enter the new **ERP model name**.
6. Select **Clone this model**.
   ERP Data Hub clones the model and displays a success message.
7. Open the ERP models page by selecting the ERP models icon ( ).
8. In the **ERP model name** column, select the cloned model you created.
9. Change the details for the ERP model on the **Details** tab, such as by updating the name.

Warning:
Changing the ERP system connected to the ERP model affects the available remote tables and extraction tables. If you change the ERP system, you must confirm the change on a warning modal.

For a description of the field values, see ERP Data Hub clone model field descriptions.

What to do next

Next, manage the model to specify additional criteria, such as which tables it reads and joins, as well as defining read and update operations and input/output parameters. For more information, see Managing how models read and update the ERP system.

Add a new ERP model

Add an ERP (Enterprise Resource Planning) model in ERP Data Hub to create a data set that contains ERP tables from the system of record, and enables you to read and send updates to the ERP system.

Before you begin

Role required: sn_erp_integration.erp_admin, sn_erp_integration.erp_user

About this task

An ERP model functions as a staging area that contains all potential fields you can add to remote and extraction tables, and read and update operations. You can then use the tables and queried data as a data source on the ServiceNow AI Platform.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP models page by selecting the ERP models icon ( ) in the side panel.
3. Select **New**.
4. On the new model tab, fill in the fields.
   For a description of the field values, see ERP Data Hub new model field descriptions.
5. Select **Save**.
6. Open the ERP models page again by selecting the ERP models icon ( ).
7. Update the page by selecting the refresh list icon.
   The new model is displayed in the list.

What to do next

After you add a new ERP model, you can manage it to specify additional criteria, such as which tables it reads and joins, any parameters for inputs and outputs, and whether it uses a BAPIs (Business Application Programming Interface) to update the system of record. For more information, see the following topics:

- Managing how models read and update the ERP system
- Add joins between ERP tables

Managing how models read and update the ERP system

After you create an ERP (Enterprise Resource Planning) model in ERP Data Hub, you can specify how it reads and updates the ERP system using parameters.

When creating a read or update operation, first add the operation. Then define operation entities and parameters in the following tabs of the ERP model manager page.

- **Manage entities**: Specify the table to read or BAPI (Business Application Programming Interface, a remote procedure call that's similar to an API) to use for reading or updating the ERP system.
- **Specify inputs**: Define how fields on the ERP system map to parameters to specify how data is queried. If parameters can't be retrieved, you can also define any default values to pass to the ERP system.
- **Choose output**: Define parameters for how returned data is stored on the ServiceNow AI Platform by specifying outputs.

Read and update operations for ERP system

ERP Data Hub contacts the ERP system using read and update operations.

- You can use either a table read or BAPI to read the ERP system.
- You must use a BAPI for update operations.
- You can add either table reads or a BAPI function call to a model for read operations, but not both.
- You can add multiple tables to a table read operation, but you can specify only one BAPI for a function call read operation.

Managing tables and entities for ERP operations

An entity is the foundation of how the operation accesses the ERP system to read or update it. Use the **Manage entities** tab to define requests and the content of their responses by specifying the BAPI or name of the table.

When you add an entity to an operation, you must specify the following information:

- How ERP Data Hub retrieves data from or sends updates to the ERP system.
  - For read operations, you must specify whether you're reading tables on the ERP system or using a pre-defined BAPI to read the system.
  - For update operations, you must use a BAPI.
- The name of the table to read or BAPI to use.

For instructions on adding entities, see Add an operation to a model in ERP Data Hub.

Managing operation input parameters

After you specify the operation's tables or BAPI, ERP Data Hub automatically populates the **Specify inputs** tab with the required input parameters.

ERP Data Hub uses parameters as part of the method/function call to define and map data that's passed to the ERP system.

- The Output parameters section is where you enter optional default values for parameters that are used to query the ERP system. If no input value is provided when querying the model, the **Default value** for each parameter is used as a fallback. Default values can also be utilized for mapping constants.
- The Tables (for read operations)/Function call (for BAPI operations) section is where you define and map fields from the ERP system that ERP Data Hub sends as parameters in the operation. When you select a field, use the automap functionality to automatically update its **Mapped value** and add a row for the parameter to the Output parameters section. If you define

a `Constant` as the **Type** and enter the constant value in the **Mapped value** field, mapped inputs can act as filter criteria. You can add and nest as many related parameters as needed.

If you're adding a complex, nested parameter, such as an address that includes several other parameters (one for street, one for city, one for country), ERP Data Hub automatically identifies that it needs additional related parameters and creates new, nested parameter rows that you must then populate with the related values. You can nest only parameters with **Object** or **Array** as the **Data type**.

The available data types for parameters are:

- Object
- Array
- String
- Date Time
- Date
- Time
- Char
- Decimal

Note:

An example use case is running a sales order BAPI to find out what items are in an order. You must specify the order ID as a mapped field in the Tables (for read operations)/ Function call (for BAPI operations) section, using the automap option to define which field is referenced in the **Mapped value** field. After defining all operation inputs and outputs, you can build a flow in Workflow Studio. In the flow, enter the order ID as the parameter in use when the flow runs to call the ERP system.

Another example would be adding a parameter for **Order billing dates** in the Table fields (for table read operations)/Function call (for BAPI operations) section. Then build a flow in Workflow Studio that enables you to specify a date or date range to retrieve all orders from that time period.

Note:
It doesn't matter what order you define parameters in. ERP Data Hub displays optional parameters in alphabetical order when you save.

For instructions on managing inputs, see Manage input parameters for an ERP Data Hub model operation.

Selecting outputs for a read operation

You must create output parameters to define how the data is mapped to the ERP system and stored on the ServiceNow AI Platform.

For instructions on managing outputs, see Choose output parameters for an ERP model.

Adding retrieved ERP fields to tables

When you add mapped fields or parameters as outputs and successfully read or update the ERP system, each parameter appears as a field that you can then add to a remote table or an extraction table. Manage the fields for the remote table or extraction table to add the retrieved parameters. For more information, see the following topics:

- Customize fields for an ERP remote table in ERP Data Hub
- Select fields for an extraction table in ERP Data Hub

Building flows to call the ERP system

After all parameters are defined and you have built and run the read or update operation, build a flow in Workflow Studio that uses the defined parameters.

Enter any filter criteria by specifying a value when you select the parameter in the **Mandatory Field** of the flow's action. For more information, see Building flows to read or update the ERP system.

Add an operation to a model in ERP Data Hub

Add an operation to an ERP (Enterprise Resource Planning) model in ERP Data Hub to define how it retrieves data from or writes data to the ERP system, or creates a new instance of the business object.

Before you begin

Role required: sn_erp_integration.erp_admin, sn_erp_integration.erp_user

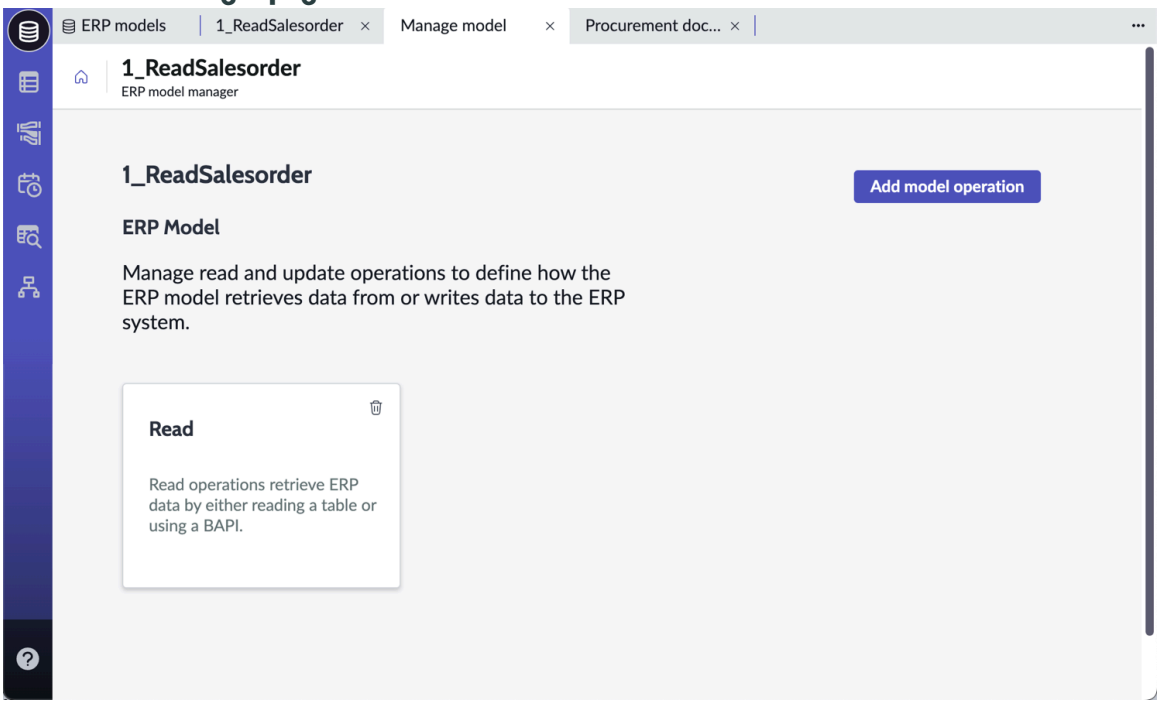
About this task

- Read operations retrieve ERP data by either reading a table or using a BAPI.
- Update operations use a BAPI to write updates to the ERP system.
- Create operation creates a new instance of the business object in the SAP system. (Available starting in Xanadu Store Release 2)

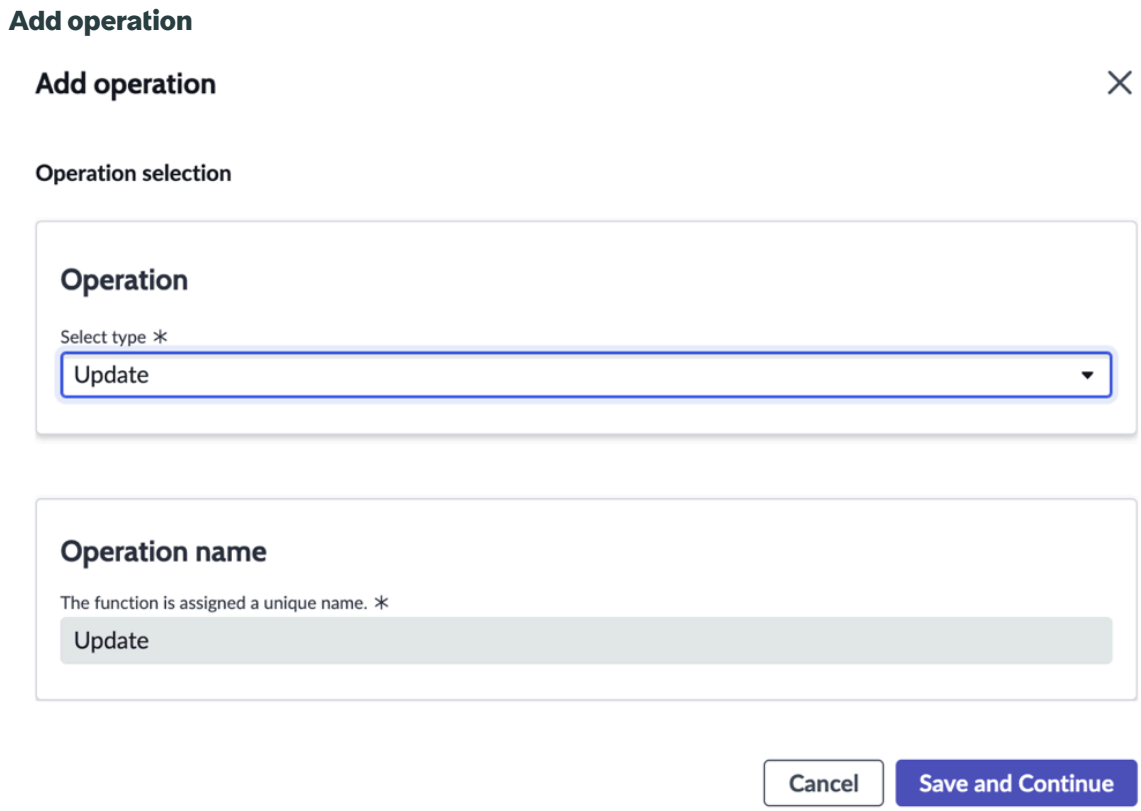
Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP model page by selecting the ERP model icon ( ) in the side panel.
3. Select the model to which you want to add an operation.
4. Select the **Manage model** button.

ERP model manager page



5. Select **Add model operation**.
6. Specify the type of operation that you're adding in the **Select type** field of the **Add operation** modal.
   - **Update** sends data back to write to the ERP system.
   - **Read** reads and retrieves data from the ERP system and brings it onto the ServiceNow AI Platform.
   - **Create** is used to create a new instance of the business object in the SAP system. (Available starting in Xanadu Store Release 2)



7. Select **Save and continue**.

Result
The foundation of the operation is created.

What to do next
Next, you must add the read or update entity to the operation. For more information, see Add a read, update, or create entity to a model in ERP Data Hub

You can select the delete icon (🗑) on the operation's card to remove any operations you don't need, or to start over.

Add a read, update, or create entity to a model in ERP Data Hub

Specify the operation entity, which is the table or BAPI (Business Application Programming Interface) that ERP Data Hub uses for read, update, or create operations.

Before you begin
You must have already added the read, write, or create operation before you can add an entity to it. For more information, see Add an operation to a model in ERP Data Hub.

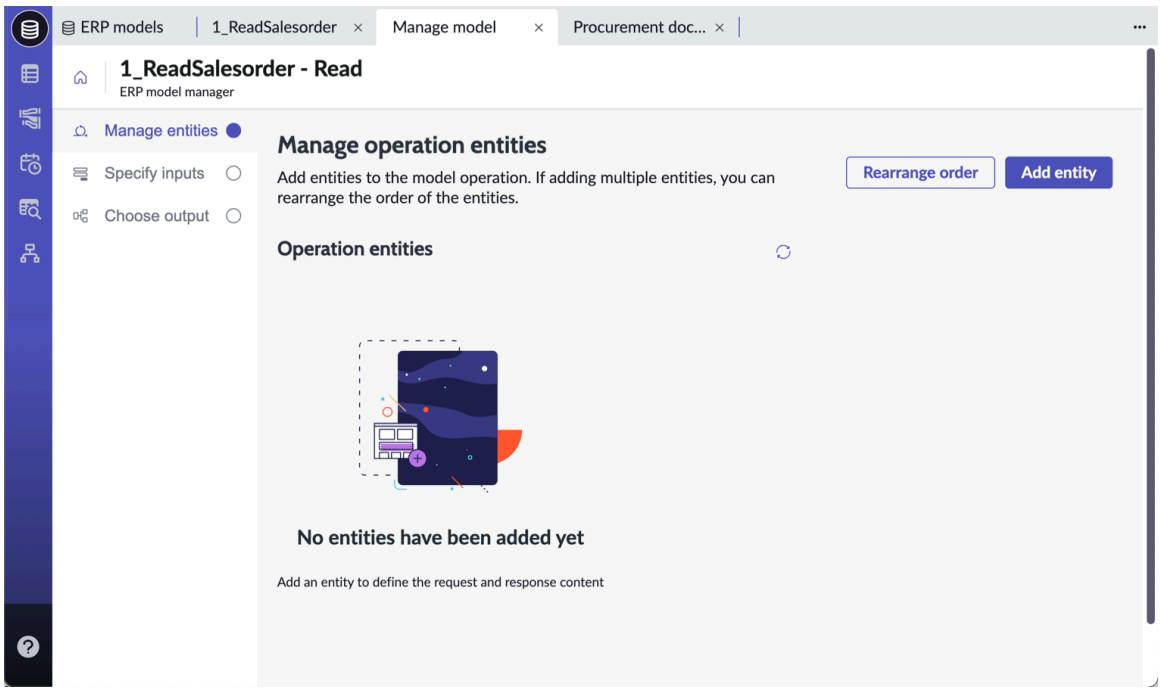
Role required: sn_erp_integration.erp_admin, sn_erp_integration.erp_user

About this task
- Read operations retrieve ERP data by either reading a table or using a BAPI.
- Update operations use a BAPI to write updates to the ERP system.
- Create operation creates a new instance of the business object in the SAP system. (Available starting in Xanadu Store Release 2)

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP model page by selecting the ERP model icon ( ) in the side panel.
3. Select the model that you want to add an operation entity to.
4. Select the **Manage model** button.
5. Select **Add entity** on the **Manage entities** tab.

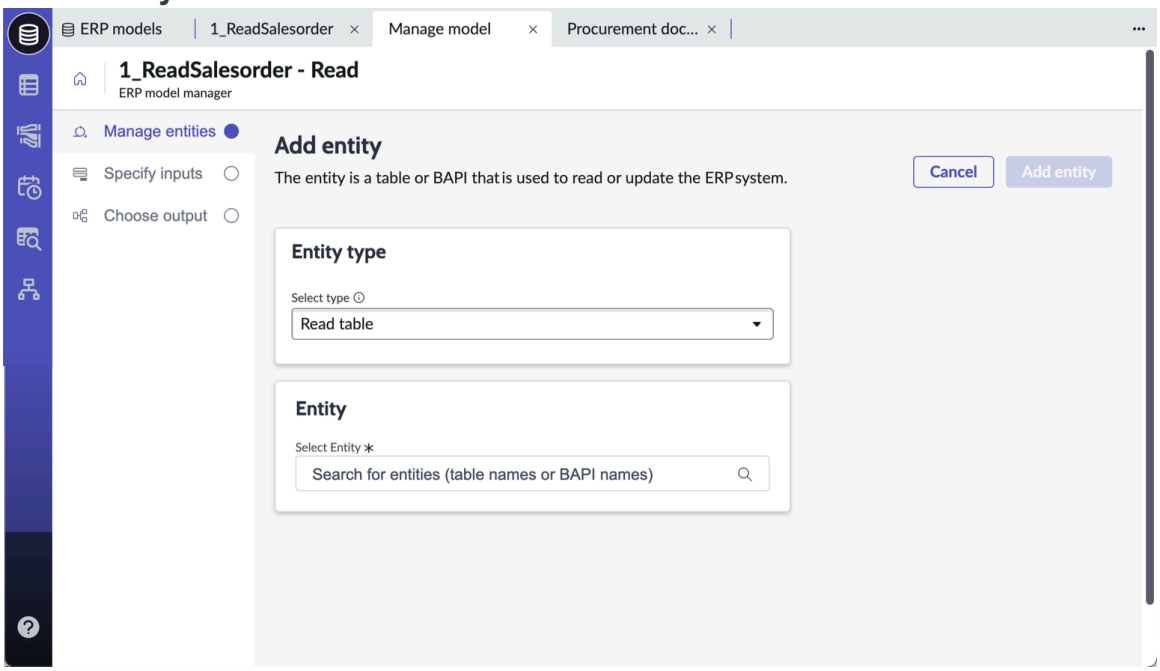
Manage entities tab



6. For read operations, specify the type of read operation in **Select type**.
   - **Read table** defines a read operation entity that retrieves data by reading it directly from a table on the ERP system.
   - **Function call (BAPIs)** defines a read operation entity that uses a BAPI remote procedure call.
   - **Function call (RFCs)** defines a read operation entity that uses an RFC call.
   - **OData** enables you to work with various business objects (for example, read, create, update, and delete) defined in the ERP system.

> **Note:**
> Update operation entities must use a BAPI Function call, so this field is read only for update operations.

Add an entity



7. Define the entity by specifying the name of the table to read or BAPI to use in **Select entity**.

   The AI Search for this field can help you find what you're looking for faster, for example by entering `Countries` instead of `T005`. AI Search also suggests tables and BAPIs that are in the connected ERP system.

   For table read operations, you may need to check the **Model entities** tab of the current or another, related model to get the table name, or ask your SAP admin. For more information, see View and edit the foundation of ERP models.

8. When you're finished, select **Add entity**.
9. **Optional:** Rearrange and delete table entities as needed.

   Important:
   Reordering deletes any existing table joins for the reordered entities.

   a. Select **Rearrange order**.

   b. Drag the tables to the order that you want.

   c. Select the delete icon (🗑) on the card for a table to remove any tables you don't need. Deleting an entity removes all of its related field mappings and table joins.

   d. Select **Confirm reorder**.

Result

The entity is added to the operation. ERP Data Hub connects to the ERP system using the specified entity and retrieves data that you can use to define input and output parameters.

When you view the entity on the **Manage operation entities** page, a status shows whether the ServiceNow AI Platform is retrieving data or has finished retrieving data.

If you added a **Read table** operation, you can repeat this process to add as many tables as necessary to the operation. Then you can add joins to create relationships between tables.

Note:
If you have more than one table for an operation, you must join the tables.

For more information, see Add joins between ERP tables.

The **Entity fields** tab for the ERP model automatically populates for the new table when the table is connected to an ERP system. View their details as needed. For more information, see ERP Data Hub ERP model table field descriptions.

What to do next
Next, you must specify the input parameters for the operation. For details, see Manage input parameters for an ERP Data Hub model operation.

Add joins between ERP tables

Link multiple ERP (Enterprise Resource Planning) tables from the system of record to build an ERP model in ERP Data Hub using table joins.

Before you begin
Table joins require a read operation that uses table read entities. For more information, see Add an operation to a model in ERP Data Hub.

When you add table joins, the parent table is the first table listed on the **Manage entities** tab of the ERP model manager page. Child tables pull information from the parent table.

Note:

- If you have more than one table for an operation, you must join the tables.
- You can create table joins only for table read operations, not for operations that use a BAPI (Business Application Programming Interface).

Role required: sn_erp_integration.erp_admin, sn_erp_integration.erp_user

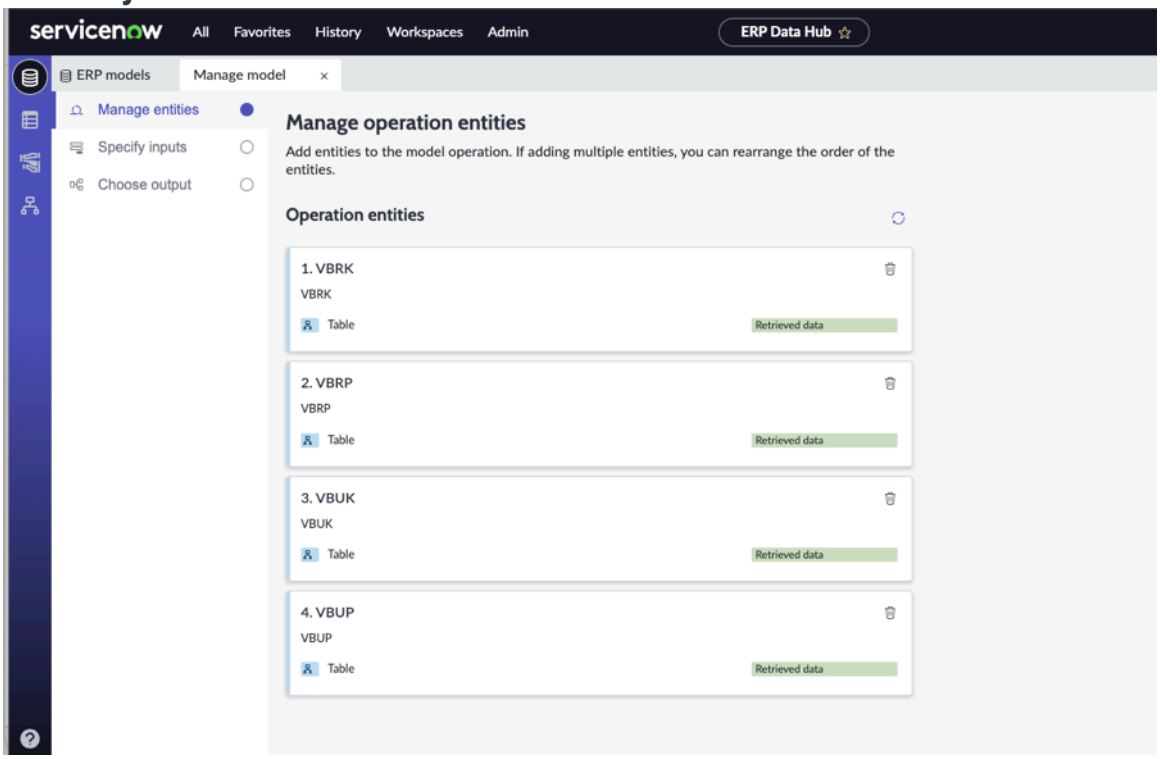
About this task
Table joins link different tables through shared fields. Joins enable you to access data from multiple tables based on logical relationships between them. The relationship can be conditional, which you specify using join conditions.

Join fields defines the common attribute or key used to connect records in a child table with their corresponding parent records.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP model page by selecting the ERP model icon () in the side panel.
3. Select the model that you want to add a join to.
4. Select the **Manage model** button.
5. Select the **Read** operation that you're adding the table join to.
6. **Optional:** Add the tables to join as **Table read** operation entities if they haven't yet been added to the model.
   For more information, see Add a read, update, or create entity to a model in ERP Data Hub.

Tables to join



7. Rearrange the tables on the **Manage entities** tab to place the parent table for the join as the first table listed on the ERP model manager page.

> **Important:**
> Reordering deletes any existing table joins for the reordered entities.

   a. Select the **Rearrange order** button.

   b. Drag the tables to the order that you want, with the parent table as the first table listed on the page.

   c. Select the **Confirm reorder** button.

8. Select the **Specify inputs** tab, which is where you update input parameters to specify table join as a parameter.
   The input parameters for each table appear.

9. Create a table join parameter.

   a. Find the parameter that will be the parent join field in the Output parameters section and note its name.

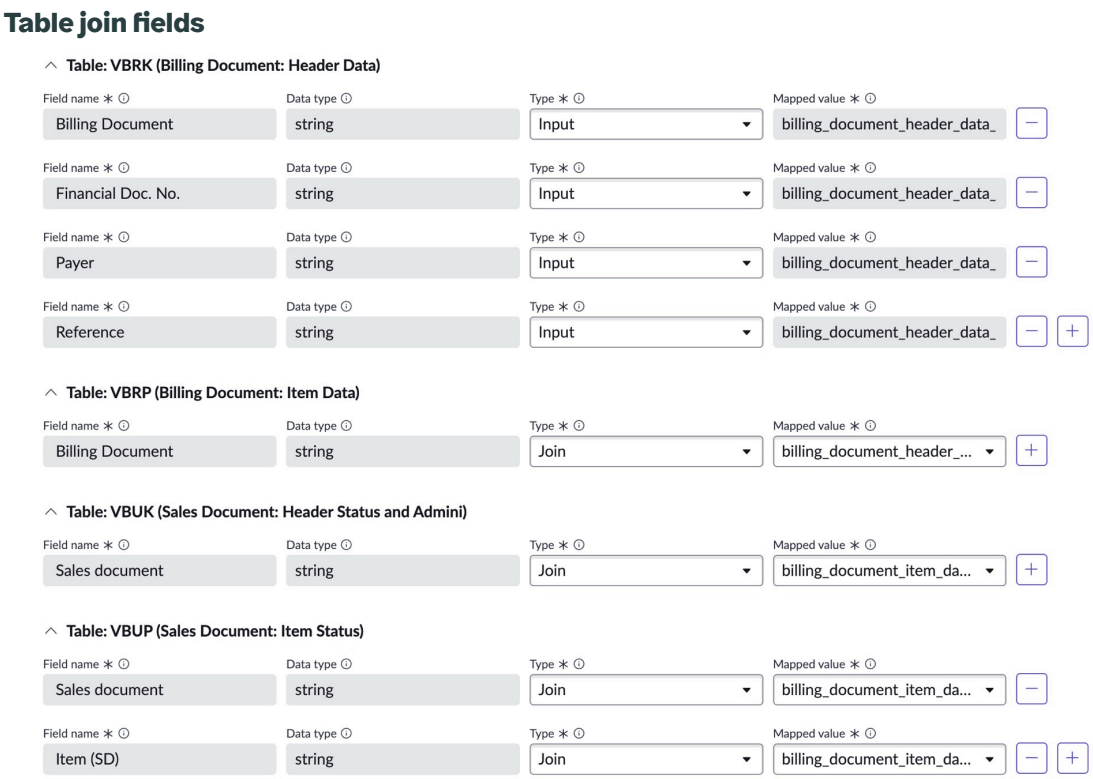
   b. Add a new mapped field row in the table by selecting the add (+) icon.

   c. Enter or select the parent join **Field name** in the new field row.

   d. Select **Join** from the **Type** field.
   The value in the child table **Mapped value** field updates automatically with the joined field name.

For example, in the SAP Material Stock model, you can specify **Material** as the parent field in the **Field name** field, and join it to the material ID by entering `mard_material_matnr` in the parent **Mapped value** field.

e. Select **Save**.



10. **Optional:** Add or update any output parameters as needed in the Output parameters section. The joined parameter that you added to as an input is automatically added to the **Choose output** tab.

    For more information, see Choose output parameters for an ERP model.

Result

After you're done creating table joins, you can specify where the returned ERP data goes, and build flows that retrieve and output the data. For more information, see the following topics:

- Specifying where the ERP system data is saved
- Building flows to read or update the ERP system

Manage input parameters for an ERP Data Hub model operation

Specify how fields on the ERP (Enterprise Resource Planning) system map to input parameters and their values to define the inputs for an operation that reads or updates the system of record from ERP Data Hub.

Before you begin

Role required: sn_erp_integration.erp_admin, sn_erp_integration.erp_user

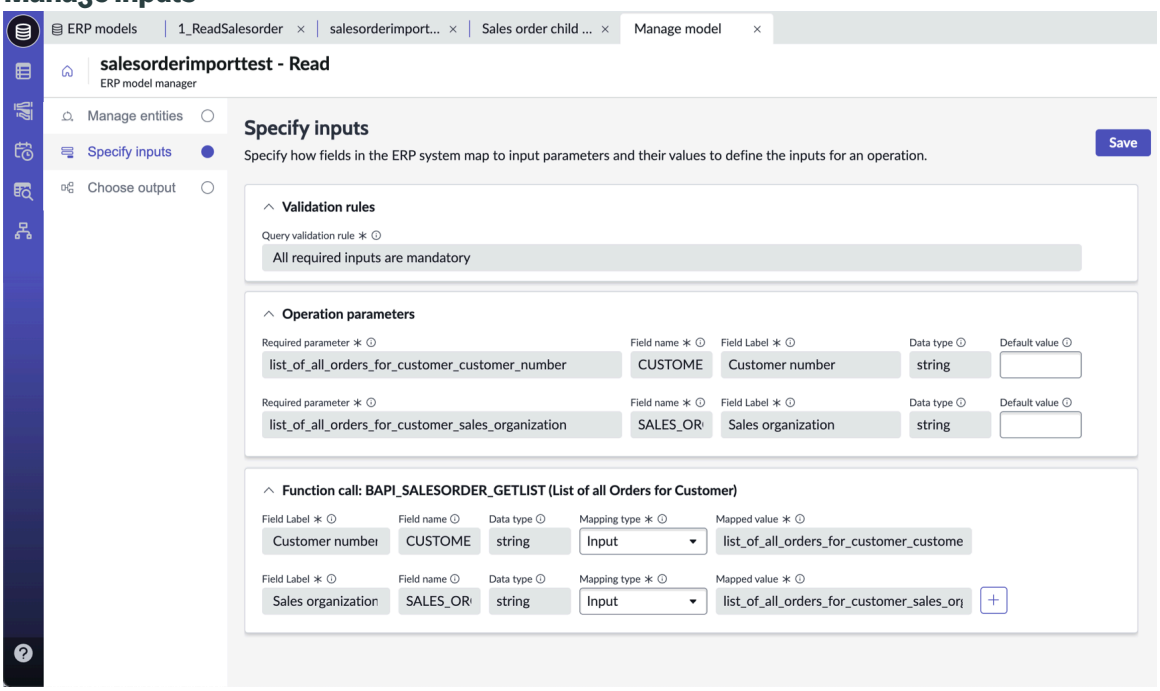
About this task

If you're already in the process of managing a model and ready to specify inputs, you can skip to step 5.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP model page by selecting the ERP model icon ( ) in the side panel.
3. Select the model with the operation that you want to add inputs to.
4. Select the **Manage model** button.
5. Open a model operation.
   If you do not have a model operation, add one to the model. For more information, see Add an operation to a model in ERP Data Hub.
6. Open an entity.
   If you do not have an entity, add one to the operation. For more information, see Add a read, update, or create entity to a model in ERP Data Hub
7. Select **Specify inputs**.
   The required parameters for the operation appear in both of the following places:
   - The Operation parameters section, where you define any default values to use if the operation fails.
   - The Tables/Function call section, where you define the parameters the operation uses. The Table section appears for read operations, and the Function call section appears for BAPI (Business Application Programming Interface) operations.

Manage inputs



8. Define whether the operation inputs are required in the **Query validation rule** field.
   - **All inputs are mandatory**
   - **At least one input is mandatory**
   - **No inputs are mandatory**
9. Review the required and optional parameters that are already defined for the operation and note what you must add.
10. Define a new input parameter to be sent when querying the ERP system.

For table read operations, a Tables section appears in the mapped table fields section for each defined table entity. Make sure that you're adding the parameter for the correct table.

Note:
It doesn't matter what order you define parameters in. ERP Data Hub displays optional parameters in alphabetical order when you save.

a. Select the add icon (+) next to the last-defined parameter.

b. Fill in the fields to define the parameter.

Fields to define an input parameter

Field	Description
Field name	Name of the field from the ERP system that you're defining as a parameter. ERP Data Hub automatically retrieves fields from the table defined in the operation entity.
Data type	(Read-only) Automatically populated value that specifies the type of data the parameter contains.
Type	Definition of how the parameter is sent. ▪ **Input** parameters have their **Mapped value** automatically populated. ▪ **Constant** defines a parameter whose value never changes, for example, to use as filter criteria. You must specify the value in the parameter's **Mapped value** field. ▪ **Join** (Table read operations only) indicates that you're creating a table join. For more information, see Add joins between ERP tables.
Mapped value	Specific value of the parameter. ▪ For **Input** parameters, accept the system-generated name. ▪ For **Constant** parameters, enter a set value that's always sent for the parameter. ▪ For **Join** parameters, select the field to join the parameter with. For details on creating joins, see Add joins between ERP tables.

If you're adding a complex, nested parameter, such as an address that includes several other parameters (one for street, one for city, one for country), ERP Data Hub automatically identifies that it needs additional related parameters and creates new, nested parameter rows that you must then populate with the related values. You can nest only parameters with **Object** or **Array** as the **Data type**.

11. Include the newly defined parameter in the Output parameters section by selecting the automap icon ( ).

ERP Data Hub automatically updates the **Mapped value** for the Tables/Function call parameter, and adds a row for the parameter to the Output parameters section.

12. **Optional:** Specify a **Default value** to be used for the parameter in the newly added **Optional parameter** row in the Output parameters section.
    The **Default value** is used when the input parameter isn't specified in the query.
13. **Optional:** Remove any optional parameters that you don't need when querying the ERP system by selecting the remove (-) icon.
    You can't remove any **Required parameters**.
14. Make any additional edits to existing parameters.
15. Select **Save**.

What to do next

Next, check the output parameters for the operation and update as needed. For more information, see Choose output parameters for an ERP model.

Choose output parameters for an ERP model

Specify output parameters for an ERP (Enterprise Resource Planning) system read or update operation in ERP Data Hub to define how fields and parameters are mapped from the ERP system to the ServiceNow AI Platform. Output parameters also define how returned data is stored on the ServiceNow AI Platform.

Before you begin

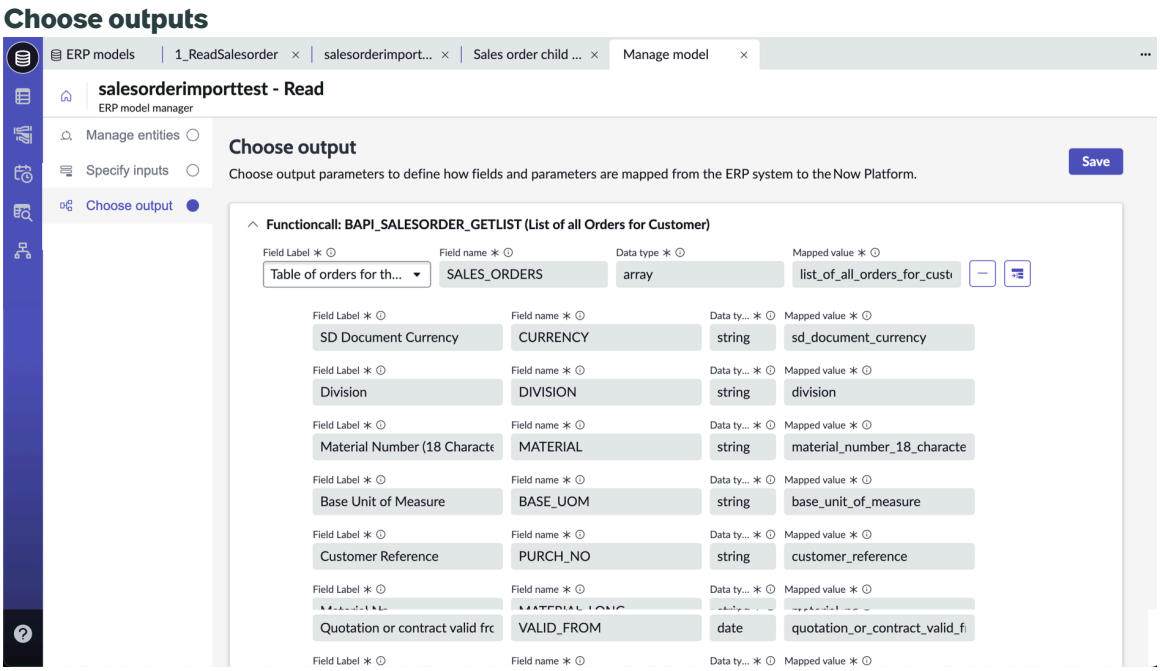
Role required: sn_erp_integration.erp_admin, sn_erp_integration.erp_user

About this task

If you're already in the process of managing a model and ready to specify outputs, you can skip to step 5.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP model page by selecting the ERP model icon ( ) in the side panel.
3. Select the model that you want to add a read operation to.
4. Select the **Manage model** button.
5. Select the **Choose outputs** tab.
6. Review the existing outputs to see if you must add any additional parameters.



7. Add a new output field or parameter.

   a. Select the added icon (+) next to the last-defined parameter.
   If you're adding a mapped field for a table read operation, make sure to add the field to the applicable table section.

   b. Search for and select the **Field name** for table reads or **Parameter name** for BAPI (Business Application Programming Interface) operations.
   The **Data type** field is automatically updated with the applicable values for your selection.

   c. Enter the corresponding value in the **Mapped value** field.

   If you're adding a complex, nested parameter, such as an address that includes several other parameters (one for street, one for city, one for country), ERP Data Hub automatically identifies that it needs additional related parameters and creates new, nested parameter rows that you must then populate with the related values. You can nest only parameters with **Object** or **Array** as the **Data type**.

8. **Optional:** Add any nested output fields or parameters to choose what data to include from a complex parameter.
   For example, you may have a full billing address in the object parameters, but only need to include the city and postal code as output.

   a. Specify the parent output field or parameter.

   b. Select the settings ( ) or add icon (+) for the parent row.

   c. In the modal that appears, select any additional, related fields or parameters to include.
   The parent automatically updates with the related, nested fields or parameters you selected.

9. Select **Save**.

Result

When you add mapped fields or parameters as outputs and successfully read or update the ERP system, each parameter appears as a field that you can then add to a remote table or an

extraction table. Manage the fields for the remote table or extraction table to add the retrieved parameters. For more information, see the following topics:

- Customize fields for an ERP remote table in ERP Data Hub
- Select fields for an extraction table in ERP Data Hub

Specifying where the ERP system data is saved

Data that ERP Data Hub retrieves from ERP (Enterprise Resource Planning) systems can be used in remote tables, extraction tables, and added to flows as data pills in Workflow Studio.

Adding retrieved data to remote tables and extraction tables

Use the JSON contents of the **Response** in a flow's output to save the ERP data in the following ways:

- Store the data in a remote table. Remote tables get their records from running an associated script against an external data source.
- Store the data in an extraction table. Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.

When you add mapped fields or parameters as outputs and successfully read or update the ERP system, each parameter appears as a field that you can then add to a remote table or an extraction table. Manage the fields for the remote table or extraction table to add the retrieved parameters. For more information, see the following topics:

- Customize fields for an ERP remote table in ERP Data Hub
- Select fields for an extraction table in ERP Data Hub

Using retrieved ERP data in flows

The **Use ERP Data** action returns ERP data in an output data pill called **Response**. The **Response** pill is available when you build a flow in Workflow Studio with the **Use ERP Data** action. The **Response** data pill is available in the following places of the action:

- The **Outputs** tab
- The **Output** section in the **Data** pane

You can then add the **Response** data pill or any of the child **record** data pills to a flow to parse the returned JSON.

For example, you can generate a record for each response from the ERP system, making that data available for use on the ServiceNow AI Platform. For more information, see Building flows to read or update the ERP system.

Next steps after extracting data from your ERP system using ERP Data Hub

After you identify and extract ERP (Enterprise Resource Planning) data with ERP Data Hub, you can use that data on the ServiceNow AI Platform as the data source for products and apps.

Use retrieved ERP data in flows

Build flows in Workflow Studio to specify details for when you query or update the ERP (Enterprise Resource Planning) system.

For example, you can generate a record for each response from the ERP system, making that data available for use on the ServiceNow AI Platform. For more information, see Building flows to read or update the ERP system.

Build a ServiceNow app that consumes ERP data

ERP data from the system of record is available in the remote tables and ERP extraction tables that you configure in ERP Data Hub. You can also use table transform maps to put extracted ERP data into a Glide table.

After ERP data is available on tables in the ServiceNow AI Platform, you can use those tables as the foundation for app builders. For example, you can use ERP tables as a data source when you add data in App Engine Studio. For more information, see Create a data model for your application.

ServiceNow low- and pro-code builders

Use any of the following ServiceNow builders to create apps using custom data:

- App Engine Studio
- Flows in Workflow Studio
- Playbooks in Workflow Studio
- Table Builder
- UI Builder
- Workspace Builder

Using Glide to query ERP data

You can also access data from the system of record through the Glide API.

For more information, see Sample Glide query for ERP data in ERP Data Hub.

Managing ERP development pipelines in ERP Data Hub

Move your ERP (Enterprise Resource Planning) systems, ERP models, tables, operations, and flows from a development instance to a production environment when they're ready.

Changes that you could promote from a development instance to a production instance include adding:

- Fields to tables
- Tables or BAPIs (Business Application Programming Interface) to ERP models
- Table joins and fields to link tables
- Read and update operations
- Flows built with the **Use ERP Data** action to query and update the system of record

> **Note:**
> You should do your development on a non-production instance. If you make changes on a production instance, and then promote changes from a non-production instance to the production instance, any changes you previously made on the production instance are overwritten.

There are several ways to move changes to your production instance on the ServiceNow AI Platform:

1. Use System Update Sets to transfer changes from a development instance to a non-production and then production instance. For more information, see System update sets.
2. Add the changes to the ServiceNow Store and use the **Share with others** option to install the updates on the production instance. For more information, see Publish an application to an Update Set.

For more information on ways to publish your ERP updates, see Application sharing.

Building flows to read or update the ERP system

After you configure an operation in ERP Data Hub, you can build a flow in Workflow Studio to specify details for querying the ERP (Enterprise Resource Planning) system. For example, build a flow that filters by Order ID.

Actions in Workflow Studio for calling the ERP system

A Use ERP Data action is automatically available in Workflow Studio after installing ERP Data Hub.

The Use ERP Data action enables you to use ERP data outside of ERP Data Hub for other processes. For example, you could create a task or an incident based on ERP data using the Use ERP Data action.

After you're done creating and managing your model in ERP Data Hub, go to Workflow Studio and use the Use ERP Data action to test your model's inputs and outputs.

When you test the action, you must specify the model you're using, which fields to read or update, and the ERP system. For more information on the action, see ERP Data Hub Use ERP Data action details for flows.

Using parameters to filter data requests

After you successfully test the Use ERP Data action, build a flow that uses that action to retrieve or update the specified data in the ERP system.

When you add the Use ERP Data action to a flow, you can specify any **Mandatory Field(s)** and **Optional Field(s)** defined as inputs when you managed the model. Required and optional fields must already be configured as required and optional input parameters.

If you select a field that was defined as a parameter for the system query, you can then enter a value to filter on. For example, if you build a flow that uses the Use ERP Data action and specify the SAP Stock Movement model, you can select **iseg_fiscal_year** as a required field, and then enter a year to filter the data returned, for example 2023.

Process for building flows to retrieve ERP data

If you want to use ERP data for processes or tasks outside of ERP Data Hub, use the following workflow:

1. In Workflow Studio, test the **Use ERP Data** action by selecting the model in the **System** field of the Test Action modal and then specifying the inputs.
   - For more information on testing actions, see Test an action.
   - For more details on inputs for the action, see ERP Data Hub Use ERP Data action details for flows.
2. After the test runs, view the action execution details. For more information, see Flow execution details.
3. The returned data from testing the action appears in the **Response** field of the output data.

- For more information on outputs for the action, see ERP Data Hub Use ERP Data action details for flows.
- For details on incorporating returned ERP data, see Specifying where the ERP system data is saved.

4. Build a flow that runs the Use ERP Data action.

   The **Use ERP Data** action returns ERP data in an output data pill called **Response**. The **Response** pill is available when you build a flow in Workflow Studio with the **Use ERP Data** action. The **Response** data pill is available in the following places of the action:

   - The **Outputs** tab
   - The **Output** section in the **Data** pane

   You can then add the **Response** data pill or any of the child **record** data pills to a flow to parse the returned JSON.

   For more information, see Building flows.

Note:
ERP Data Hub appears as **ERP Integration** when you work with it in Workflow Studio.

Connect ERP Data Hub to SAP using OData and HTTP

Extract data securely from ERP OData APIs using ETL for use in remote tables and extraction tables. OData connects to SAP via HTTP.

Important:

OData v2 does not use snapshot isolation so you might experience some data consistency issues when retrieving data from an external ERP source.

Providing OData access to users

You must have an SAP system that has been enabled to make an OData connection.

As of Xanadu Store Release 2, extract data using OData and an HTTP connection. To give users OData access, see the following instructions on the SAP help site Back-End Server: Assign OData Service Authorization to Users.

New properties

The following are properties related to OData available for ERP in Xanadu Store Release 2.

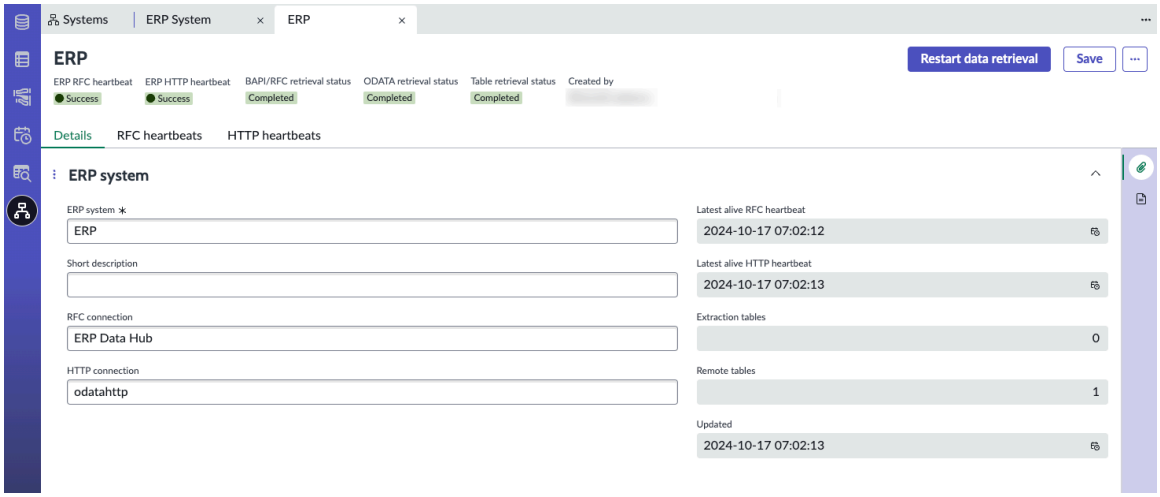
Property	Type	Description	Create manually?
sn_erp_integration.response_timeout	integer	Specifies the timeout value for OData response. If OData calls are timed out frequently, increase the timeout value. Specify the value in seconds. The default is 100 seconds. This value is used for responses both from	No

Property	Type	Description	Create manually?
		external web and from a MID Server.	
sn_erp_integration.use_csrf_token	true | false	Indicates if CSRF token should be sent for OData calls in ERP Data Hub operations.	No
sn_erp_integration.catalog_service_path	string	After the hostname and port, this is the path to connect with any SAP catalog service. The default is: /sap/opu/odata/iwfnd/CATALOGSERVICE;v=2/ServiceCollection. After creating the property and setting it to true, a list of all services are retrieved from SAP. The information is stored in an XML file and attached to the system record. The XML can be used later. For example, parse the XML while offline with no connection to SAP.	Yes
sn_erp_integration.odata_service_path	string	After the hostname and port, this is the path to connect with any SAP OData service. Add a URL in **Value** to specify the OData service. The default is: /sap/opu/odata/sap.	Yes

Note:
To add a new property manually, verify that your scope is set to ERP Data Hub, then navigate to sys_properties.list and select **New**.

Heartbeat information

For an ERP system, there are separate heartbeat indicators for RFC and HTTP. When a system is established, the heartbeats become active and the status is updated, including any errors. If the heartbeat calls are successful, BAPI and OData retrieval is triggered in parallel and the status can be seen on the system record. BAPI and table list retrieval is done via RFC. OData retrieval is done via HTTP.



More information

For more information about using OData in ERP Data Hub, see Create an OData connection in ERP Data Hub and OData capabilities supported by ERP Data Hub.

OData capabilities supported by ERP Data Hub

Details about the OData query capabilities supported in ERP Data Hub.

For information about OData connections in ERP Data Hub, see Connect ERP Data Hub to SAP using OData and HTTP and Create an OData connection in ERP Data Hub.

OData query capabilities

OData query	
$select, $filter, $count	Available in the **Use ERP** flow. When using the inputs and outputs parameters in the model, $select and $filter are applied. Use $count when invoking the **Use ERP** flow.
$orderby	Supported only via encoded query while executing the remote tables or extraction tables.
$top and $skip	When doing the internal operations, $top and $skip are supported only while pagination is done. Cannot send $top and $skip via any flow.

Create an OData connection in ERP Data Hub

Create an OData v2 connection to link to SAP via HTTP so data can be extracted for use in remote tables and extraction tables.

Before you begin

Role required: sn_erp_integration.erp_admin

For information about OData connections in ERP Data Hub, see Connect ERP Data Hub to SAP using OData and HTTP and OData capabilities supported by ERP Data Hub.

About this task

Your administrator must enable the *sn_erp_integration.enableModelModification* property for you to edit, customize, and clone ERP models and tables. After you enable the *sn_erp_integration.enableModelModification* property, ERP Data Hub retrieves all tables and BAPIs (Business Application Programming Interface) to use when managing models. The property must be configured for either a non-production or production state. System properties are maintained in the System Property table [sys_properties], which you can access using the module navigator, or directly typing `sys_properties.list` in the Navigator Filter.

Note:
You must enable the *sn_erp_integration.enableModelModification* property on the correct scope. Enabling the *sn_erp_integration.enableModelModification* on a production instance can create new metadata records when new models and fields are added in ERP Data Hub.

Procedure

1. Confirm that you have an SAP system that has been enabled to make an OData connection.
2. Create a connection and credential alias, specifying HTTP as the **Connection type**.
   For more information, see Create a Connection & Credential alias.
3. Create an HTTP connection and associate it with the new alias.

   Note:
   For more information, see Create an HTTP(s) connection. If you choose to use a MID Server, users with access to the services can use the same credential for RFC and HTTP.

4. Create a system with the HTTP connection.
   For more information, see Create an ERP system in ERP Data Hub.
5. On the system record, confirm that the heartbeats are successful and the retrieval status is complete.
   If any have failed, select **Restart data retrieval**. Any data retrieval that failed (BAPI, OData, or tables) is fetched again.
6. Create a model and, after saving, open the model record.
   For more information, see Add a new ERP model.
7. Select **Manage model**.
8. Select **Add model operation**.
   a. Choose a **Select type**.
   b. Select **Save and Continue**.
9. Select the new operation.
10. Select **Add entity**.
    a. In **Select type**, choose **OData**.
    b. In **Select entity**, search for and select a service, for example **API_BUSINESS_PARTNER (Remote API for business partner)** in the OData service catalog.
    When you specify the service, a call is made to the SAP service to read its metadata. The default service is /sap/opu/odata/iwfnd/CATALOGSERVICE;v=2/ServiceCollection. If you

must change the service, create a property named sn_erp_integration.odata_service_path and set the value.

c. In **Select the endpoints**, search for and select an endpoint, for example **A_BusinessPartnerType**.

d. Select **Add entity**.
The flow named **GET SAP BAPIs and tables when system becomes active** runs to retrieve the data. The BAPI, table, and catalog tables are populated.

11. Select **Specify inputs** to check the information and edit as needed.
For more information, see Manage input parameters for an ERP Data Hub model operation.
12. Select **Choose output** to check the information and edit as needed.
For more information, see Choose output parameters for an ERP model.
13. Open the ERP systems list by selecting the systems icon ( ) in the side panel.
14. Select the system.
15. Check the heartbeat and retrieval status, and select the refresh list icon as needed until the heartbeats are successful and the retrieval status completed.
16. Create a model.
    a. Open the ERP models page by selecting the ERP models icon ( ) in the side panel.
    b. Select **New**.
    c. Add an **ERP model name**, **ERP module**, and **ERP system**, and select **Save**.
    d. Select the new model in the list.
    e. Select **Manage model**.
    f. Select **Add model operation**.
    g. Select a **Select type**.
    h. Select **Save and Continue**.
17. Select the new operation.
18. Select **Add entity**.
    a. In **Select type**, select **OData**.
    b. Select a **Select entity**, select an entity, for example **API_BUSINESS_PARTNER (Remote API for Business Partner) OData service catalog**.
    c. In **Select the endpoints**, search for and select an endpoint, for example **A_BusinessPartnerType Entity Name: Business Partner --- Return Type: Business Partner**.
    d. Select **Add entity** and wait for retrieval to complete.
19. Select **Specify inputs** to check the information and edit as needed.
    a. If there are required fields, select **Select mandatory fields**.
    b. Select any mandatory inputs listed and select **OK**.
    c. Select **Save**.
20. Select **Choose output**.

a. Select **+ New output**.

b. Select fields, for example **BusinessPartnerName** and **Full Name**.

c. Select **Save**.

21. Test with Workflow Studio.

a. Navigate to **All** > Workflow Studio.

b. On the homepage, select **Actions**.

c. In the **Name** column, filter for **contains Value** and type `Use ERP`.

d. Select **Apply**.

e. Select **Use ERP data**.

f. Select **Test**.

g. Specify the **ModelName**, **ModelOperation**, **Mandatory fields** (provide a value for the mandatory field if necessary), select a **System**, and select **Run Test**.

h. When complete, select the link **Your test has finished running. View the Action execution details**.

i. View and check the output, for example, in **Output Data**, select the **Response** to view the output.

Using ERP remote tables in ERP Data Hub

ERP (Enterprise Resource Planning) remote tables in ERP Data Hub enable you to view and query data from the ERP system of record on the ServiceNow AI Platform.

Note:
ERP Data Hub doesn't replicate data into the ServiceNow AI Platform. It mirrors data that lives in the ERP system of record, and remains protected there.

Remote tables are linked to ERP models

Each ERP remote table is connected to one ERP model, which links the remote table to the ERP system and other related values. When you build a remote table, you can add any of the fields that ERP Data Hub finds when scanning the ERP system of record, which is connected through the ERP model. Scanning for and adding fields ensures that all necessary fields are available, such as when using the table as a data source when building an app.

The connected ERP model, which is defined on the remote table **Details** tab, controls the available fields on the remote table. If you change the ERP model for a remote table, the available fields change as well.

Limitations of remote tables

To ensure data integrity, you can't create new remote tables in ERP Data Hub, but you can clone existing ERP models and extend standard ERP remote tables to customize them. For more information, see Clone an ERP model in ERP Data Hub.

ERP remote tables have a limit of 1,000 records. If you need a larger amount of data, use an ERP extraction table. For more information, see Extracting and transforming data in ERP Data Hub.

View and edit ERP remote table details with ERP Data Hub

View and edit details for ERP (Enterprise Resource Planning) remote tables in ERP Data Hub, such as their attachment settings and short descriptions.

Before you begin

Role required:

- To modify remote tables: sn_erp_integration.erp_admin
- To read remote tables: sn_erp_integration.erp_user

About this task

To ensure data integrity, you can't create new remote tables in ERP Data Hub, but you can clone existing ERP models and extend standard ERP remote tables to customize them. For more information, see Clone an ERP model in ERP Data Hub.

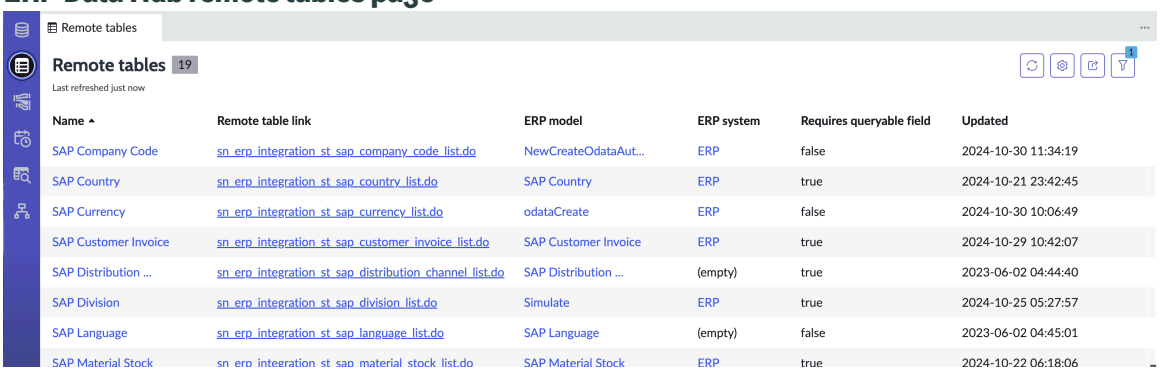
For information on adding and removing fields to and from remote tables, see Customize fields for an ERP remote table in ERP Data Hub.

ERP remote tables have a limit of 1,000 records. If you need a larger amount of data, use an ERP extraction table. For more information, see Extracting and transforming data in ERP Data Hub.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the Remote tables page by selecting the remote tables icon ( ) in the side panel.

ERP Data Hub remote tables page



3. View the details for a remote table by selecting the remote table **Name**.
4. **Optional:** Refresh the data stored in the remote table attachment outside of the scheduled cycle by selecting the **Refresh attachment data** button.
   The remote table attachment is the cached response from the ERP system. If the remote table **Attachment setting** field is set to **Use attachment**, ERP Data Hub doesn't call the ERP system when someone fetches data from the remote table.
5. Edit the remote table details as needed.
   For a description of the field values, see ERP Data Hub remote table form field descriptions.
6. View the fields in the remote table by selecting the **Remote table fields** tab.
7. **Optional:** View the ERP remote table as a list on the ServiceNow AI Platform by selecting the **Open remote table list** button.
8. Select **Save**.

Customize fields for an ERP remote table in ERP Data Hub

Add or remove columns in remote tables in ERP Data Hub to create your ERP (Enterprise Resource Planning) model. For example, remove fields with sensitive information, such as birthdays.

Before you begin

Role required:

- To modify remote tables: sn_erp_integration.erp_admin
- To read remote tables: sn_erp_integration.erp_user

About this task

ERP Data Hub scans the system of record for the ERP model to find all available fields that you can add to a remote table. You can view all available fields from the ERP system in the ERP model. Using the **Manage fields** modal to add available columns from the ERP system to a remote table automatically creates them in the remote table.

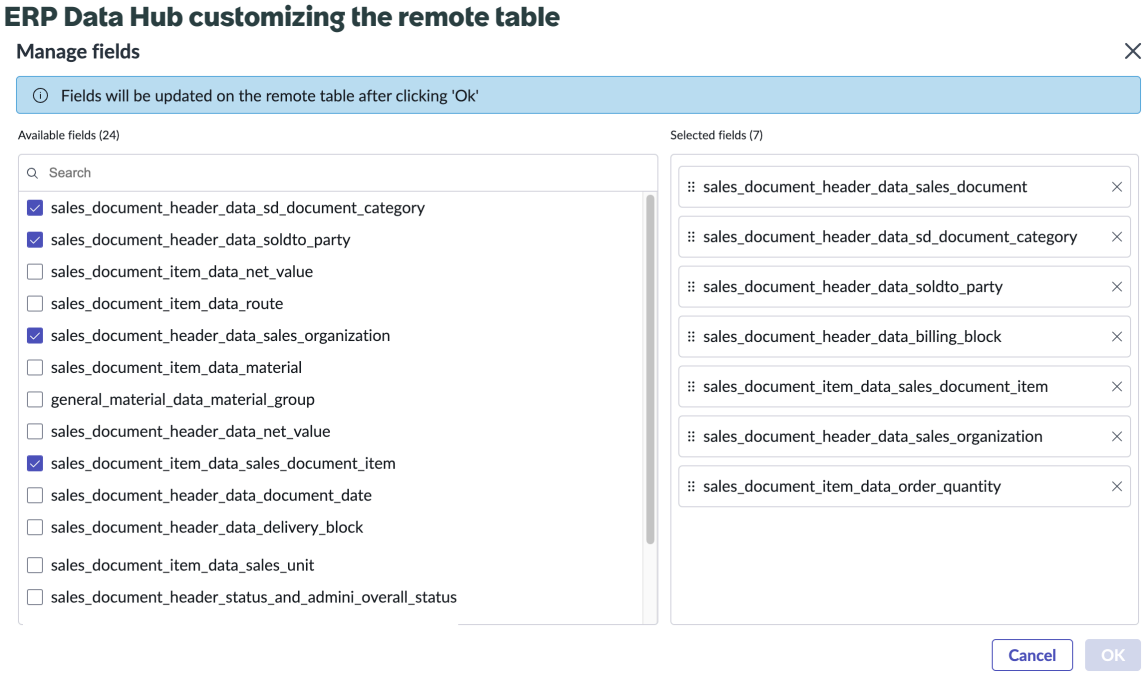
If you don't see the fields that you want to add to the remote table, you must first add them to the model. For more information, see Choose output parameters for an ERP model.

The connected ERP model, which is defined on the remote table **Details** tab, controls the available fields on the remote table. If you change the ERP model for a remote table, the available fields change as well.

Note:
You can't edit field names in remote tables, but you can add a new label for a field.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the Remote tables page by selecting the remote tables icon () in the side panel.
3. Select a table to work with by selecting the **Name**.
4. Manage the columns to build the remote table by selecting the **Manage fields** button.
   a. On the modal, find and select the field that you want to add to the remote table using the search box.
   b. Select the button to move the field from the **Available fields** list to the **Selected fields** list.
   c. Drag to rearrange how fields appear in the table.
   d. After you finish adding all the fields, select **OK** to save your changes.
   The ServiceNow AI Platform updates the fields on the remote table with your changes.



5. Confirm that the fields appear correctly as columns on the remote table by selecting the **Remote table fields** tab.

Query a remote table using ERP Data Hub

Query ERP (Enterprise Resource Planning) remote tables from a system of record directly from the **All** menu using ERP Data Hub.

Before you begin

Role required: sn_erp_integration.erp_user

About this task

You can query the system of record to create an ERP model for your ERP processes. For more information, see Building and managing ERP models to work with ERP data.

Note:
ERP Data Hub doesn't replicate data into the ServiceNow AI Platform. It mirrors data that lives in the ERP system of record, and remains protected there.

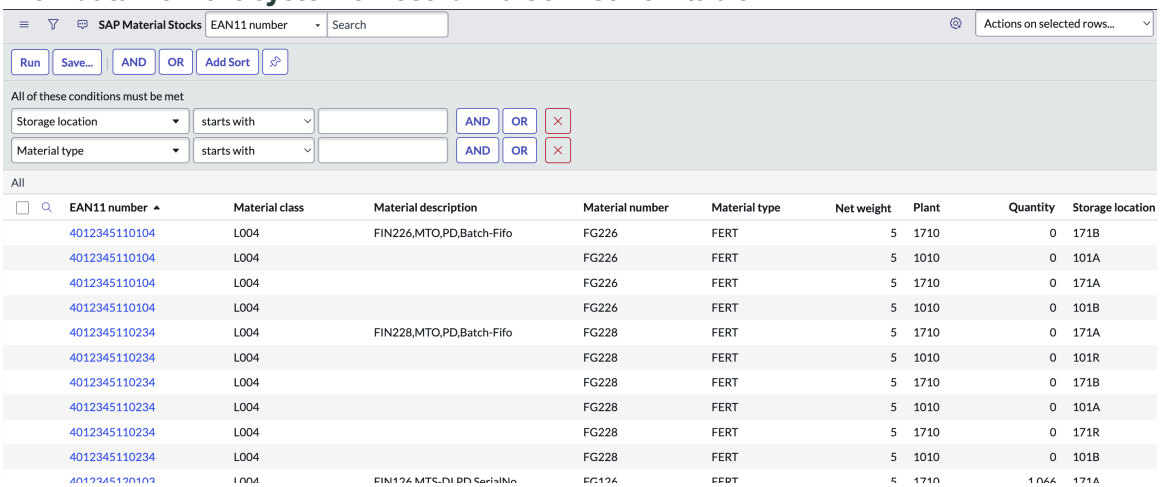
Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the Remote tables page by selecting the remote tables icon ( ) in the side panel.
3. Select the remote table from the navigation menu, such as **SAP Material Stock**.
4. Query the remote table using Glide for SAP.

   For example, you can select and hold (or right-click) on a column, such as **Storage location**, and select **Show Matching** records.

The SAP data is accessible for you to work with using standard ServiceNow AI Platform searching, sorting, and filtering, such as the condition builder. For more information, see Condition builder.

View data from the system of record in a ServiceNow table



EAN11 number	Material class	Material description	Material number	Material type	Net weight	Plant	Quantity	Storage location
4012345110104	L004	FIN226,MTO,PD,Batch-Fifo	FG226	FERT	5	1710	0	171B
4012345110104	L004		FG226	FERT	5	1010	0	101A
4012345110104	L004		FG226	FERT	5	1710	0	171A
4012345110104	L004		FG226	FERT	5	1010	0	101B
4012345110234	L004	FIN228,MTO,PD,Batch-Fifo	FG228	FERT	5	1710	0	171A
4012345110234	L004		FG228	FERT	5	1010	0	101R
4012345110234	L004		FG228	FERT	5	1710	0	171B
4012345110234	L004		FG228	FERT	5	1010	0	101A
4012345110234	L004		FG228	FERT	5	1710	0	171R
4012345110234	L004		FG228	FERT	5	1010	0	101B

Extracting and transforming data in ERP Data Hub

ERP (Enterprise Resource Planning) extraction tables enable you to create daily processes that extract large amounts of data. You can then include the extracted data in an ERP model in ERP Data Hub.

Hold large amounts of data in extraction tables

Use extraction tables, which are ETL (extract, transform, and load) data sources, to extract large amounts of data on a regular basis.

For example, you can extract all open purchase orders for the current month from an SAP system into a Glide table once a day at a set time. You can then access that data via ServiceNow AI Platform, such as Source-to-Pay Operations or a PO workspace created in App Engine Studio.

After the extraction process is run, use import sets to map imported data into ServiceNow AI Platform tables. For more information, see Import sets.

Extraction tables use transform table maps

ERP extraction tables save data to a local transform table on the ServiceNow AI Platform. The transform table is a temporary table that holds data during the data integration or transformation process. Transform tables are an intermediary step before data is further processed, cleaned, or loaded into the target destination. For example, after import, the extraction table can run a Glide query on a transform table and save the output to several different target tables.

If you use a custom transform table, you must first create the table on the ServiceNow AI Platform, and the table must be in the application scope. For more information on creating table transform maps, see Create a transform map.

ETL processes in ERP Data Hub are configured in the ServiceNow AI Platform app that needs the data, not ERP Data Hub. For example, you can configure an extraction process in a flow in Workflow Studio.

Automatically available standard extraction tables

When you install ERP Data Hub, the ServiceNow AI Platform automatically loads standard ERP extraction tables for you to work with. For example, sales, delivery, and procurement tables. For a list of standard extraction tables, see Standard ERP models and extraction tables for ERP Data Hub.

If you installed ERP Customization Mining, that app populates some additional extraction tables, such as ERP application activity, Collector directory data, and Namespace data.

View ERP extraction tables

Work with ETL (extract, transform, and load) processes in ERP Data Hub to extract large amounts of ERP (Enterprise Resource Planning) data from the ERP system. Extracted data is stored in Glide tables in the ServiceNow AI Platform.

Before you begin

Role required: sn_erp_integration.erp_user

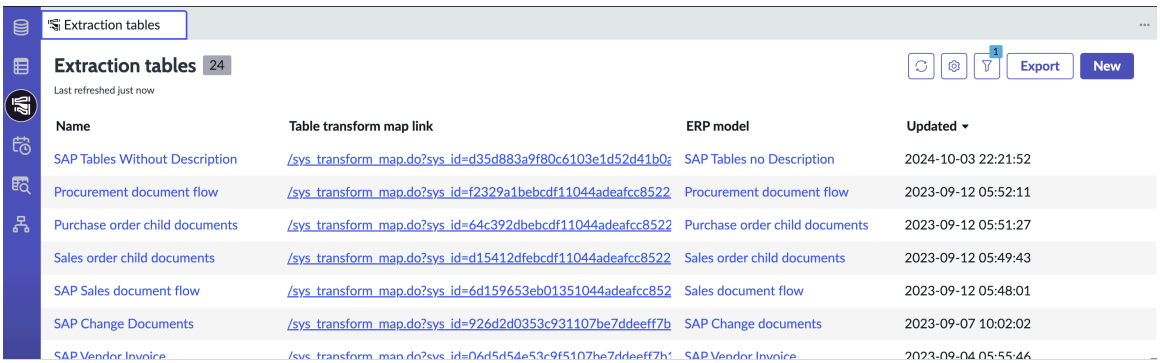
About this task

ERP extraction tables extract and save data to a transform table on the ServiceNow AI Platform. The transform table is a temporary table that holds data during the data integration or transformation process. Transform tables are an intermediary step before data is further processed, cleaned, or loaded into the target destination.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP extraction tables page by selecting the ERP extraction tables icon ( ) in the side panel.

Extraction tables in ERP Data Hub



3. View the list of ERP extraction tables and note of the tables that you can use to build your ERP model.

ETL source columns

Column	Description
Name	Name of the ETL process.
Short description	Brief description of the purpose of the ETL process.
Tables transform map link	Table that the extracted data is cached and stored in.

Column	Description
	Select the table name to view the table on the ServiceNow AI Platform in a new browser tab. For more information on creating table transform maps, see Create a transform map.
ERP model	ERP model used in the ETL extraction. Select the ERP model name to view the details. For more information, see View and edit the foundation of ERP models.

4. **Optional:** View the details for an extraction table by selecting the table name.

What to do next

After the extraction process is run, use import sets to map imported data into ServiceNow AI Platform tables. For more information, see Import sets.

Add a new ERP extraction table in ERP Data Hub

Create an ERP (Enterprise Resource Planning) extraction table to capture large amounts of data from the system of record every day, and save the data to a transformation (staging) table. The data is then available on the ServiceNow AI Platform, and you can add the extracted data to an ERP model or remote table.

Before you begin

You must first configure the source table, target table, and table transform map before those tables can be added to an ERP extraction table. For more information on creating table transform maps, see Create a transform map.

Role required:

- To modify extraction tables: sn_erp_integration.erp_admin
- To read extraction tables: sn_erp_integration.erp_user

About this task

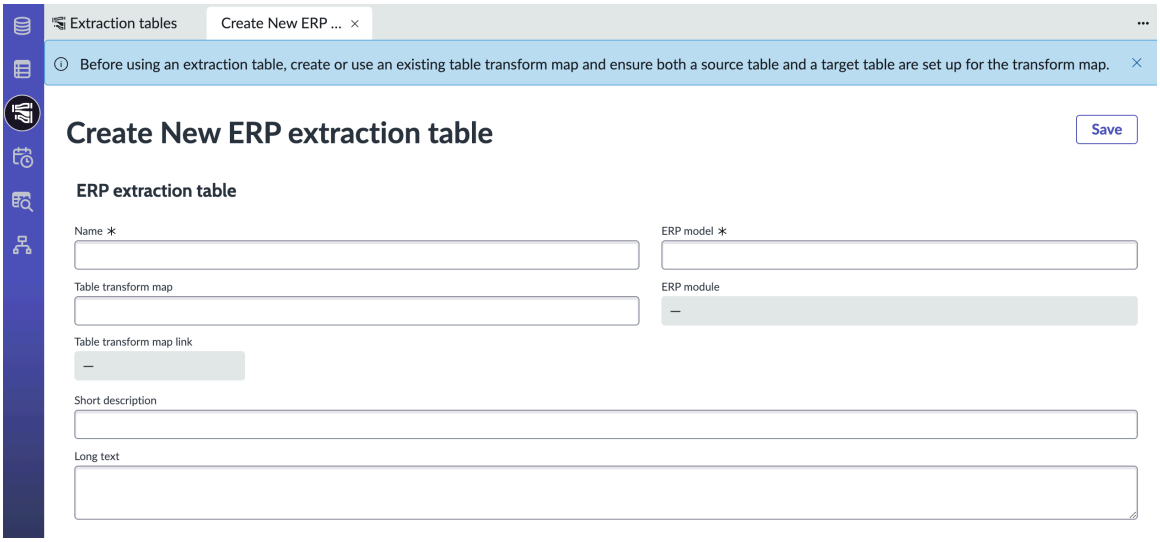
ERP Data Hub provides a number of standard extraction tables, which you can use as-is. If you must change a standard extraction table, copy the table and then update the copied version.

You can create multiple ERP extraction tables, and multiple extraction tables can use the same ERP model. For example, create separate extraction tables for sales contracts, sales inquires, and old sales orders.

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP extraction tables page by selecting the ERP extraction tables icon ( ) in the side panel.
3. Select the **New** button.

Add an extraction table in ERP Data Hub



4. On the form, fill in the fields.
   For a description of the field values, see ERP Data Hub extraction table field descriptions.
5. Select **Save**.

Select fields for an extraction table in ERP Data Hub

Add or remove fields for an extraction table in ERP Data Hub. For example, you may want to remove fields with sensitive information, such as birthdays.

Before you begin

If you don't see the fields that you want to add to the extraction table, you must first add them to the model. For more information, see Choose output parameters for an ERP model.

Role required:

- To modify extraction tables: sn_erp_integration.erp_admin
- To read extraction tables: sn_erp_integration.erp_user

Procedure

1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP extraction tables page by selecting the ERP extraction tables icon ( ) in the side panel.
3. Select an extraction table to work with by selecting the **Name**.
4. Manage the columns to build the extraction table by selecting the **Select fields** button.
   a. On the modal, find and select the field that you want to add to the extraction table using the search box.
   b. Select the button to move the field from the **Available columns** list to the **Selected columns** list.
   c. Drag to rearrange how fields appear in the table.

d. After you finish adding all the fields, select **OK** to save your changes.
The ServiceNow AI Platform updates the fields on the extraction table with your changes.

5. Confirm that the fields appear correctly as columns on the extraction table by selecting the **Extraction table fields** tab.

Create a scheduled extraction

Schedule extraction of information for an ERP (Enterprise Resource Planning) extraction table to capture large amounts of data from the system of record at a regular interval.

Before you begin

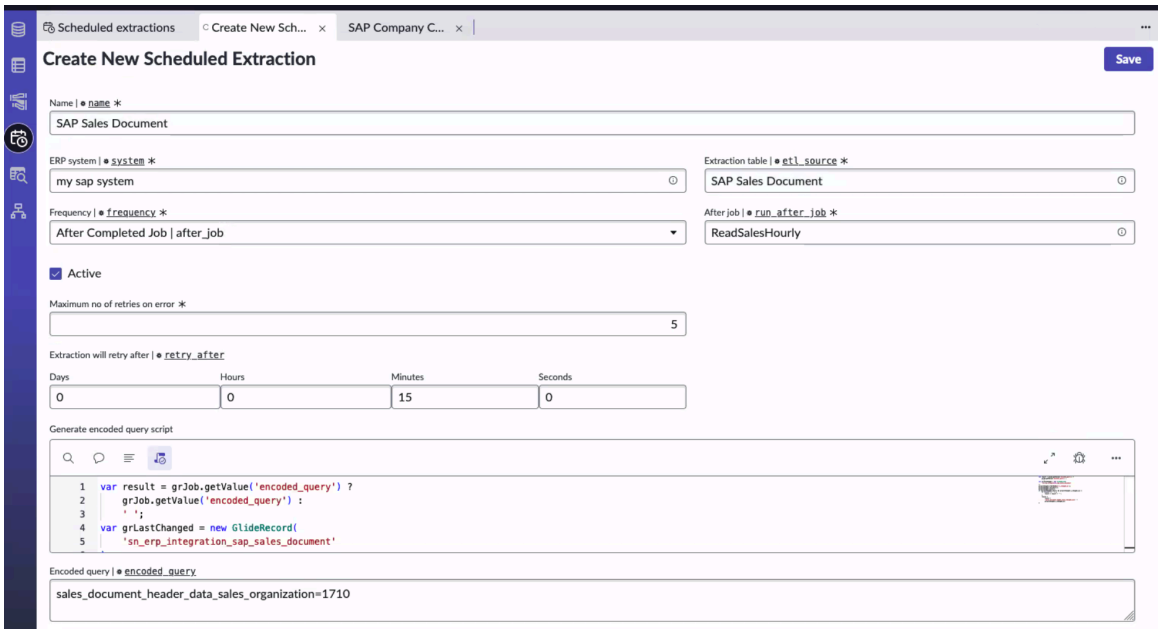
You must have a standard or custom ERP extraction table in place to use. For more information, see Add a new extraction table in ERP Data Hub.

The system property sn_erp_integration.enableJobModification must be set to true to enable creation and modification of scheduled jobs in ERP Data Hub.

Role required: en_erp_integration.erp_user

Procedure

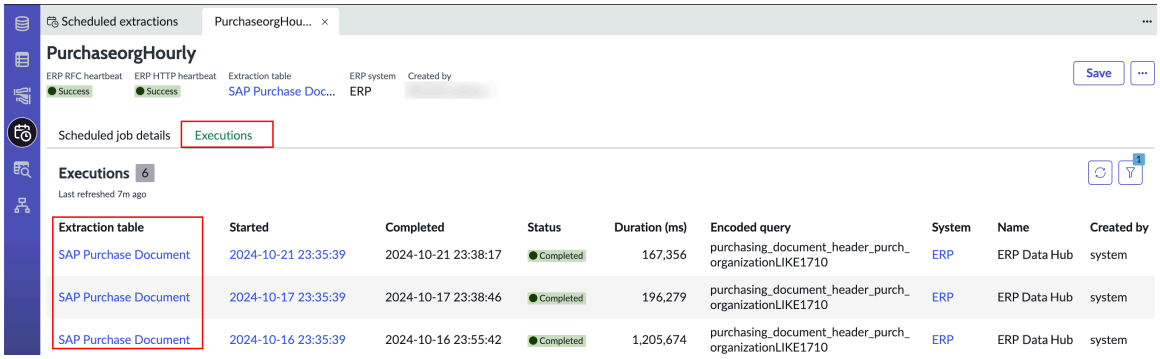
1. Navigate to **All** > **ERP Data Hub** > **ERP Data Hub Home**.
2. Open the ERP scheduled extractions page by selecting the Scheduled extractions icon ( ) in the side panel.
3. Select the **New** button.



4. On the form, fill in the fields.
For a description of the field values, see ERP Data Hub scheduled extraction field descriptions.
5. Select **Save**.

What to do next

Check the executions. After the scheduled job has run, select the **Executions** tab. For details about an extraction, select any line item in the **Extraction table** column.



Monitor ERP Data Hub logged extraction and remote lookup transactions

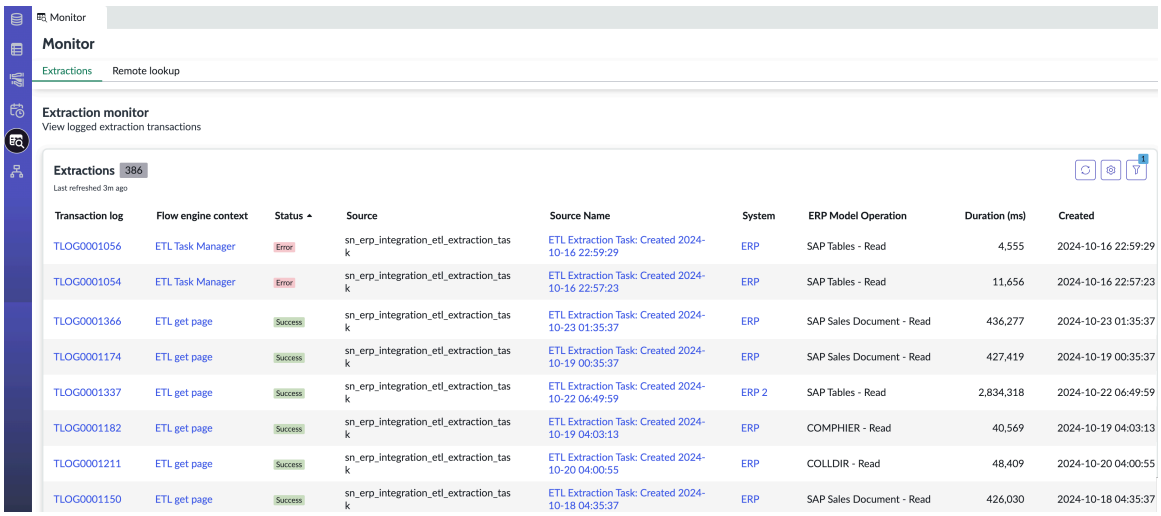
Use the new monitoring feature to track each ERP transaction and its progress. Filter the ERP Data Hub task information, such as successes, failures, and more, as needed.

Before you begin

Role required: admin and sn_erp_integration.erp_user

Procedure

1. Navigate to **All** > **ERP Data Hub**.
2. Open the ERP monitor page by selecting the Monitor icon ( ) in the side panel.
3. Select the **Extractions** and **Remote lookup** tabs to view transaction information.



For a description of the field values, see ERP Data Hub Monitor field descriptions.

ERP Data Hub reference

Find reference information for ERP Data Hub, including ERP (Enterprise Resource Planning) table details.

ERP Data Hub and domain separation

Domain separation is unsupported for ERP Data Hub (Enterprise Resource Planning). Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Support level: No support

- The domain field may exist on data tables but there is no business logic to manage the data.
- This level is not considered domain-separated.

For more information on support levels, see Application support for domain separation.

Related topics

Domain separation for service providers

Sample Glide query for ERP data in ERP Data Hub

Access data from the ERP (Enterprise Resource Planning) system of record through the Glide API.

The following is an example of a Glide query that fetches a customer name.

```
var sap_customer_gr = new
 GlideRecord('sn_erp_integration_st_sap_sales_customer');
sap_customer_gr.get('customer_number', '100032');
sap_customer_gr.getValue('name');
```

ERP Data Hub standard tables, fields, and models

Find details on standard ERP (Enterprise Resource Planning) remote tables, extraction tables, fields, and models in ERP Data Hub.

Standard remote tables for ERP Data Hub

ERP Data Hub accesses several standard remote tables for ERP (Enterprise Resource Planning) models.

The following remote tables are available through ERP Data Hub and ERP-CM.

Remote tables for ERP Data Hub and ERP-CM

Label	Name	ERP module
SAP Company Code	sn_erp_integration_st_sap_company_code	Basis
SAP Country	sn_erp_integration_st_sap_country	Basis
SAP Currency	sn_erp_integration_st_sap_currency	Basis
SAP Language	sn_erp_integration_st_sap_language	Basis
SAP Material Stock	sn_erp_integration_st_sap_material_stock	Plant maintenance
SAP Purchase Document	sn_erp_integration_st_sap_purchase_document	Purchasing document
SAP Purchasing Organization	sn_erp_integration_st_sap_purchasing_organization	Purchasing organization
SAP Customer Invoice	sn_erp_integration_st_sap_customer_invoice	Sales invoice
SAP Distribution Channel	sn_erp_integration_st_sap_distribution_channel	Sales
SAP Division	sn_erp_integration_st_sap_division	Sales
SAP Sales Customer	sn_erp_integration_st_sap_sales_customer	Sales
SAP Sales Delivery	sn_erp_integration_st_sap_sales_delivery	Sales
SAP Sales Document	sn_erp_integration_st_sap_sales_document	Sales

Remote tables for ERP Data Hub and ERP-CM (continued)

Label	Name	ERP module
SAP Sales Organization	sn_erp_integration_st_sap_sales_org	Sales_organization
SAP Sales Revenue Recognition	sn_erp_integration_st_sap_sales_rev	Sales_revenue_recognition
SAP Vendor	sn_erp_integration_st_sap_vendor	Sales
SAP Vendor Invoice	sn_erp_integration_st_sap_vendor_inv	Sales_invoice
SAP Transport	sn_erp_integration_st_sap_transport	Transport

For more details on working with remote tables, see Remote tables.

You can use any of the standard remote tables as data sources when building apps in ServiceNow products, such as:

- App Engine Studio
- Flows in Workflow Studio
- Playbooks in Workflow Studio
- Table Builder
- UI Builder
- Workspace Builder

You can also access data from the system of record through the Glide API.

The following is an example of a Glide query that fetches a customer name.

```
var sap_customer_gr = new
 GlideRecord('sn_erp_integration_st_sap_sales_customer');
sap_customer_gr.get('customer_number', '100032');
sap_customer_gr.getValue('name');
```

Standard ERP Data Hub fields within remote tables

The standard ERP (Enterprise Resource Planning) remote tables available for use in ERP Data Hub contain fields from additional SAP tables.

The standard remote tables contain the following additional fields. For details on the standard tables, see Standard remote tables for ERP Data Hub.

Standard ERP Data Hub and ERP-CM tables within remote tables

Remote table	Source table	ERP field name	Mapped field name
SAP Sales Document	VBAK	VBELN	document_number
SAP Sales Document	VBAK	ERDAT	date_of_document
SAP Sales Document	VBAK	ERZET	time_of_document
SAP Sales Document	VBAK	VKORG	sales_organization
SAP Sales Document	VBAK	VBTYP	document_category

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Sales Document	VBAK	AUART	document_type
SAP Sales Document	VBAK	AUGRU	order_reason
SAP Sales Document	VBAK	LIFSK	delivery_block
SAP Sales Document	VBAK	FAKSK	billing_block
SAP Sales Document	VBAK	KUNNR	customer_number
SAP Sales Document	VBAK	AUFNR	order_number
SAP Sales Document	VBAK	NETWR	document_value
SAP Sales Document	VBAK	WAERK	currency_code
SAP Sales Document	VBUK	LFGSK	delivery_status
SAP Sales Document	VBAP	MATNR	material_number
SAP Sales Document	VBAP	ARKTX	material_description
SAP Sales Document	VBAP	KWMENG	ordered_quantity
SAP Sales Document	VBAP	KLMENG	confirmed_quantity
SAP Sales Document	VBAP	NETWR	item_value
SAP Sales Document	VBAP	VRKME	sales_unit
SAP Sales Document	VBAP	ROUTE	delivery_route
SAP Sales Document	MARA	MTART	material_type
SAP Sales Document	VBUP	GBSTA	overall_item_status
SAP Sales Customer	KNA1	KUNNR	customer_number
SAP Sales Customer	KNA1	LAND1	country_code
SAP Sales Customer	KNA1	NAME1	name
SAP Sales Customer	KNA1	ORT01	city
SAP Sales Customer	KNA1	PSTLZ	postal_code
SAP Sales Customer	KNA1	STRAS	street
SAP Sales Customer	KNA1	STKZU	vat_liable
SAP Sales Customer	KNA1	STCEG	vat_reg_number
SAP Sales Customer	KNVV	VKORG	sales_organization
SAP Sales Customer	KNVV	VTWEG	distribution_channel
SAP Sales Customer	KNVV	SPART	division
SAP Sales Customer	KNVV	INCO1	inco_terms
SAP Customer Invoice	VBRK	VBELN	document_number
SAP Customer Invoice	VBRK	FKART	billing_type
SAP Customer Invoice	VBRK	WAERK	currency_code

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Customer Invoice	VBRK	ZTERM	payment_terms
SAP Customer Invoice	VBRK	NETWR	document_value
SAP Customer Invoice	VBRK	KUNRG	payer
SAP Customer Invoice	VBUK	GBSTK	overall_document_status
SAP Customer Invoice	VBRP	MATNR	material_number
SAP Customer Invoice	VBRP	ARKTX	material_description
SAP Customer Invoice	VBRP	NETWR	item_value
SAP Customer Invoice	VBRP	FKLMG	billing_quantity
SAP Customer Invoice	VBRP	VRKME	sales_unit
SAP Customer Invoice	VBRP	SHKZG	is_returns_item
SAP Customer Invoice	VBUP	GBSTA	overall_item_status
SAP Sales Organization	TVKO	VKORG	sales_organization
SAP Sales Organization	TVKO	EKORG	purchase_organization
SAP Sales Organization	TVKO	BUKRS	company_code
SAP Purchase Document	EKKO	EBELN	document_number
SAP Purchase Document	EKKO	BUKRS	company_code
SAP Purchase Document	EKKO	BSTYP	document_category
SAP Purchase Document	EKKO	LIFNR	vendor_number
SAP Purchase Document	EKKO	ZTERM	payment_terms
SAP Purchase Document	EKKO	EKORG	purchase_organization
SAP Purchase Document	EKKO	WAERS	currency_code
SAP Purchase Document	EKKO	IHREZ	ext_reference
SAP Purchase Document	EKKO	RESWK	supplying_plant
SAP Purchase Document	EKKO	BEDAT	date_of_document
SAP Purchase Document	EKPO	MATNR	material_number
SAP Purchase Document	EKPO	EMATN	supplier_material_number
SAP Purchase Document	EKPO	MENGE	ordered_quantity
SAP Purchase Document	EKPO	MEINS	order_unit
SAP Purchase Document	EKPO	NETPR	item_value
SAP Purchase Document	EKPO	MWSKZ	vat_applicable
SAP Purchase Document	EKPO	ELIKZ	fully_delivered
SAP Purchase Document	EKPO	REPOS	fully_invoiced
SAP Material Stock	MARA	MATNR	material_number

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Material Stock	MARA	MTART	material_type
SAP Material Stock	MARA	MATKL	material_class
SAP Material Stock	MARA	NTGEW	net_weight
SAP Material Stock	MARA	EANNR	ean_number
SAP Material Stock	MARA	EAN11	ean11_number
SAP Material Stock	MARA	MSTAE	material_status
SAP Material Stock	MAKT	MAKTX	material_description
SAP Material Stock	MARD	WERKS	plant
SAP Material Stock	MARD	LGORT	storage_location
SAP Material Stock	MARD	LABST	quantity
SAP Material Stock	MARD	DLINL	date_of_count
SAP Vendor Invoice	RSEG	BELNR	document_number
SAP Vendor Invoice	RSEG	GJAHR	document_year
SAP Vendor Invoice	RSEG	EBELN	purchase_document_number
SAP Vendor Invoice	RSEG	EBELP	purchase_document_item
SAP Vendor Invoice	RSEG	MATNR	material_number
SAP Vendor Invoice	RSEG	BUKRS	company_code
SAP Vendor Invoice	RSEG	WERKS	plant
SAP Vendor Invoice	RSEG	WRBTR	amount
SAP Vendor Invoice	RSEG	SHKZG	credit_debit_indicator
SAP Vendor Invoice	RSEG	MWSKZ	vat_applicable
SAP Vendor Invoice	RSEG	MENGE	quantity
SAP Purchasing Organization	T024E	EKORG	purchase_organization
SAP Purchasing Organization	T024E	EKOTX	name
SAP Purchasing Organization	T024E	BUKRS	company_code
SAP Sales Delivery	LIKP	VBELN	document_number
SAP Sales Delivery	LIKP	KUNNR	customer_number
SAP Sales Delivery	LIKP	ERDAT	date_of_document
SAP Sales Delivery	LIKP	ERZET	time_of_document
SAP Sales Delivery	LIKP	VKORG	sales_organization
SAP Sales Delivery	LIKP	LFART	delivery_type
SAP Sales Delivery	LIKP	ROUTE	delivery_route
SAP Sales Delivery	VBUK	GBSTK	overall_document_status

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Sales Delivery	LIPS	MATNR	material_number
SAP Sales Delivery	LIPS	CHARG	batch_number
SAP Sales Delivery	LIPS	WERKS	plant
SAP Sales Delivery	LIPS	LFIMG	quantity
SAP Sales Delivery	LIPS	NTGEW	net_weight
SAP Sales Delivery	LIPS	GEWEI	delivery_unit
SAP Sales Delivery	LIPS	VOLUM	volume
SAP Sales Delivery	LIPS	VOLEH	volume_unit
SAP Sales Delivery	LIPS	ARKTX	material_description
SAP Vendor	LFA1	LIFNR	vendor_number
SAP Vendor	LFA1	NAME1	name
SAP Vendor	LFA1	ORT01	city
SAP Vendor	LFA1	PSTLZ	postal_code
SAP Vendor	LFA1	STRAS	street
SAP Vendor	LFA1	STCEG	vat_reg_number
SAP Vendor	LFA1	WERKS	plant
SAP Vendor	LFM1	EKORG	purchase_organization
SAP Vendor	LFM1	WEBRE	gr_invoice_indicator
SAP Sales Revenue Recognition	VBREVK	VBELN	document_number
SAP Sales Revenue Recognition	VBREVK	POSNR	document_item
SAP Sales Revenue Recognition	VBREVK	SAKRR	accr_val_clearing_account_number
SAP Sales Revenue Recognition	VBREVK	SAKRRK	offset_clearing_account_number
SAP Sales Revenue Recognition	VBREVK	ACC_VALUE	total_accrued_value
SAP Sales Revenue Recognition	VBREVK	WRBTR	amount
SAP Sales Revenue Recognition	VBREVK	RVAMT	revenue_amount
SAP Sales Revenue Recognition	VBREVK	WAERK	currency_code
SAP Country	T005	LAND1	country_code
SAP Country	T002	SPRAS	language_code

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Country	T002	LAISO	language_iso_code
SAP Country	T005T	LANDX	description
SAP Language	T002	SPRAS	language_code
SAP Language	T002	LAISO	language_iso_code
SAP Currency	TCURC	WAERS	currency_code
SAP Currency	TCURC	ISOCD	currency_iso_code
SAP Currency	TCURT	LTEXT	description
SAP Distribution Channel	TVTW	VTWEG	distribution_channel
SAP Distribution Channel	TVTWT	VTEXT	description
SAP Division	TSPA	SPART	division
SAP Division	TSPAT	VTEXT	description
SAP Company Code	T001	BUKRS	company_code
SAP Company Code	T001	BUTXT	description
SAP Company Code	T001	LAND1	country_code
SAP Company Code	T001	WAERS	currency_code
SAP Company Code	T001	SPRAS	language_code
SAP Transport	E071	AS4DATE	date_of_transport
SAP Transport	E071	AS4TIME	time_of_transport
SAP Transport	E071	AS4USER	user
SAP Transport	E071	TRFUNCTION	type
SAP Transport	E071	TRKORR	number
SAP Transport	E071	TRSTATUS	status
SAP Transport	E071	OBJECT	object_type
SAP Transport	E071	OBJ_NAME	object_name
SAP Transport	E071	PGMID	program_id

Standard ERP models and extraction tables for ERP Data Hub

ERP Data Hub provides a number of standard extraction tables that you can link to ERP (Enterprise Resource Planning) models.

ERP models function as templates for sets of tables that give you access to ERP data. You can use the standard ERP Data Hub models as-is, or clone them to make changes.

The following extraction tables are available through ERP Data Hub.

Extraction tables for ERP Data Hub

Label	ERP module
SAP Company Code	Basis
SAP Country	Basis
SAP Currency	Basis
SAP Language	Basis
SAP Material Stock	Procurement
SAP Purchase Document	Procurement
SAP Purchasing Organization	Procurement
SAP Customer Invoice	Sales
SAP Distribution Channel	Sales
SAP Division	Sales
SAP Sales Customer	Sales
SAP Sales Delivery	Sales
SAP Sales Document	Sales
SAP Sales Organization	Sales
SAP Sales Revenue Recognition	Sales
SAP Vendor	Sales
SAP Vendor Invoice	Sales
SAP Transport	Transport

For more details on working with remote tables, see Remote tables.

ERP Data Hub field descriptions

Some tables of field descriptions in ERP Customization Mining (ERP-CM) are too large to maintain in task topics. Find information on the those large tables in this section.

ERP Data Hub connection and credentials field descriptions

The Connection and Credential Alias modal contains connection and credential details that specify how ERP Data Hub connects to the ERP (Enterprise Resource Planning) system.

Enter the Connection Information field descriptions

Enter the Connection Information field descriptions

Field	Description
Connection Name	Label or alias for the connection.
Host name or IP Address to SAP Application server or the Load Balancer	Host name or IP address of the ERP system server or load balancing host that connects to the ERP system.
Login Language	Two-character language code of the language of the ERP server you're connecting to.

Enter the Connection Information field descriptions (continued)

Field	Description
SAP Client	Client value of the connection host where the SAP instance is installed.
SAP System Number	System number of the connection host where the SAP instance is installed. Don't enter anything in this field if the connection you're specifying is for a load balancer.
Load Balancing: SAP Message Server Port number or logical name	Port number or name of the message server host. Enter this value only if the connection you're specifying is for a load balancer.
Load Balancing: SAP R3 Name	Three-character system ID of the ABAP system to be addressed. Enter this value only if the connection you're specifying is for a load balancer.
Load Balancing: SAP Load Balancing Group	Name of load balancer. Enter this value only if the connection you're specifying is for a load balancer.

Enter the Credential Information field descriptions

Enter the Credential Information field descriptions

Field	Description
User Name	User name to log in to the system of record to access ERP data.
Password	Password to authenticate and log in to the system of record.
Credential Name	Alias or name for the login credential.

ERP Data Hub new system field descriptions

The Create new system form in ERP Data Hub contains information on connection details for the ERP (Enterprise Resource Planning) system.

Create new ERP system fields

Field	Description
ERP system	Name of the ERP system to help identify the business area.
Short description	Brief description of what the ERP system is for, such as sales orders.

Create new ERP system fields (continued)

Field	Description
Connection (HTTP connection available starting in Xanadu Store Release 2)	Alias of the connection credential that you configured to connect to the system of record. You can select only from connections in the ERP Data Hub scope. For more information, see Configure the ERP Data Hub credentials and connection.
Latest alive heartbeat (HTTP heartbeat available starting in Xanadu Store Release 2)	Date and time that the ServiceNow AI Platform most recently connected to the ERP system. This field isn't editable and has a value only if the ERP system has been successfully saved and contacted.
ETL data sources/Extraction tables	Number of ERP extraction tables in the ERP system. Link a system to an extraction table by adding the system to the ERP extraction table record in ERP Data Hub. For more information, see Add a new ERP extraction table in ERP Data Hub.
Remote tables	Number of remote tables in the ERP system. Link a system to a remote table by adding the system to the ERP remote table record in ERP Data Hub. For more information, see View and edit ERP remote table details with ERP Data Hub.
Updated	Date and time the ERP system record was most recently saved.
KB link (on heartbeat tabs)	If there's an error and a relevant knowledge base article is available, a link to the article is provided.
Status (on heartbeat tabs)	State of the heartbeat: **Started**, **Success**, or **Error**.
Error text (on heartbeat tabs)	Details about the error.
Updated (on heartbeat tabs)	Date and time when the heartbeat was last changed.

ERP Data Hub system list field descriptions

The Create new system form in ERP Data Hub contains information on connection details for the ERP (Enterprise Resource Planning) system.

ERP system list details

Column	Description
ERP system	Name of the ERP system.
ERP heartbeat (HTTP connection available starting in Xanadu Store Release 2)	Latest status of the SAP system connection, either **Success** or **Error**. The connection is checked every 5 minutes automatically.
Retrieval status (OData available starting in Xanadu Store Release 2)	When the system is first set up and connects to the SAP system, metadata is retrieved. • BAPI/RFC: For BAPI, a list of functions available to call on the system are collected. For RFC, a check is done to determine the tables available on the database. • Table: The tables from the database are retrieved. • Odata: The models available on the HTTP protocol are retrieved.

ERP Data Hub ERP model table field descriptions

The **Entity fields** tab for an ERP (Enterprise Resource Planning) model in ERP Data Hub displays the table fields that are included in the ERP model.

ERP Data Hub automatically scans the linked ERP system to retrieve the latest entity data. However, you can select the refresh icon to update the data on demand.

Entity fields tab details

Column	Description
Field name	Name of the field on the system of record.
Name	Name of the table on the system of record that contains the field.
Mapped field name	Name of the mapped field. The ServiceNow AI Platform automatically creates this mapped field name in the ERP model after scanning the linked tables on the system of record.
Field label	Natural language field name for the automatically mapped field on the ServiceNow AI Platform.
Is custom	Option to indicate whether the field on the system of record is standard or customized.
ERP data type	Type of data the field contains. The options are: • char (character) • date

Entity fields tab details (continued)

Column	Description
	• decimal • numc (number) • time
Is queryable	Option to indicate whether you can retrieve data from the field without querying the source table. Smaller tables, such as a currency table, can have data stored locally and thus don't require a query to retrieve fields.

ERP Data Hub clone model field descriptions

The Clone ERP (Enterprise Resource Planning) model form in ERP Data Hub enables you to clone an ERP model so you can edit the new model.

ERP model fields

Field	Description
ERP model name	Name of the ERP model.
ERP module	ERP module in the system of record. For example, sales orders or inventory. ERP modules represent a distinct set of features and functionalities tailored to address business processes or activities.
ERP system	ERP system the ERP model connects to. The connected ERP system enables access to information about fields and tables and interaction between the model and the connected ERP system. For more information, see Create an ERP system in ERP Data Hub and View a list of ERP Data Hub systems.
Application	Application scope the ERP model is linked to. For example, if you create a custom model, the model will be in that scope.
Short description	Brief description of what the ERP model represents.
Long text	Any longer description or information about the ERP data model.
Work notes	Notes on the ERP model. Work notes are private, and not viewable by other ServiceNow AI Platform users.

ERP model fields (continued)

Field	Description
Comments	Comments on the ERP model. Comments are public, and visible to other ServiceNow AI Platform users.

ERP Data Hub new model field descriptions

The Create new system form in ERP Data Hub contains details for the ERP (Enterprise Resource Planning) model.

ERP model fields

Field	Definition
ERP model name	Name of the ERP model.
ERP module	ERP module in the system of record, for example, sales orders or inventory. ERP modules represent a distinct set of features and functionalities tailored to address business processes or activities.
ERP system	ERP system the ERP model connects to. The connected ERP system enables access to information about field and tables and interaction between the model and the connected ERP system. For more information, see Create an ERP system in ERP Data Hub and View a list of ERP Data Hub systems.
Application	Application scope the ERP model is linked to. For example, if you create a custom model, the model will be in that scope.
Short description	Brief description of what the ERP model represents.
Long text	Any longer description or information about the ERP data model.
Work notes	Notes on the ERP model. Work notes are private, and not viewable by other ServiceNow AI Platform users.
Comments	Comments on the ERP model. Comments are public, and visible to other ServiceNow AI Platform users.

ERP Data Hub remote table form field descriptions

The Remote table form in ERP Data Hub enables you to create and edit remote tables in the ERP (Enterprise Resource Planning) model.

Remote table details

Field	Description
Name	Name of the remote table on the ServiceNow AI Platform.
ERP model	ERP model connected to the remote table. The ERP model controls the available fields on the remote table. **Note:** Changing the ERP model removes any previously configured table fields that aren't available on the new model. To change the model for an ERP remote table, you must select the **Unlock ERP model field** button and then select the **Unlock model change** button on the confirmation dialog.
ERP module	Functional area or component within an ERP system. The module represents a distinct set of features and functionalities tailored to address a set of business processes or activities.
ERP system	ERP system that the remote table is linked to. The connected system represents the ERP instance that is linked to the model, enabling data flow and interaction between the model and the connected ERP system. For more information, see Create an ERP system in ERP Data Hub.
Remote table link	Name of and link to the remote table on the ServiceNow AI Platform. Selecting the remote table link opens a new browser tab, where you can explore, query, and manage the remote table.
Attachment setting	How the remote table uses attachments, which contain the remote data from the ERP system. The options are: • **None**: No attachment is used for the remote table. • **Save attachment**: Generate a new attachment when specified in the scheduled query or by selecting the **Refresh attachment data** button. • **Use attachment**: The current attachment, if available, is used locally.

Remote table details (continued)

Field	Description
Short description	Brief description of the ERP remote table.
Long text	Additional details on the contents of the remote table.
Requires queryable field	Indicates whether you can retrieve data from the remote table without querying the source table. Smaller tables, such as a currency table, can have data stored locally and thus don't require a query to retrieve fields.
Enable schedule	Option to create a scheduled query (ETL extraction) to fetch a new version of the attachment data. **Note:** The **Attachment setting** field must be set to **Save attachment** or **Use attachment** for this option to be available.
Daily schedule time	Time, in 24-hour format, for when the schedule runs every day.
Scheduled encoded query	Details on any scheduled extraction queries for the remote table. Create an encoded query string using a filter on the remote table list and paste the string into the **Encoded query** field. For more information on creating encoded queries, see Encoded query strings.

ERP Data Hub extraction table field descriptions

The Extraction table form in ERP Data Hub enables you to create and edit extraction tables in the ERP (Enterprise Resource Planning) model.

New ERP extraction table fields

Field	Description
Name	Name of the ERP extraction table.
ERP model	ERP model for the extraction table source, which must be configured for you to select.
Table transform map	Table that the extracted data is cached and stored in. The transform table is a temporary table that holds data during the data integration or transformation process. Transform tables are an intermediary step before data is further

New ERP extraction table fields (continued)

Field	Description
	processed, cleaned, or loaded into the target destination. The specified Glide table connects the source ERP extraction table and the target table for the extracted data. If you use a custom transform table, you must first create the table on the ServiceNow AI Platform, and the table must be in the application scope. For more information on creating table transform maps, see Create a transform map.
ERP module	Name of the ERP module linked to the ERP extraction table, which is controlled by the ERP model.
Table transform map link	Link to the Glide table that the extracted ERP data is cached and stored in. After you save the extraction table, when you select the link, the ServiceNow AI Platform table opens in a new browser tab.
System	ERP system that the ETL (extract, transform, and load) process extracts data from on the system of record. The system must already be configured.
Short description	Brief description of the ETL source.
Long text	Any additional text to describe the ERP extraction table.
Comments	Public comment that is visible to anyone working on the ERP extraction table.
Work notes	Private notes that only you can view.

ERP Data Hub scheduled extraction field descriptions

The Scheduled extraction form in ERP Data Hub enables you to create and edit jobs to extract data at regular intervals.

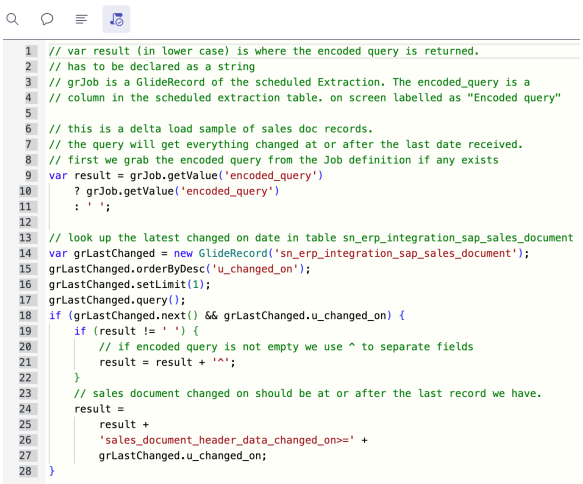
ERP scheduled extraction table fields

Field	Description
Name	A unique name that identifies this scheduled extraction.
ERP system	ERP system that the extraction table is linked to. The system must already be configured.
Extraction table	Name of the ERP extraction table from which to pull data.

ERP scheduled extraction table fields (continued)

Field	Description
Frequency	When to run the extraction. • Hourly: Specify the next scheduled start in hours, minutes, and seconds to repeat hourly. For example, adding 20:30:00 starts the scheduled extraction the next time the clock reaches 20:30:00 (9:30 pm) and then hourly after that (21:30:00, 22:30:00, 23:30:00, and so on). • Daily: Specify the next scheduled start in hours, minutes, and seconds to repeat daily. For example, adding 20:30:00 starts the scheduled extraction the next time the clock reaches 20:30:00 (9:30 pm) and then every subsequent day at 20:30:00. • Weekly: In **Weekday**, select the day of the week to run the scheduled extraction. Specify the next scheduled start in hours, minutes, and seconds to repeat weekly. For example, selecting Sunday and specifying 03:00:00 starts the scheduled extraction the next Sunday at 3:00 am and then every subsequent week on Sunday at 3:00 am. • After Completed Job: In **After job**, select an existing job that, when it completes, the new job you're creating should start. The job you specify must be active. **Note:** If you don't see the job that you want to use listed in **After job**, check if the job is already scheduled before or after another job. Only jobs that won't create a loop are permitted.
Next scheduled start	Time, in 24-hour format, for when the scheduled extraction should run.
Active	Option that indicates the scheduled extraction is active and should be executed at the specified frequency and time.
Maximum no of retries on error	If the job fails, set the maximum number of retries (from 0 through 10) the scheduled job should attempt before stopping.

ERP scheduled extraction table fields (continued)

Field	Description
	Note: Each retry uses the same query and retries the entire job. For example, if the total job contains 5000 records and the job fails after 2000 records are successfully processed, the entire job is rerun again on the next retry. For more information, see Import sets key concepts.
Extraction will retry after	If the **Maximum no of retries on error** field is set to a number between 1-10, specify the amount of time (in days, hours, minutes, and seconds) the system should wait before attempting the extraction again.
Generate encoded query script	Generate an encoded query script to use on the extraction table to fetch the data. For example:  The script entered in **Generate encoded query script** takes precedence over information entered into the **Encoded query** field. You can append the encoded query to the script (as in the example).
Encoded query	Create an encoded query string using a filter on the extraction table list and paste the string into this field. For example:

```
// var result (in lower case) is where the encoded query is returned.
// has to be declared as a string
// grJob is a GlideRecord of the scheduled Extraction. The encoded_query is a
// column in the scheduled extraction table. on screen labelled as "Encoded query"

// this is a delta load sample of sales doc records.
// the query will get everything changed at or after the last date received.
// first we grab the encoded query from the Job definition if any exists
var result = grJob.getValue('encoded_query')
    ? grJob.getValue('encoded_query')
    : ' ';

// look up the latest changed on date in table sn_erp_integration_sap_sales_document
var grLastChanged = new GlideRecord('sn_erp_integration_sap_sales_document');
grLastChanged.orderByDesc('u_changed_on');
grLastChanged.setLimit(1);
grLastChanged.query();
if (grLastChanged.next() && grLastChanged.u_changed_on) {
    if (result != ' ') {
        // if encoded query is not empty we use ^ to separate fields
        result = result + '^';
    }
    // sales document changed on should be at or after the last record we have.
    result =
        result +
        'sales_document_header_data_changed_on>=' +
        grLastChanged.u_changed_on;
}
```

ERP scheduled extraction table fields (continued)

Field	Description
	Encoded query | encoded_query sales_document_header_data_sales_organization=1710 {SAPTODAY} {SAPYESTERDAY} {TODAY} {YESTERDAY} For more information, see Encoded query strings.

ERP Data Hub Monitor field descriptions

The ERP Data Hub Monitor enables you to track ERP transactions and their progress.

ERP Monitor Extractions and Remote lookup tab fields

Transaction log	A unique, system-assigned transaction log identification number. Select the number for details.
Flow engine context	If the transaction occurs in a flow, the specific flow is logged and displayed here. Select the context name to open the flow in Workflow Studio and obtain more information. **Note:** System-provided flows are logged. Custom flows aren't logged.
Status	Extraction or remote lookup state: Started, Success, or Error.
Source	Source table in ERP Data Hub. For example, sn_erp_integration_etl_extraction_task or sn_erp_integration_remote_table.
Source Name	System ID of the source identified in the **Source** field. Select the name to obtain more information.
System	ERP system on which the transaction took place.
ERP Model Operation	Operation called by the transaction. For example, if you're creating an ERP model with a read operation and call it, the model's read operation is displayed here.
Duration (ms)	Amount of time (in milliseconds) that the transaction took to process.
Created	The date and time the transaction completed.

ERP Monitor Extractions and Remote lookup tab fields (continued)

Async	If Async is true, the job runs in the background, for example, extractions are asynchronous. If Async is false, the user initiates the job, for example, remote lookup is synchronous.
Encoded query	Encoded query being used for the specific transaction. For more information, see Encoded query strings.
Logged on	UNIX timestamp for transaction start time.
Response item count	The number of records that were read.

ERP Data Hub Use ERP Data action details for flows

After you're done creating and managing your model in ERP Data Hub, go to Workflow Studio and use the Use ERP Data action to test your model's inputs and outputs. You can then build a flow with the action to incorporate ERP (Enterprise Resource Planning) data into the ServiceNow AI Platform.

For information on using the Use ERP Data action to retrieve ERP data, see Building flows to read or update the ERP system.

Roles and availability

Available as part of ERP Data Hub.

Role requirements

This action requires roles granted by delegated development or assigned to the user. For more information, see User access to Workflow Studio flows.

Inputs

Provide a value for each input that your action needs. To add dynamic values, you can also select data pills using the pill picker.

ModelName

Data type: *Dynamic Choice*

Name of the ERP model that contains the read or update operation. The model must already be created, and have the read or update operation defined in ERP Data Hub. For more information, see Managing how models read and update the ERP system.

ModelOperation

Data type: *Dynamic Choice*

Whether to **Read** or **Update** the ERP system. The read or update entity must already be defined in ERP Data Hub. For more information, see Add a read, update, or create entity to a model in ERP Data Hub.

Mandatory Field(s)

Data type: *Dynamic Template*

Fields that are required for the operation. Fields must already be defined as required input parameters when you manage the model in ERP Data Hub. For more information, see Manage input parameters for an ERP Data Hub model operation.

Optional Field(s)

Data type: *Dynamic Template*

Fields that are optional for the operation. Fields must already be defined as optional input parameters when you managed the model. For more information, see Manage input parameters for an ERP Data Hub model operation.

System

Data type: *Reference*

Name of the connected ERP system. For more information, see Working with ERP systems in ERP Data Hub.

Outputs

You can use these outputs as inputs for other items.

Action status

Data type: *Object*

Data pill that contains the current runtime details about the action. The **Action Status** object consists of a code and a message.

Don't Treat As Error

Data type: *True/False*

Option to report the Action Status as an error or a success when returned to a flow.

Error Message

Data type: *String*

String data pill containing the error message produced by the action, step, or system operation.

Response

Data type: *Dynamic Object*

Requested data that's returned from the ERP system. For more information, see Choose output parameters for an ERP model.

Flows that ship with ERP Data Hub

ERP Data Hub includes two ERP (Enterprise Resource Planning) flows that run automatically.

You don't need to do anything to activate the flows, but you can customize them in Workflow Studio if you want. For example, to change the time that they run daily. For more information, see Edit a flow.

Note:
ERP Data Hub appears as **ERP Integration** when you work with it in Workflow Studio.

The flows that ERP Data Hub automatically runs are:

- **Get SAP BAPIs and tables when system becomes active**: This flow is triggered automatically as soon as the ERP heartbeat for an ERP system is live. It detects and retrieves the available BAPIs (Business Application Programming Interface) and tables for use when managing models. This flow also reads the OData catalog service if the OData heartbeat is successful.
- **ETL SAP BAPIs and Tables From all systems**: This flow calls the **Get SAP BAPIs and tables when system becomes active** flow every night to retrieve any changes to available BAPIs and tables on the ERP system.

The flows save their changes to the following tables:

- sn_erp_integration_sap_tables_list
- sn_erp_integration_sap_bapi_list_list

ERP Data Hub and ERP-CM glossary

Learn about ERP (Enterprise Resource Planning) terminology and concepts that apply to ERP Data Hub and ERP Customization Mining (ERP-CM).

BAPI

A Business Application Programming Interface (BAPI) is a standard interface to the business object models in SAP products, similar to an API.

candidate

A candidate is a group of recommended remote tables that you can use to create an application based on a custom transaction (ERP root) in the system. Candidates can also be existing applications that sit on top of the legacy ERP system of record.

Good candidates for replatforming tend to be smaller applications that use data from the system of record.

candidate score

A metric for how well a custom application (ERP root) fits to a remote table in ERP Customization Mining.

connection and credential alias

The credential alias is the system connection access enabler that's maintained in the ServiceNow credential alias. The credential alias is used to access the ERP system of record.

custom fields

Custom fields are values added to a table by the customer.

entity

The table or Business Application Programming Interface (BAPI) function call that ERP Data Hub uses to read or update the system of record.

ERP application

The ERP (Enterprise Resource Planning) custom application that you choose to replatform using ERP Customization Mining is an ERP root in the ERP system.

ERP model

The ERP model represents a distinct set of features and functionalities tailored to address specific business processes or activities. An ERP model encompasses multiple tables from

the system of record, as well as APIs and ETL processes, to create a holistic data set. For example, you can have one ERP model for sales orders and another for inventory.

ERP module

The functional business area on the ERP system. The ERP root can only have one selected business area.

ERP system

An ERP system represents a connection to a section of your ERP system of record. The system plays a crucial role in data synchronization, sharing, and collaboration, enabling seamless integration and operation between the ERP model and the connected ERP system.

ERP table

Individual tables (both custom and standard) in the ERP system are part of the remote tables. For example, in sales, ERP tables could include VBAK, VBAP, VBFA, VBPA, VBEP, MKPF, or MSEG.

extraction table

Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.

mapped value

Value of the field on the ERP system that's being mapped as an operation parameter in ERP Data Hub.

odata

In ERP Data Hub, create an OData connection to link to SAP via HTTP so data can be extracted for use in remote tables and extraction tables.

operation

Individual maintenance task performed to read or update the system of record.

remote table

A remote table is an aggregation of the system of record tables that provide value when gathered in one table. The remote table is the foundation of the ERP model. A candidate can consist of several remote tables.

system of record

The ERP system of record is where the data lives and is distributed from, for example, SAP.

ERP Customization Mining (ERP-CM)

The ServiceNow® ERP Customization Mining (ERP-CM) product enables Solution Integration consultants to find application candidates with customized code in their ERP (Enterprise Resource Planning) system. ERP-CM ranks the candidate suitability for replatforming onto the ServiceNow AI Platform.

ERP Data Hub and ERP Customization Mining (ERP-CM) are App Engine for ERP products of ServiceNow. They help organizations modernize and orchestrate legacy code, complimenting processes and integrations that are underserved with the existing system of record, such as SAP. Together, ERP Data Hub and ERP-CM make it faster and easier for you to build applications on the ServiceNow AI Platform using data from a legacy ERP system.

Replatforming is the process of scanning legacy ERP system code to find potential candidates to move onto your ServiceNow AI Platform instance as new apps. You can use data from the ERP system as a source for apps built on the ServiceNow AI Platform, improving performance, enhancing security, and reducing maintenance.

The replatforming of legacy code enables innovation on top of the system of record without knowledge of the legacy system. Database administrators and developers are then relieved of time-consuming efforts to create database views or endpoints in the system of record, freeing them to work on other projects, such as migration.

Note:
For information about new and updated features in the Xanadu release, see ERP Customization Mining release notes.

Explore

Learn about ERP Customization Mining concepts and features.

Configure

Install and configure ERP-CM connections.

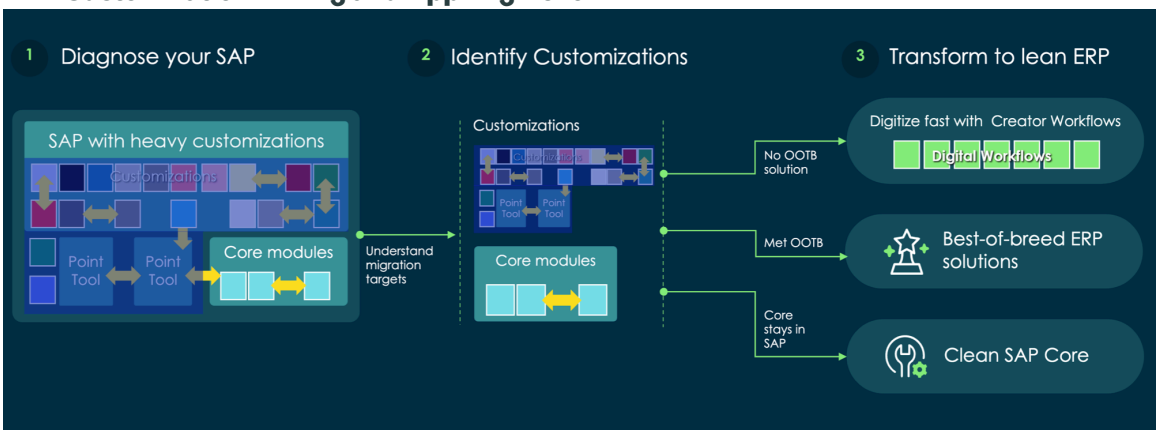
Work

Use ERP-CM to identify candidates to replatform.

Reference

Get details about ERP-CM components, such as tables and terminology.

ERP Customization Mining and App Engine for ERP



Learning resources for ERP Customization Mining

Learn more about ERP Customization Mining	ServiceNow resources
ERP Customization Mining has a number of training and learning resources for you to get started. (Some of these resources require logging in to the ServiceNow Learning site.)	Partner Essentials: Clean Core for App Engine ERP Customization Mining Overview ERP Clean Core with App Engine Overview Get to a clean ERP core with ServiceNow ERP Customization Mining

Request ERP Customization Mining on the store

Visit the ServiceNow Store website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Before you can use ERP-CM, you must first download ERP-CM from the ServiceNow Store. After you have completed the download, you may need to clear your local cache before ERP-CM appears on your instance.

After you download ERP-CM, install it on your instance. For more information, see Install ERP Customization Mining.

Exploring ERP Customization Mining

ERP Customization Mining (ERP-CM) enables you to identify candidates for replatforming. These candidates can be custom applications that were built on legacy ERP (Enterprise Resource Planning) systems that you want to quickly replatform.

Identifying candidates to replatform with ERP-CM

Legacy ERP systems can contain old tables with custom fields that have become difficult to maintain. Use ERP Data Hub and ERP-CM to identify candidates with custom fields to extract onto their instance as remote tables or extraction tables. Low-code developers can then use

the legacy ERP fields as data sources for apps created on the ServiceNow AI Platform, using a builder such as App Engine Studio, to replatform them.

You can use a combination of remote tables and extraction tables to retrieve data from your legacy ERP system.

- Remote tables get their records from running an associated script against an external data source.
- Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.

Replatforming is the process of scanning legacy ERP system code to find potential candidates to move onto your ServiceNow AI Platform instance as new apps. You can use data from the ERP system as a source for apps built on the ServiceNow AI Platform, improving performance, enhancing security, and reducing maintenance.

When you find ERP candidates to replatform, ERP-CM also provides suggestions for next actions and similar candidates. ERP-CM supports any ERP modules, or functional areas that admins configure in ERP Data Hub. Some example modules are Finance, Procurement, and Sales. For more information, see Building and managing ERP models to work with ERP data.

Replatformed data is immediately available, mirrored in easy-to-manage tables and apps. Users no longer need to request information from database administrators, which can take weeks. Replatformed apps use the ERP system of record as the live data source.

Workflow for replatforming customized ERP applications

Complete the following workflow to replatform legacy ERP data with custom code from the system of record on the ServiceNow AI Platform using ERP Customization Mining (ERP-CM).

1. Meet with the customer and agree to run an analysis with ERP Customization Mining on their ERP system of record.
2. Connect the customer ERP system of record to the ServiceNow instance using ERP Data Hub.

   Note:
   Most customers have their own instance.

   ERP-CM uses the system connections configured in ERP Data Hub. For more information, see Working with ERP systems in ERP Data Hub.

3. Use ERP Data Hub to build ERP models from fields on the available remote tables. For more information, see the following topics:
   - Building and managing ERP models to work with ERP data
   - Using ERP remote tables in ERP Data Hub
4. Run ERP Customization Mining to find candidates. Candidates are custom code in the system of record that you can replace with ServiceNow apps. For more information, see Browse an overview of candidates in ERP-CM.
5. Choose the candidate to replatform. For more information, see Save potential candidates to replatform.
6. Use the candidate details in ERP-CM as a central place to enter comments and save attachments relating to the candidate. For more information, see View and work with candidate details in ERP-CM.
7. In the candidate details, identify any similar candidates that you could combine into a single replatformed app. For more information, see How ERP Customization Mining determines candidate score and potential.

8. Return to ERP Data Hub to continue building data models with remote tables and extraction tables. Confirm that all the necessary data is available in the ServiceNow AI Platform. For more information, see Building and managing ERP models to work with ERP data.
9. In App Engine Studio (AES) or another ServiceNow application, build a scoped app by using the replatformed data as a source. For more information, see Build apps using App Engine Studio.
10. Measure and monitor the performance of the new app using applicable metrics and parameters with your preferred analytic tools.

Benefits of ERP Customization Mining

Benefits of ERP-CM

Benefit	Feature	Role
Quickly identify candidates with custom code to replatform without waiting for your ERP administrator	Save potential candidates to replatform	sn_erp_mining.erp_admin, sn_erp_mining.erp_user
View suggestions for next steps and actions to guide you in replatforming	Check candidate recommendations in ERP-CM	sn_erp_mining.erp_admin, sn_erp_mining.erp_user
Compile selected candidates in a list by saving them	Save potential candidates to replatform	sn_erp_mining.erp_admin, sn_erp_mining.erp_user

Additional resources for ERP-CM

Learn more about ERP-CM	ServiceNow resources
ERP-CM enables Solution Integration consultants to find application candidates with customized code in their ERP system.	App Engine for ERP Overview on ServiceNow University **Note:** You must log in to ServiceNow University to access this resource. ServiceNow Community site Video: Unlock the full potential of your ERP system

How ERP-CM extracts and processes data

ERP Customization Mining (ERP-CM) retrieves data from the ERP (Enterprise Resource Planning) system using extractors and processes it before the data is available on the ServiceNow AI Platform.

Use an SAP extractor to analyze the ERP data sources and validate the syntax. The following extractors are available to retrieve data from the ERP system:

- APPstats
- COMPHIER
- SQLM

The supported extractors process data using ServiceNow AI Platform tables.

SAP data extractors

Extractor	Full name	Description	ServiceNow AI Platform processing tables	Populated ServiceNow AI Platform tables
APPstats	Application Statistics	Collects and analyzes system-level statistics and performance metrics from the ERP application.	sn_erp_mining_erp_appstat_data_staging sn_erp_mining_erp_appstat_data_agg	sn_erp_mining_erp_application sn_erp_mining_erp_user
COMPHIER	Component Hierarchy	Analyzes the hierarchy and relationships among different components at SAP, including business processes and module configurations.	sn_erp_mining_erp_comp_hier_data_staging sn_erp_mining_erp_comp_hier_data_agg	sn_erp_mining_erp_application sn_erp_mining_erp_module
SQLM	SQL Monitor	Enables the monitoring of all SQL statements and operations that are executed by running ABAP applications. The data collected by SQLM provides aggregated performance indicators (such as the number of executions, execution time, or number of effected rows) for all executed	sn_erp_mining_erp_sqlm_data_staging sn_erp_mining_erp_sqlm_data_agg	sn_erp_mining_erp_application sn_erp_mining_erp_table sn_erp_mining_erp_application_ta sn_erp_mining_erp_application_us

SAP data extractors (continued)

Extractor	Full name	Description	ServiceNow AI Platform processing tables	Populated ServiceNow AI Platform tables
		database operations.		

How ERP Customization Mining determines candidate score and potential

ERP Customization Mining (ERP-CM) generates a score to rank the potential for replatforming legacy ERP (Enterprise Resource Planning) candidates onto the ServiceNow AI Platform.

Every candidate has an ERP module specified in the candidate details in ERP-CM. The ERP module is used to evaluate the potential score of the candidate for replatforming, as well as the remote tables and extraction tables the model contains.

- Remote tables get their records from running an associated script against an external data source.
- Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.

Note:

Admins must first configure the connection to the ERP system in ERP Data Hub. For more information, see Working with ERP systems in ERP Data Hub.

High and low scores for candidate potential

ERP-CM evaluates candidates based on how well their tables and fields are supported by the ServiceNow AI Platform.

- A high potential indicates that ERP-CM can immediately use remote tables and extraction tables that match the ERP model for the application candidate without making additional changes.
- A low potential indicates that the application candidate matches few of the remote tables and extraction tables in the ERP models in ERP Data Hub.

How scores are calculated

The candidate potential score is calculated using the following metrics:

- ERP models: The number of ERP models that the candidate uses remote tables and extraction tables from.
- Similar candidates: The number of candidates with a similarity score above the threshold, which accounts for both table-based similarity and model-based similarity.

  The threshold can be adjusted in the System Properties [sys_properties] table, and the default `OR` condition can be changed to `AND`.

- Supported table score: The ratio of the number of custom app tables that are supported by any ERP model relative to the number of custom app tables.

Note:
Tables from either the Technical or ServiceNow table clusters are ignored from these computations.

- Supported table usage: The ratio of tables supported by the relevant ERP models that are used by the custom app.
- Unsupported model penalty: A penalty for the number of unsupported operations on tables in ERP models. The number of unsupported operations is passed through a sigmoid function, so it ranges from 0.0 and 1.0.
- Unsupported table extensions: The number of custom app tables that are also suggested as model extensions.
- Model inaccuracy: The number of tables supported by relevant ERP models that aren't used by custom apps, and are passed through a sigmoid function.

How ERP-CM works with ERP Data Hub and remote tables

ERP Data Hub enables you to connect to your ERP (Enterprise Resource Planning) system of record, and to organize its data.

Using ERP Data Hub, you can access standard fields for remote tables and ERP extraction tables, while ERP Customization Mining (ERP-CM) enables you to find good candidates for replatforming from the system of record to the ServiceNow AI Platform.

ERP-CM suggests candidates and possible next steps, such as updating remote tables and extraction tables to access ERP data. Remote tables send data to the ServiceNow instance as an attachment, which is then analyzed using AI/ML to identify similar candidates to replatform.

After you've identified candidates to replatform and taken the recommended action in ERP-CM, you need to use only ERP Data Hub to access the remote tables and extraction tables. These tables are data sources for building apps, flows, and workspaces.

Using remote tables and extraction tables with ERP Data Hub and ERP-CM

You can use a combination of remote tables and extraction tables to retrieve data from your legacy ERP system.

- Remote tables get their records from running an associated script against an external data source.
- Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.

Remote tables describe the schema for the data that you want to retrieve from an external source, such as the system of record. Use remote tables to connect the ServiceNow AI Platform to third-party sources, or to another instance, so that you can retrieve external data and optionally cache it in the memory. You can view external data in lists or forms and process it with standard Glide scripts. You can also group, sort, aggregate, and filter the data just like you would for standard internal tables.

By using a remote table, you can retrieve the data from external sources or from another instance with REST or SOAP services. The external data lives in the memory in read-only mode, which makes the data temporary, or transient, within the ServiceNow AI Platform. You can then view and manipulate the external data without importing or storing it. For more information, see Remote tables.

Use an extraction table to work with large amounts of ERP data. ERP extraction tables regularly save data to a local transform table on the ServiceNow AI Platform, which you can then process and use as the data foundation of a replatformed app.

Recommendations and similar candidates in ERP-CM

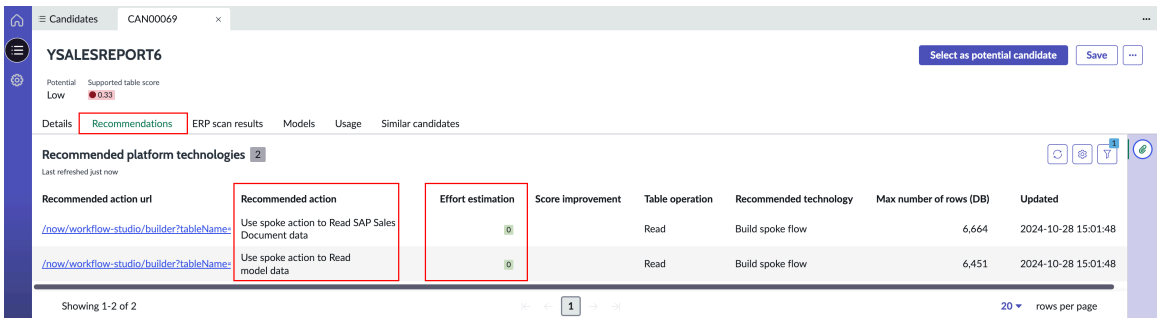
The record for each candidate in ERP Customization Mining (ERP-CM) displays information on suggested next steps and similar candidates to help you in replatforming an ERP (Enterprise Resource Planning) app.

Recommended actions for candidates

When you view a candidate, ERP-CM displays the number of recommended actions or next steps to take. The candidate details also have a **Recommendations** tab where you can view the suggested actions, and select them to open the associated app on the ServiceNow AI Platform.

For example, you may need to create a workflow in Workflow Studio for one candidate, update an integration spoke for another, and read an extraction table or remote table for a third candidate.

Each recommended action has an estimated effort, which is a numerical score that assesses how well the ServiceNow AI Platform has matching functionality. Green actions require little-to-no effort, while red actions are more difficult.



For more information on recommended actions, see Check candidate recommendations in ERP-CM.

Similar candidates in candidate details

For each candidate, ERP-CM displays a **Similar candidates** tab, which lists each similar candidate, a numerical score to represent how closely the two candidates are related, and a link to other similar candidates.

Similar candidates are helpful when planning how to best replatform a legacy app. When you replatform a custom app from the system of record, you don't have to replicate the old app exactly. Use the replatforming process to design a better app, perhaps one that addresses the needs of multiple similar candidates in a single, new app built using low-code tools on the ServiceNow AI Platform.

Next steps when replatforming apps to ServiceNow using ERP-CM

After you use ERP Customization Mining (ERP-CM) to identify legacy ERP (Enterprise Resource Planning) candidates, use additional ServiceNow AI Platform products and resources to replatform your app.

First, identify the customization you want to replatform from the ERP system with ERP-CM.

Then, use the remote tables and extraction tables linked in the candidate mining results for fast access to ERP data.

That ERP data can be used as a data source when you build new apps on the ServiceNow AI Platform, for example, with App Engine Studio.

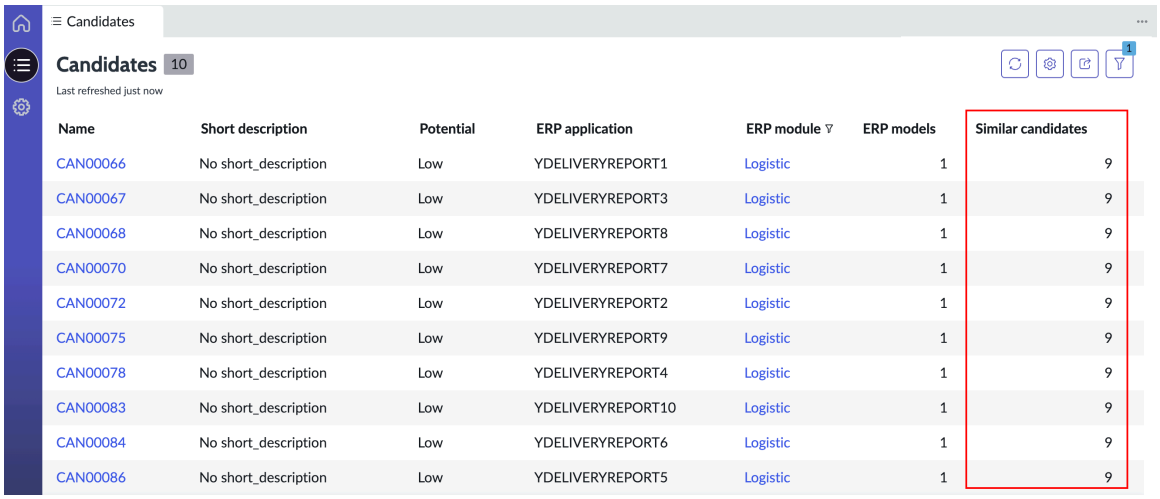
Build a ServiceNow app that consumes ERP data

The next step in replatforming with App Engine for ERP (Enterprise Resource Planning) is to build an app on the ServiceNow AI Platform that consumes the ERP data.

As you plan to replatform a legacy app on the ServiceNow AI Platform, consider where the data is coming from. For example, an old app may retrieve data from a third party into the system of record. When you build a new, replatformed app on the ServiceNow AI Platform, you can configure the new app to pull data directly from that third party instead of having the ERP Data Hub model pull it from the ERP system, which adds an extra step of retrieval.

Working with similar candidates when replatforming apps

If ERP-CM shows that a candidate has a number of similar candidates, consider building one app that meets the needs of some or all similar candidates when you replatform.



Name	Short description	Potential	ERP application	ERP module	ERP models	Similar candidates
CAN00066	No short_description	Low	YDELIVERYREPORT1	Logistic	1	9
CAN00067	No short_description	Low	YDELIVERYREPORT3	Logistic	1	9
CAN00068	No short_description	Low	YDELIVERYREPORT8	Logistic	1	9
CAN00070	No short_description	Low	YDELIVERYREPORT7	Logistic	1	9
CAN00072	No short_description	Low	YDELIVERYREPORT2	Logistic	1	9
CAN00075	No short_description	Low	YDELIVERYREPORT9	Logistic	1	9
CAN00078	No short_description	Low	YDELIVERYREPORT4	Logistic	1	9
CAN00083	No short_description	Low	YDELIVERYREPORT10	Logistic	1	9
CAN00084	No short_description	Low	YDELIVERYREPORT6	Logistic	1	9
CAN00086	No short_description	Low	YDELIVERYREPORT5	Logistic	1	9

When you replatform a custom app from the system of record, you don't have to replicate the old app exactly. Use the replatforming process to design a better app, perhaps one that addresses the needs of multiple similar candidates in a single, new app built using low-code tools on the ServiceNow AI Platform. App Engine Studio is the quickest app to use, but there are other builders available to you, depending on your licensing.

ServiceNow low- and pro-code builders

After you identify ERP data to replatform, citizen developers can use ServiceNow builders to create apps quickly from the data. Any custom fields that exist in the ERP system of record, such as SAP, can be leveraged by the apps you build using the ServiceNow AI Platform. For more on citizen development, see Delegated development and deployment.

Use any of the following ServiceNow builders to create apps using custom data:

- App Engine Studio
- Flows in Workflow Studio
- Playbooks in Workflow Studio
- Table Builder
- UI Builder
- Workspace Builder

For example, in App Engine Studio, use a template (or build the app from the ground up) and incorporate remote tables using ERP models and remote tables. You can combine legacy data from the ERP models and remote tables with other ServiceNow data in tables.

Using Glide to query ERP data

You can also access data from the system of record through the Glide API.

For an example Gilde query, see Sample Glide query for ERP data in ERP Customization Mining.

After you replatform custom code to a ServiceNow app

Replatformed apps on the ServiceNow AI Platform use live data from the system of record without writing any code back to it.

After you've identified candidates to replatform and taken the recommended action in ERP-CM, you need to use only ERP Data Hub to access the remote tables and extraction tables. These tables are data sources for building apps, flows, and workspaces.

If you're sure that the legacy code on the system of record isn't referenced anywhere else, you can remove it from the system after it's replatformed to a ServiceNow instance.

Configuring ERP Customization Mining

Set up ERP Customization Mining (ERP-CM) to enable the app to scan the system of record for candidates to replatform. Administrators can also quickly check the health of the ERP (Enterprise Resource Planning) connection and investigate any issues, and view knowledge articles related to errors.

You must install ERP Data Hub and configure connections and ERP data models before you can install ERP Customization Mining.

Server requirements

For the credentials, use an SAP service type user account in your system of record that requires the following S_RFC SAP authorization objects:

- S_RFC with Activity = 16 (Execute) for the following Function Modules:
  - SAPWL_WORKLOAD_GET_STATISTIC
  - SWNC_COLLECTOR_GET_AGGREGATES
  - SWNC_COLLECTOR_GET_DIRECTORY
  - SQLM_API_GET_NEXT_DATA_PACKAGE
- S_TABU_NAM with Activity 03 (Display) for the following table: TRNSPACET

There are additional authorizations needed for ERP Data Hub. For more information, see Configuring ERP Data Hub.

Note:
The credentials you specify for the ERP Data Hub connection must match the service user credentials in the system of record.

Types and number of available connections

ERP Data Hub and ERP-CM currently support ECC (minimum SAP Netweaver 7.31) and S/4HANA SAP systems.

Note:
Each installation of ERP-CM supports adding up to 10 systems. However, ERP-CM can connect to only one live ERP system at a time.

You should install ERP Data Hub and add credentials there before you install and configure ERP-CM. For more information, see Configure the ERP Data Hub credentials and connection.

The number of ERP connections you can have per ServiceNow instance depends on your license. If you have the ERP-CM license, you get one connection per instance.

For details on Connections and Credentials, see Connections and Credentials.

Plugins for ERP Customization Mining

You must have the following plugins installed:

Plugins for ECM

Plugin	Full name	Dependencies
ERP Data Hub	com.snc.sn_erp_integration	com.glide.script.vtable
ERP Customization Mining	com.snc.uib.sn_erp_mining	• com.snc.sn_erp_integration • com.glide.platform_ml

Request apps on the Store

Visit the ServiceNow Store website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Configure SAP for ERP-CM

Enable SQLM (SQL Monitor) on the productive system and confirm that ST03 (Workload Monitor) is collected for daily workloads before you can install ERP Customization Mining (ERP-CM).

Before you begin
Role required: none

About this task

Warning:
If you don't perform the SAP configurations from this procedure, and if there's no data in SAP, then ERP-CM won't work.

Procedure

1. Collect SQLM information from the productive environment by enabling SQLM in the SAP system.

   a. In the production system, start the transaction SQLM.
   For more information, refer to the SAP SQL Monitor documentation.

   b. Confirm that the **Activation state** field is set to **Globally active**.

   c. If SQLM isn't active, start the transaction SQLM again.

d. Navigate to **SQL Monitor** > **Setting** > **General Settings**.

e. Confirm that the **Delete Older Than [Days]** option is set to a minimum of `365` days, and select **Confirm**.

f. Activate the setting by selecting **All Servers**.

2. Verify that ST03 is collected for daily workloads from the productive environments.
   a. In the production system, start the transaction ST03.
   b. Verify that the **Workload data** field is set to **Daily**.

Install ERP Customization Mining

Install the ERP Customization Mining (ERP-CM) application (sn_erp_mining) if you have the admin role from the ServiceNow Store.

Before you begin

You must:

- Have a license and get entitlement to ERP-CM before you can install the application.

  For more information, see Licensing.
- Install ERP Data Hub. Use the application to configure connections to the system of record, as well as ERP (Enterprise Resource Planning) data models.
- Configure the JCO connector before you install ERP-CM. See the SAP documentation for more information.

The following plugins are required:

- ERP Data Hub
- Predictive Intelligence

Role required: admin

About this task

Procedure

1. In the ServiceNow Store, confirm that you have entitlements (or licenses) to the product and dependent applications.
   You can access the Store by navigating to **System Applications** > **All Available Applications** > **Available To Obtain From Store**.
2. Search for ERP Customization Mining and select **Install**.
3. **Optional:** Select **Load demo data** to create demo data in the app.
4. Select **Install** to confirm the installation of ERP-CM.

Result

The installation is complete. Select **Close** to return to the ServiceNow Store.

What to do next

After you install ERP-CM, ERP data from the connected system of record populates the ERP extraction tables in ERP Data Hub. For example, ERP application activity, Collector directory data, and Namespace data. You can then incorporate extracted data into ERP data models and remote

tables for use as a data source when building apps on ServiceNow. For more information, see the following topics:

- View ERP extraction tables
- Standard ERP models and extraction tables for ERP Data Hub

Run Guided Setup for ERP Customization Mining

Run the Guided Setup to configure ERP Customization Mining (ERP-CM).

Before you begin

You must first download and install ERP-CM from the ServiceNow Store.

Role required: sn_erp_integration.erp_admin, sn_erp_mining.erp_admin

About this task

For more information on using Guided Setup, see Guided setup.

Alternatively, you can configure ERP-CM without Guided Setup. For more information, see Configure a new ERP Customization Mining connection.

Procedure

1. Navigate to **All** > **ERP Customization Mining** > **ERP Customization Mining Guided Setup**.
2. Complete the Guided Setup for ERP Data Hub by selecting **Configure ERP Data Hub**.

   ERP Data Hub is required by ERP-CM to connect to systems.

   Follow the steps to configure ERP Data Hub, and select **Continue**. For more information, see Run Guided Setup for ERP Data Hub.
3. Connect ERP Customization Mining to the ERP system.
   a. Select **Select the ERP system to connect**, follow the steps, and select **Continue**.
   b. Select **Validate the connection status**, follow the steps, and select **Continue**.

Configure a new ERP Customization Mining connection

Specify the Connections and Credentials alias for ERP Customization Mining (ERP-CM) to connect to the ERP (Enterprise Resource Planning) system.

Before you begin

You should install ERP Data Hub and add credentials there before you install and configure ERP-CM. For more information, see Configure the ERP Data Hub credentials and connection.

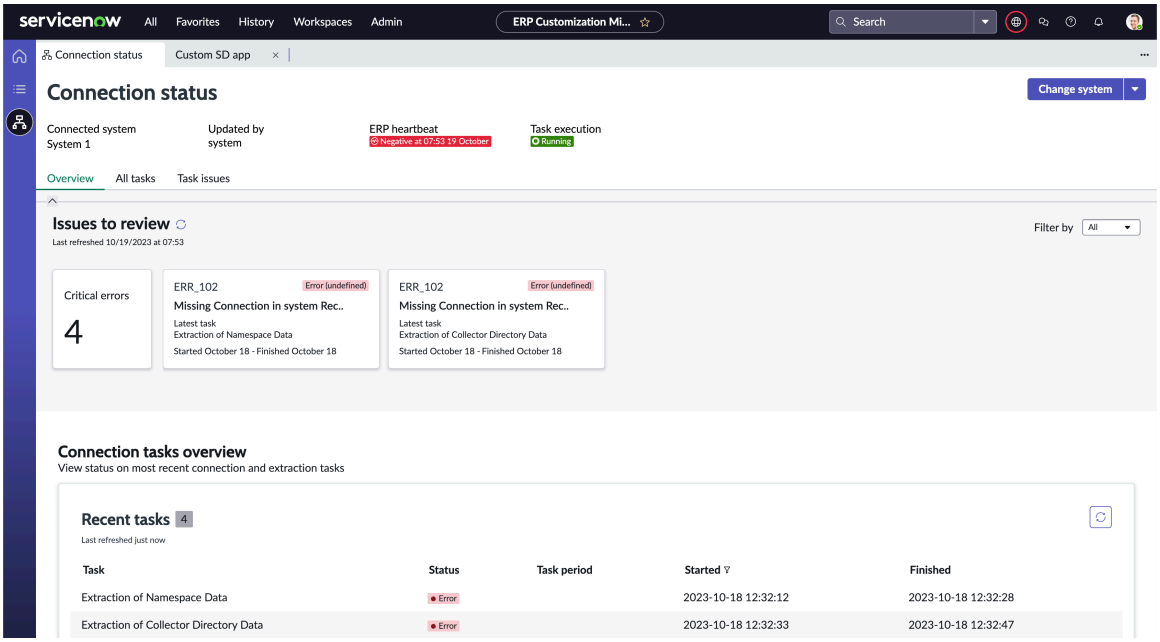
Role required: sn_erp_integration.erp_admin, sn_erp_mining.erp_admin

Procedure

1. Navigate to **All** > **ERP Foundation** > **ERP Customization Mining**.
2. Select the connection status icon () in the side panel.
3. Select the **Select connection** button.
   If you have already configured a system connection and want to use a different connection and credential, see Update an ERP-CM connection.
4. In the Connections and Credentials alias box, choose the credentials you configured in ERP Data Hub from the **Select connection & credential alias** field.
5. Select **Connect**.

ERP-CM begins extracting data from the connection.

6. Refresh the Issues to review list and the Executed tasks list by selecting their respective refresh icons ( ).



What to do next

After the system is connected, you can check the connection status and investigate errors at any time on the Connection status page. For more information, see Check and troubleshoot the data refresh status for ERP Customization Mining.

Update an ERP-CM connection

Update the Connections and Credentials alias for ERP Customization Mining (ERP-CM) to change the connection to the ERP (Enterprise Resource Planning) system. For example, you may want to change from a non-production system to a production system.

Before you begin

You should install ERP Data Hub and add credentials there before you install and configure ERP-CM. For more information, see Configure the ERP Data Hub credentials and connection.

Role required: sn_erp_integration.erp_admin, sn_erp_mining.erp_admin

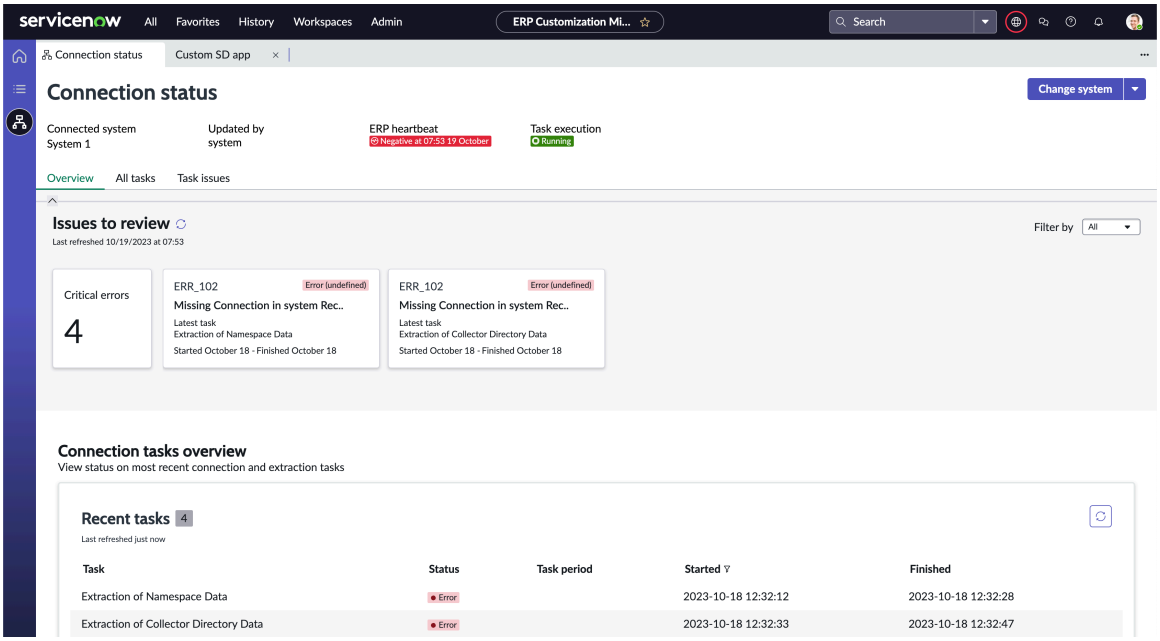
About this task

Warning:
Changing the system and credential alias deletes all existing data in the ERP-CM workspace and restarts the data integration process.

Procedure

1. Navigate to **All** > **ERP Foundation** > **ERP Customization Mining**.
2. Select the connection status icon ( ) in the side panel.
3. Select the **Change system** button.
4. In the **Change system & delete data** box, choose the system you configured in ERP Data Hub from the **Select system to change to** field.

5. Confirm your choice by selecting the **Yes, I am sure I want to change system and delete data** option.
6. Select the **Change and delete data** button.
   ERP-CM begins to delete current data and restarts the data integration process.
7. Refresh the Issues to review list and the Executed tasks list by selecting their respective refresh icons ( ).



What to do next

After the system is connected, you can check the connection status and investigate errors at any time on the Connection status page. For more information, see Check and troubleshoot the data refresh status for ERP Customization Mining.

Check and troubleshoot the data refresh status for ERP Customization Mining

Check the data refresh status to find out when ERP Customization Mining (ERP-CM) most recently loaded ERP (Enterprise Resource Planning) data from the system of record.

Before you begin

Role required: sn_erp_mining.erp_admin and sn_erp_mining.erp_user

About this task

To verify that ServiceNow AI Platform® is synchronized with the ERP system of record, ServiceNow AI Platform reloads data every 24 hours from the system of record.

You can receive email notifications for connection task success and failures. For more information, see Getting notifications for ERP Customization Mining connection updates.

Procedure

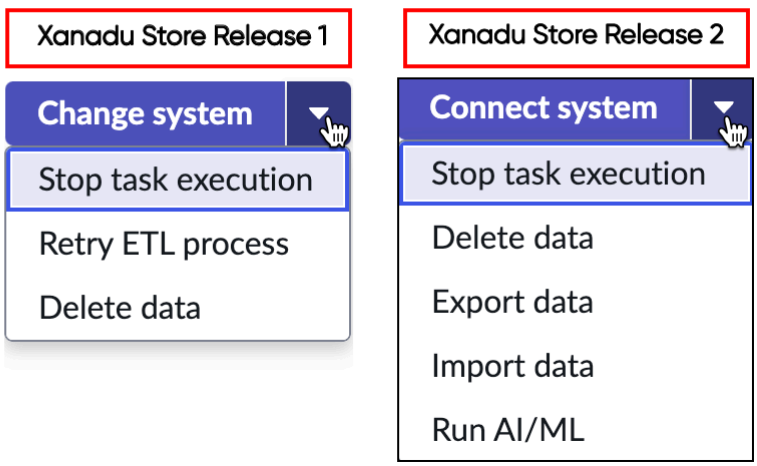
1. Navigate to **All** > **ERP Foundation** > **ERP Customization Mining**.
2. In the side panel, select the connection status icon ( ).
   The **Overview** tab displays a summary of connection issues to review, as well as a list of the most recent active connection tasks.
3. Check the most important connection details in the **Connection status** page header.

Connection status header details

Field	Definition
Credential alias	Alias of the connection credentials configured in ERP Data Hub.
Updated by	Name of the last account to update the connection credentials.
ERP heartbeat	Indicates whether a ping to the connection is currently successful.
Task execution	Indicates whether ServiceNow AI Platform® is currently attempting to connect to the instance.

4. **Optional:** View a refined subset of connection tasks by selecting one of the following tabs.

5. Troubleshoot any system connection issues using the **Change system** button (Xanadu Store Release 1) or **Connect system** button (Xanadu Store Release 2).



Trouble?

- Pause the task execution by selecting **Stop task execution**.
- Restart the ETL (extract, transform, and load) process by selecting **Retry ETL process**. (Xanadu Store Release 1)
- Delete all data in ERP-CM and start over by selecting **Delete data**. On the confirmation dialog, you must select the **Yes, I am sure I want to delete data** option and select the **Delete data** button.

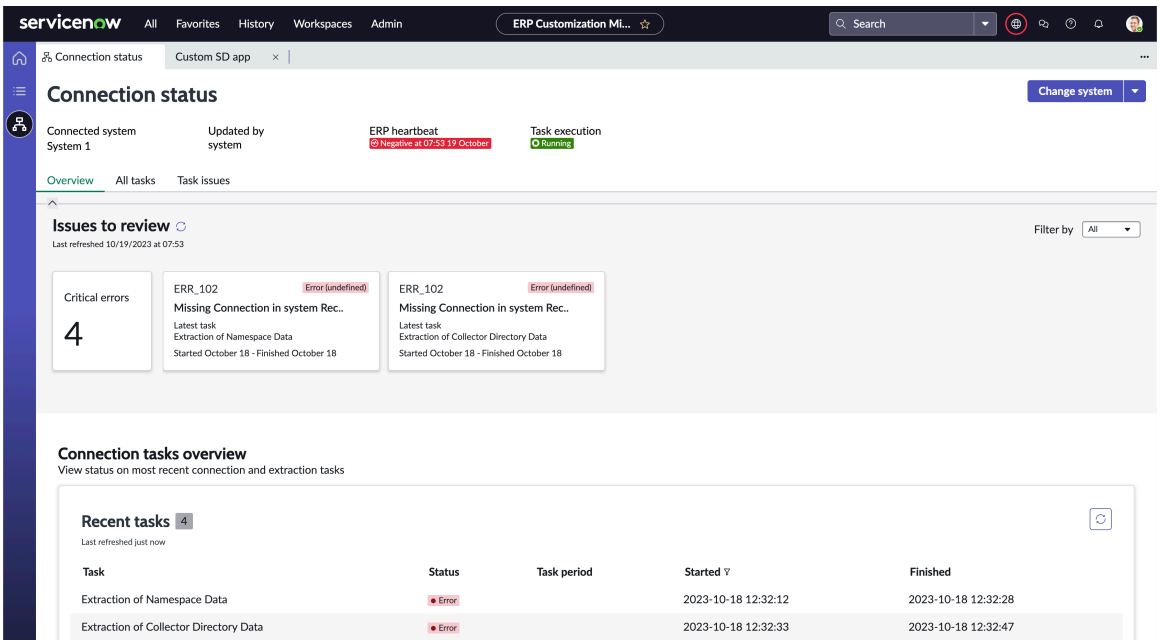
> ⚠ **Warning:**
> This action deletes all existing data in the ERP-CM workspace and restarts the data integration process.

- Share and back up data at any time by selecting **Export data**. A downloadable zip file is created.
- After exporting data, use the created zip file **Import data** for sharing and back up.
- Run AI/ML

6. View the information for each task and note any actions that you must take by scrolling to the task list.
   For a description of the field values, see ERP-CM task list field descriptions.

   For any column, you can select the more options icon ( ) to perform additional actions, such as **Show matching** and **Filter out**.

7. **Optional:** Update the status of any open task in error to indicate changes to its status.
   You can't resolve errors directly in ERP-CM, but you can mark them as resolved or irrelevant.
   a. Select the **Task** tab.
   b. On the **Task** tab, select the task that you want to update.
   c. Update the **Status** on the task record to **Resolved** or **Irrelevant**.
8. Refresh the Issues to review list and the Executed tasks list by selecting their respective refresh icons ( ).



Create a snapshot to share and save data in ERP Customization Mining

Export and import ERP Customization Mining base data to save and share.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **ERP Foundation** > **ERP Customization Mining**.
2. In the side panel, select the configuration icon ( ).
3. Select the **Connect system** button.

4. Select **Export data**.

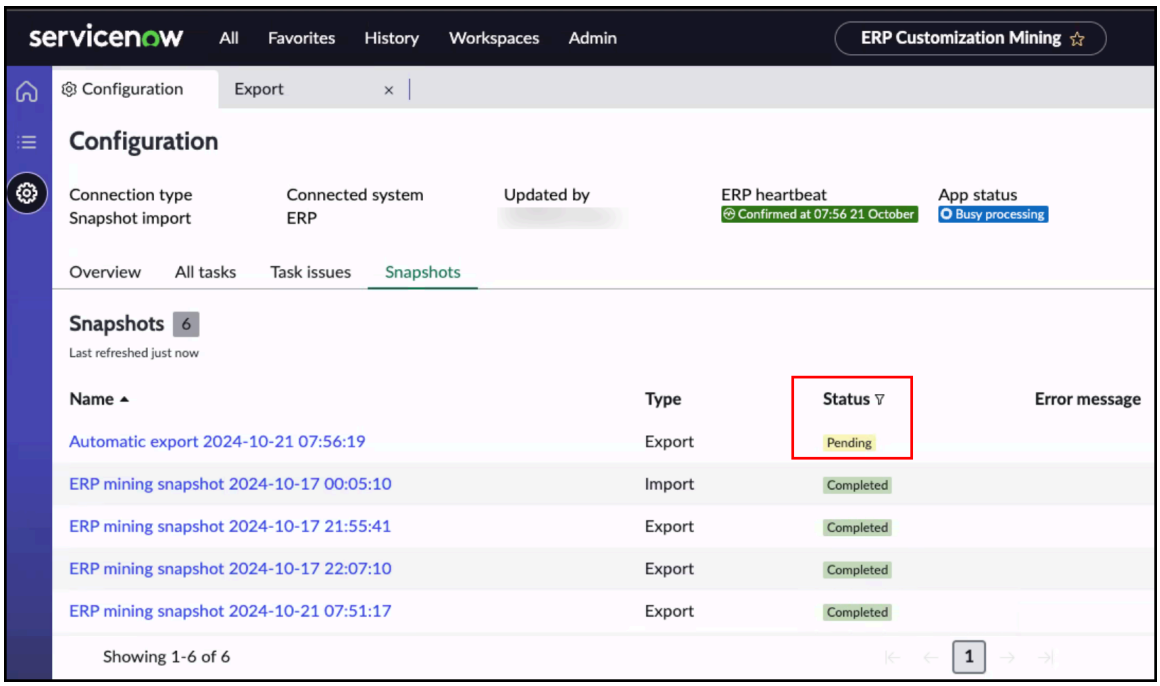
   A check is performed automatically prior to exporting. For details about the process, see ERP Customization Mining snapshot prerequisite check.

   If another export is in progress, an export unavailable message is displayed. Select **OK** and view the **Snapshots** tab to monitor the progress.

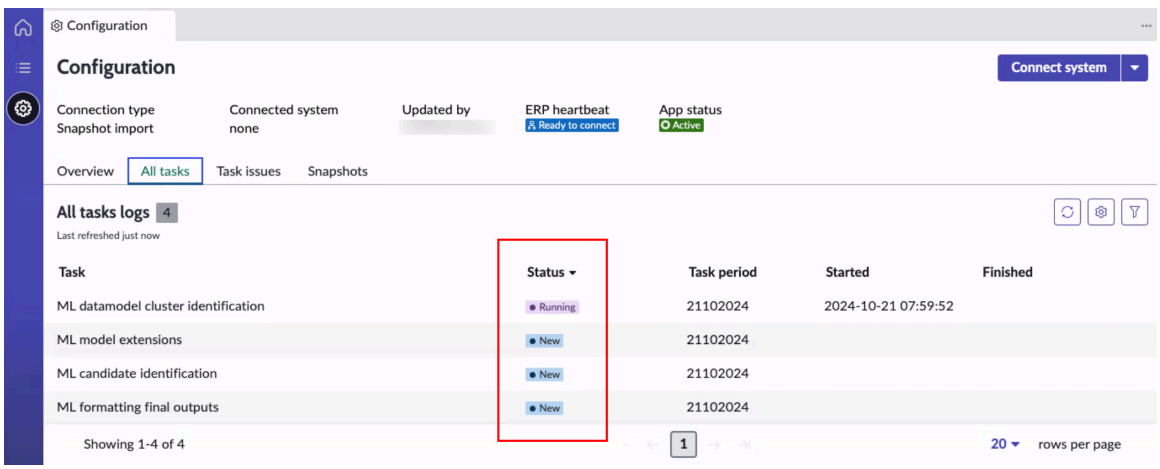
   a. On the pop-up window, review the message and select **Export data**.

      An automatic export is only performed before an import because an import overwrites the existing data. An export saves the data for a later import. An automatic export saves the data as a backup.

      In the **Snapshots** tab, the automatic export is **Pending**. Select the refresh icon to see the **Status** change to **In progress** and then **Complete**.



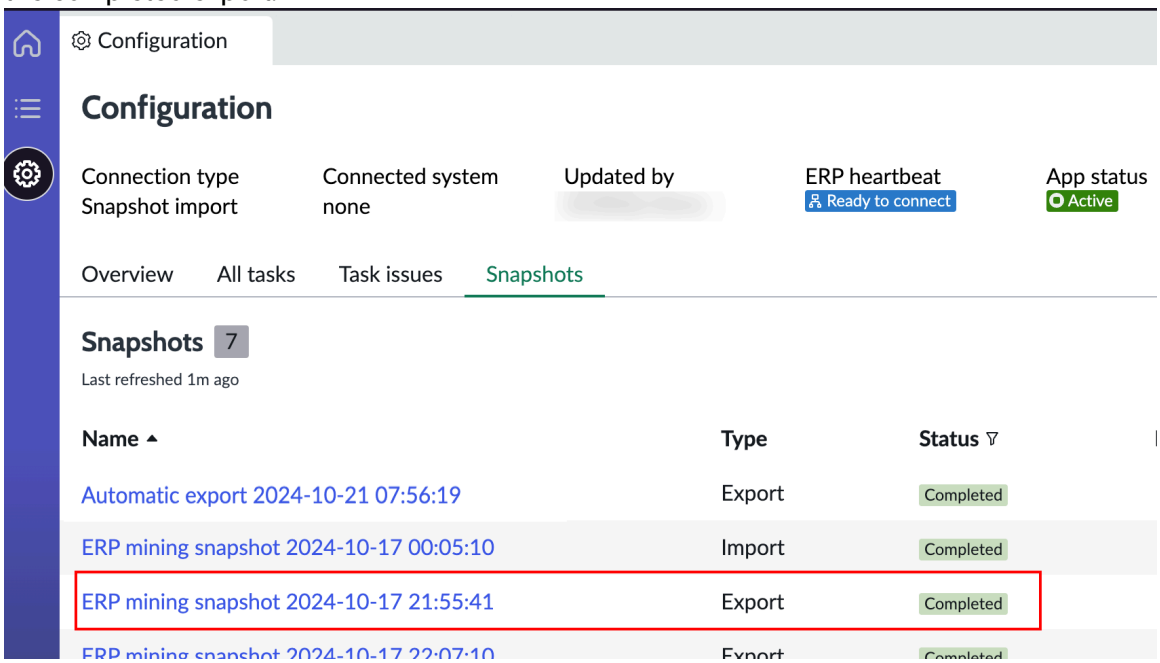
      You can also check the task status on the **All tasks** tab.



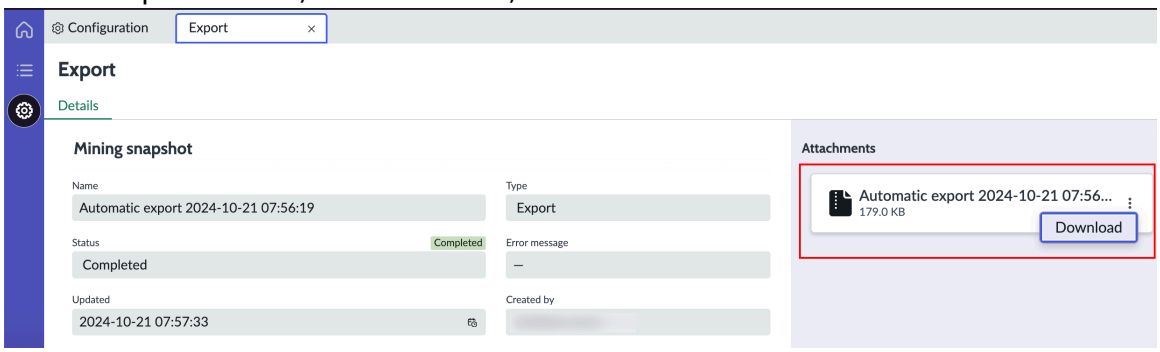
      For more execution details, open Workflow Studio, select **Operations**, and select **Flows**. The flow named **Export snapshot** calls the sub flow **Check set Snapshot to in progress**,

and then calls the sub flow **Create Snapshot**. For more information about viewing flows in Workflow Studio, see Workflow Studio.

b. When the export is finished, go to the **Snapshots** tab on the **Configuration** page and select the completed export.



c. On the snapshot record, in **Attachments**, select the actions icon and then select **Download**.



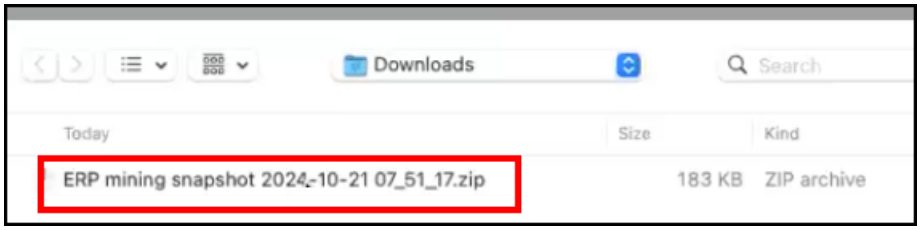
Next, import the data into another system.

5. Select the **Connect system** button, and then select **Import data**.

a. On the pop-up window, review the message and select the **I understand and want to import** option.

b. Select **+ Add file**.

c. Navigate to your downloads folder, select the ERP mining snapshot zip file, and select **Open**.



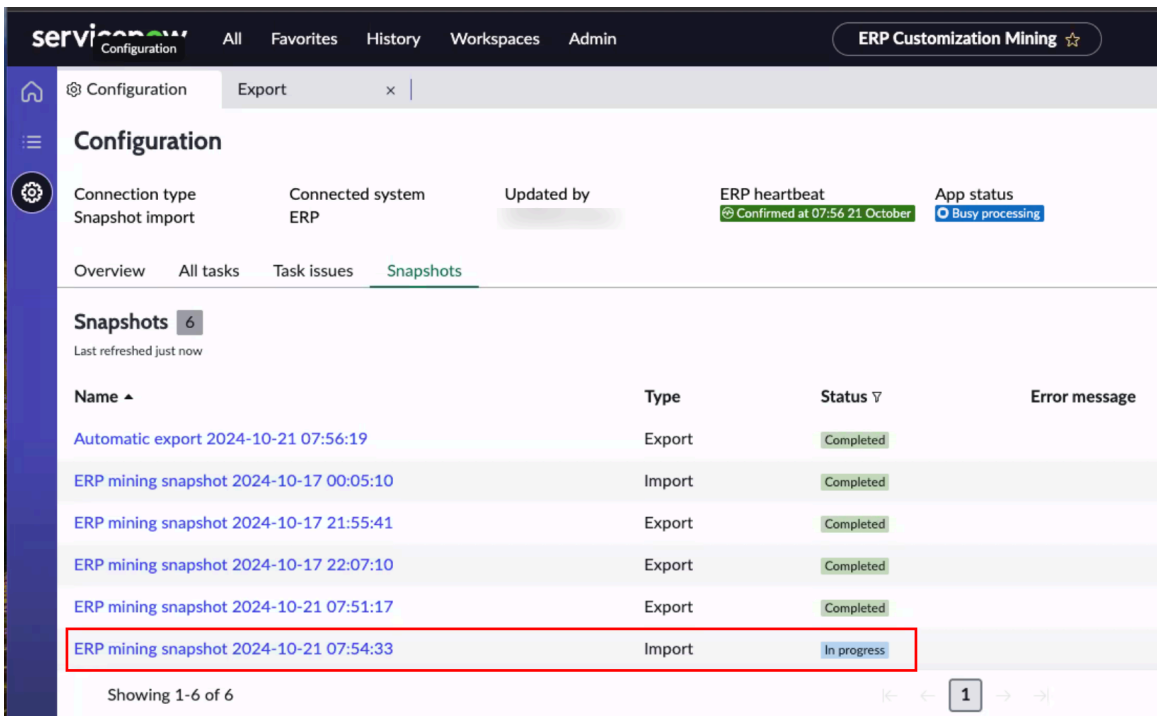
The zip process is done using the ServiceNow Integration Hub Professional Pack Installer plugin. If the plugin isn't installed, the process uses the existing zip functionality on the MID Server.

d. In the pop-up window, select **Upload**.

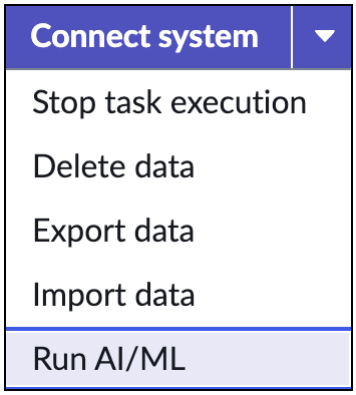
Validation is done automatically, for example, determining if there's an incomplete import job or an export job in progress.

e. When validation is complete, select **Import data**.

f. Check the status on the **Snapshots** tab.

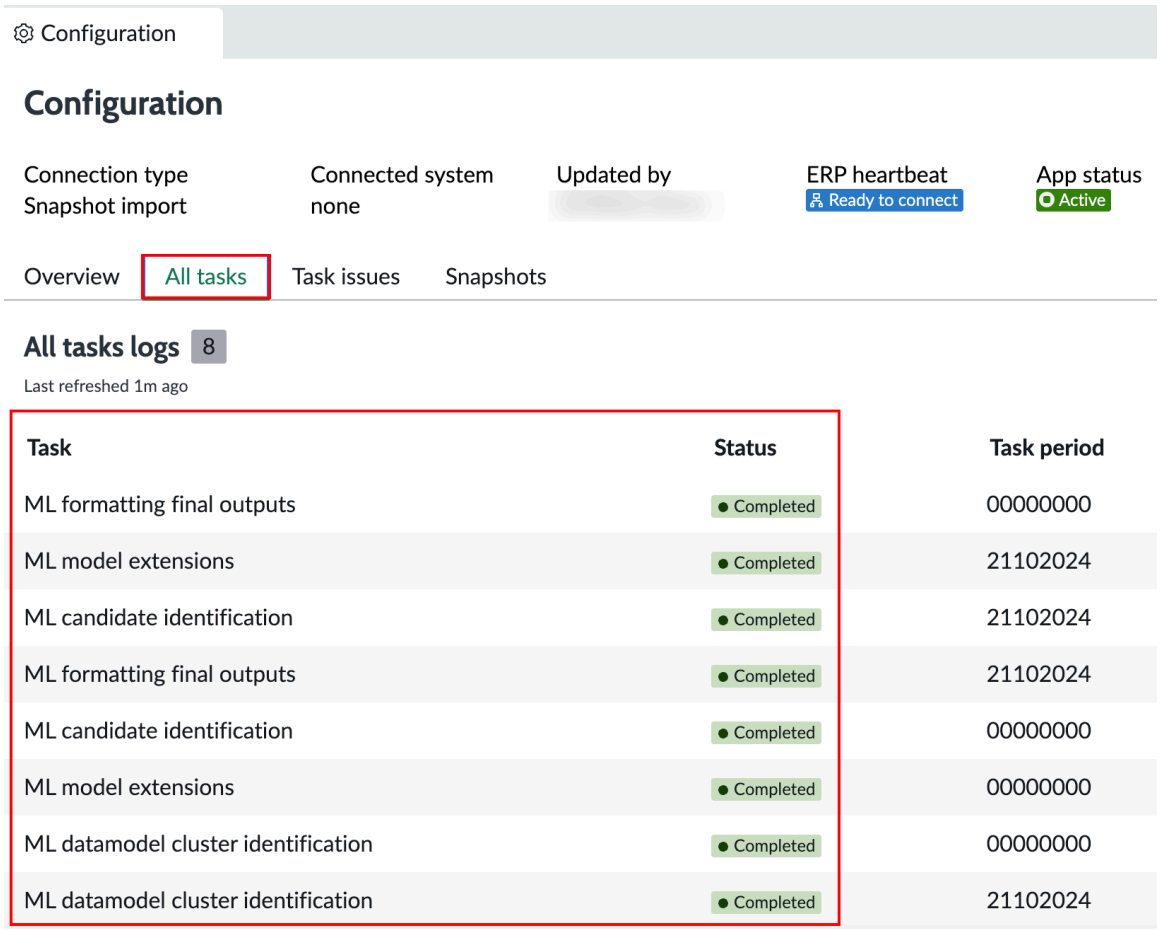


g. When the import is complete, manually run AI/ML to obtain candidates out of the new imported data by selecting the **Connect system** button and **Run AI/ML**.



AI/ML runs once according to day automatically, but you can initiate the process manually at any time using the **Run AI/ML** option. The manual and automatic process are the same and identify technical clusters, then candidate identification, and, lastly, model extensions.

View the **All tasks** tab to check the status.



Getting notifications for ERP Customization Mining connection updates

ERP Customization Mining (ERP-CM) can email you about the success and failures of ERP (Enterprise Resource Planning) system connections.

Notifications for success and failure

ERP-CM customization mining jobs run to find candidates for replatforming. With notifications, you can get an email when a customization mining task succeeds or fails.

- Success notifications indicate that all the customization mining tasks have finished running with no errors.
- Failure notifications indicate that one or more of the customization mining tasks are in error status.

The notification email you receive contains a link that takes you to the record for the customization mining job. You can view the progress of its tasks by selecting the **Show training progress** Related Link. The tasks there also appear in the Connection tasks overview list on the **Overview** tab of the Connection status page. You could then select to **Show matching** on a day's **Task period** value in the Connection tasks overview list to see the status of all tasks for that day.

Selecting your email notifications

Notification emails aren't enabled by default, and you must configure them for yourself in your ServiceNow AI Platform preferences. For details on configuring notifications, see Configure notifications for ERP-CM tasks.

Before you can set up notifications for yourself, your admin must add you to the ERP Customization Mining Notification group. For more information, see Add a user to a group.

Notifications also appear in the Notifications table

Notifications appear in the Notifications table, which you can access by navigating to **All** > **System Notification** > **Notifications**.

Flow that runs notifications

The ECM Statistical Data Extractor flow in Workflow Studio runs automatically to power notifications. You don't need to do anything to activate the flow, but you can customize it in Workflow Studio. For more information, see Edit a flow.

Configure notifications for ERP-CM tasks

Enable notifications to find out when ERP Customization Mining (ERP-CM) succeeds or fails in a mining job for the ERP (Enterprise Resource Planning) system of record.

Before you begin

Before you can set up notifications for yourself, your admin must add you to the ERP Customization Mining Notification group. For more information, see Add a user to a group.

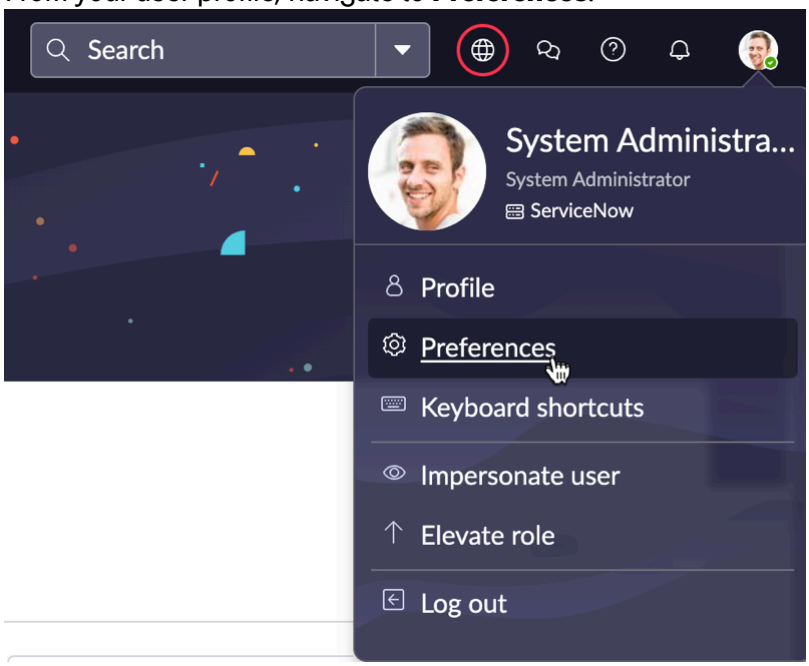
Role required: sn_erp_mining.erp_admin, sn_erp_mining.erp_user

About this task

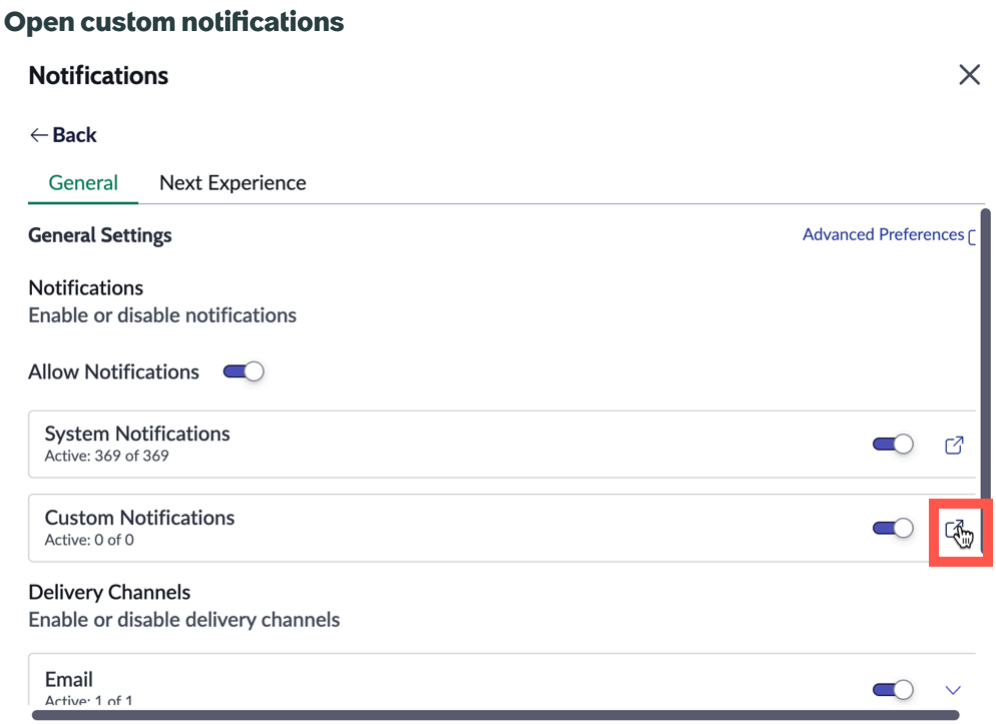
For general information on preferences, see System and custom notification and delivery channel preferences in Next Experience.

Procedure

1. From your user profile, navigate to **Preferences**.



2. Select the **Notifications** card.
3. Enable the toggle for **Custom Notifications** in **General Settings**.
4. Select the open in new tab icon for **Custom Notifications**.



5. Select the **Create notification** button.
6. Enter the basic details for the notification in the **Select notification** section of the **Create Notification** modal.

Basic notification fields

Field	Description
Personal notification name	Descriptive name for the notification you're setting up. For example, `Ann's ERP failure email notification`.
Notification	Select the notification that you want. The options are: ○ **ERP Extraction Completion Notification** ○ **ERP Extraction Failure Notification**

7. Select the **Next** button.
8. Select the delivery channel.
   a. Expand the method for how you want to be notified, such as **Email**.
   b. Enable the toggle for the email address that you want to receive the notification.
   Email addresses must already be associated with your ServiceNow AI Platform user account.
   c. Select the **Next** button.
9. Select the **Schedule** for your notification.
10. Finish setting up the notification by selecting the **Save** button.

Result

The notification email you receive contains a link that takes you to the record for the customization mining job. You can view the progress of its tasks by selecting the **Show training progress** Related Link. The tasks there also appear in the Connection tasks overview list on the **Overview** tab of the Connection status page. You could then select to **Show matching** on a day's **Task period** value in the Connection tasks overview list to see the status of all tasks for that day.

ERP Customization Mining roles

Administrators assign roles to give team members permission to configure or use ERP Customization Mining (ERP-CM).

Important:
When you assign ERP-CM roles to a user, you must include the scope. For example, assign the `sn_erp_mining.erp_admin` role, not just `erp_admin`.

For more on assigning roles, see Assign a role to a user.

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

ERP-CM roles

Role	Description	Additional access
sn_erp_mining.erp_admin	Grants the user access to updating application setup.	Contains sn_erp_mining.erp_user.
sn_erp_mining.erp_user	Grants the user read access on all tables needed for ERP Customization Mining, as well as some field-level write access in the sn_erp_mining_app_candidate table.	Contains the following roles: • sn_erp_integration.erp_data_pill • sn_erp_integration.erp_user

ERP-CM and security

In addition to role-based security and access control, ERP Customization Mining (ERP-CM) protects personally identifiable ERP (Enterprise Resource Planning) data in other ways.

Personally identifiable data is secured with ERP Data Hub and ERP-CM in several ways.

- You can customize ERP models and remote tables to exclude personal data in a specified field, such as email address.
- All remote tables are secured using access control rules (ACLs). If you have a remote table that contains sensitive data, use ACLs to restrict that table from ServiceNow users. For more information, see ServiceNow® access control.

Finding and working with candidates to replatform

Use ERP Customization Mining (ERP-CM) to identify potential ERP (Enterprise Resource Planning) app candidates for replatforming.

Replatforming is the process of scanning legacy ERP system code to find potential candidates to move onto your ServiceNow AI Platform instance as new apps. You can use data from the ERP system as a source for apps built on the ServiceNow AI Platform, improving performance, enhancing security, and reducing maintenance.

ERP Customization Mining evaluates application candidates and presents a numeric score. The higher the score, the better the candidate is for replatforming.

- A high potential indicates that ERP-CM can immediately use remote tables and extraction tables that match the ERP model for the application candidate without making additional changes.
- A low potential indicates that the application candidate matches few of the remote tables and extraction tables in the ERP models in ERP Data Hub.

Good candidates for replatforming tend to be smaller applications that use data from the system of record.

Note:

If you delete a candidate from ERP-CM, it automatically reappears the next time the ERP system is scanned. Instead of deleting candidates, use the **Save as potential candidate** feature to organize your candidates.

If ERP-CM shows that a candidate has a number of similar candidates, consider building one app that meets the needs of some or all similar candidates when you replatform.

Browse an overview of candidates in ERP-CM

View the ERP Customization Mining (ERP-CM) home page for a summary of ERP (Enterprise Resource Planning) app candidates to replatform onto the ServiceNow AI Platform.

Before you begin

Admins must first configure the connection to the ERP system in ERP Data Hub. For more information, see Working with ERP systems in ERP Data Hub.

Role required: sn_erp_mining.erp_user

Procedure

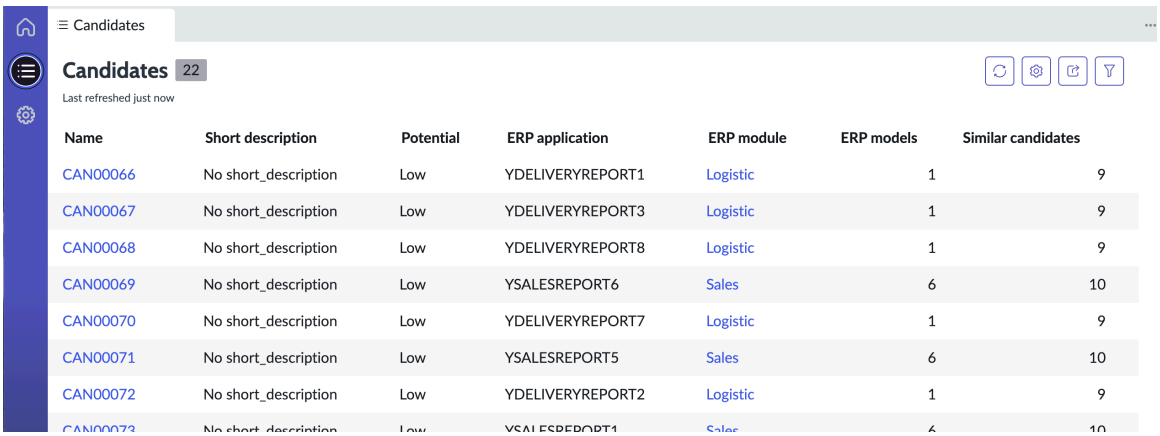
1. Navigate to **All** > **ERP Foundation** > **ERP Customization Mining**.
2. **Optional:** Filter potential candidates by selecting the card for the module in the **Candidates grouped per ERP module** section.
3. View the list of potential candidates.

   For a description of the field values, see ERP-CM candidate list field descriptions.

   Note:

   You can view the same list of candidates by selecting the candidates icon ( ) in the side panel.

   Candidate list in ERP-CM

Name	Short description	Potential	ERP application	ERP module	ERP models	Similar candidates
CAN00066	No short_description	Low	YDELIVERYREPORT1	Logistic	1	9
CAN00067	No short_description	Low	YDELIVERYREPORT3	Logistic	1	9
CAN00068	No short_description	Low	YDELIVERYREPORT8	Logistic	1	9
CAN00069	No short_description	Low	YSALESREPORT6	Sales	6	10
CAN00070	No short_description	Low	YDELIVERYREPORT7	Logistic	1	9
CAN00071	No short_description	Low	YSALESREPORT5	Sales	6	10
CAN00072	No short_description	Low	YDELIVERYREPORT2	Logistic	1	9
CAN00073	No short_description	Low	YSALESREPORT1	Sales	6	10

4. View and edit the details for a candidate by selecting the candidate.
   For more information, see Save potential candidates to replatform.

View and work with candidate details in ERP-CM

View and edit candidate details and recommended actions in ERP Customization Mining (ERP-CM). Analyze ERP (Enterprise Resource Planning) system scan results, linked ERP models, usage, and similar candidates.

Before you begin

Role required: sn_erp_mining.erp_user

Procedure

1. Navigate to **All** > **ERP Foundation** > **ERP Customization Mining**.
2. In the side panel, select the candidates icon ().
3. View and edit the basic details for a candidate in a new tab within ERP-CM by selecting the candidate **Name** in the candidates list.

   Alternatively, you can select a candidate directly on the ERP-CM home page. For more information, see Browse an overview of candidates in ERP-CM.

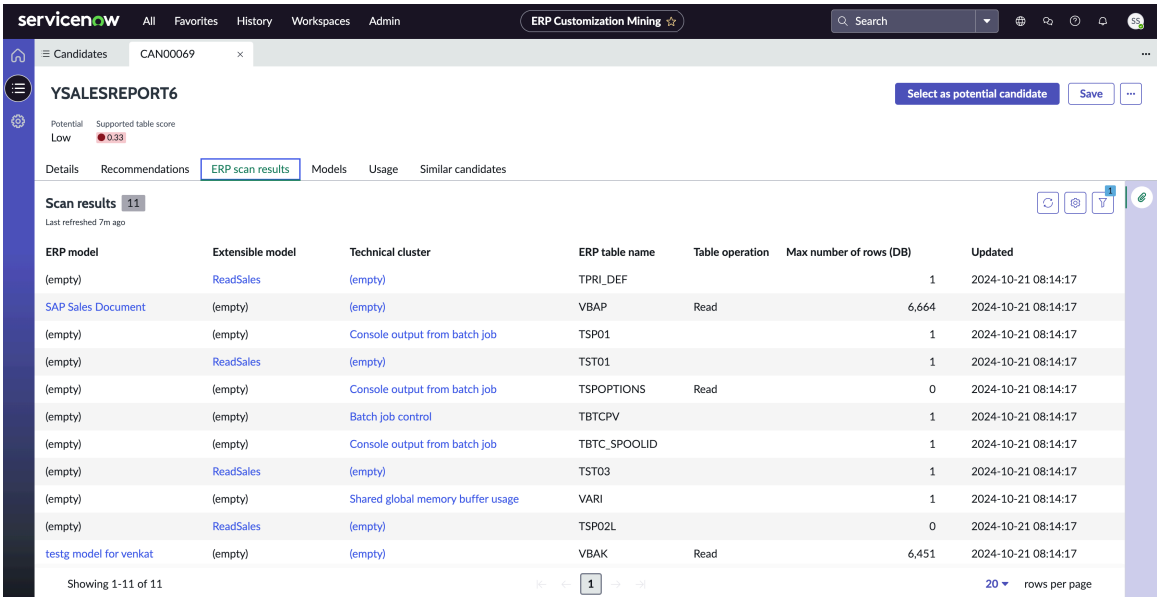
   The **Details** tab of the candidate is where you can review and update any basic details, as well as work with comments, attachments, and the Activity stream.

   #### Basic candidate details

Field	Description
Name	Record number of the potential candidate, which is automatically generated.
Status	Status of the candidate, either **Draft** or **Potential**. When you evaluate a draft candidate and decide the candidate is good to replatform, you can change the status from **Draft** to **Potential**.
ERP application	Name of the ERP application on the ERP system.
ERP module	Functional business area of the ERP system, such as **Sales**.
Short description	Brief description of the candidate, what the application does.
Long text	Additional information about the candidate.

4. Work with the next recommended steps by selecting the **Recommendations** tab.
   For more information, see Check candidate recommendations in ERP-CM.
5. View details on which tables were relevant to the candidate and the details of their scan on the ERP system by selecting the **ERP scan results** tab.
   For a description of the field values, see ERP-CM candidate scan results field descriptions.

Candidate scan results



6. Check and note which ERP models are ERP-CM identified as being related to the candidate by selecting the **Models** tab.

Models tab details

Field	Description
Model type	ERP model relevant to the candidate. This model can be a standard ERP model (which you work with in ERP Data Hub), a workflow, or a custom business area.
Relevant for implementation	Whether the ERP model is required for replatforming.
Module	ERP business area for the model on the system of record, such as **Sales**.
Table names	Names of the tables in the ERP model on the ERP system.
Updated	Date and time the list was most recently updated.

You can note which ERP models to work with in ERP Data Hub.

7. **Optional:** Check the number of system users that use the candidate application by selecting the **Usage** tab.
8. Research candidates related to the current candidate and their tables by selecting the **Similar candidates** tab.

   Similar candidates are helpful when planning how to best replatform a legacy app. When you replatform a custom app from the system of record, you don't have to replicate the old app exactly. Use the replatforming process to design a better app, perhaps one that addresses the needs of multiple similar candidates in a single, new app built using low-code tools on the ServiceNow AI Platform.

For a description of the field values, see ERP-CM similar candidates field descriptions.

9. Select the **Save** button and save any changes you made to the candidate details.

What to do next

After you work with the candidate details and identify similar candidates, you can:

- Check candidate recommendations in ERP-CM.
- Save potential candidates to replatform.
- Continue building remote table and extraction tables in the relevant ERP model in ERP Data Hub, making the data available on the ServiceNow AI Platform. For more information, see Using ERP models, extraction tables, and remote tables.

Note:

If you delete a candidate from ERP-CM, it automatically reappears the next time the ERP system is scanned. Instead of deleting candidates, use the **Save as potential candidate** feature to organize your candidates.

Check candidate recommendations in ERP-CM

Check the actions that ERP Customization Mining (ERP-CM) suggests to improve the ease of replatforming an ERP (Enterprise Resource Planning) candidate.

Before you begin

Role required: sn_erp_mining.erp_user

About this task

When you view a candidate, ERP-CM displays the number of recommended actions or next steps to take.

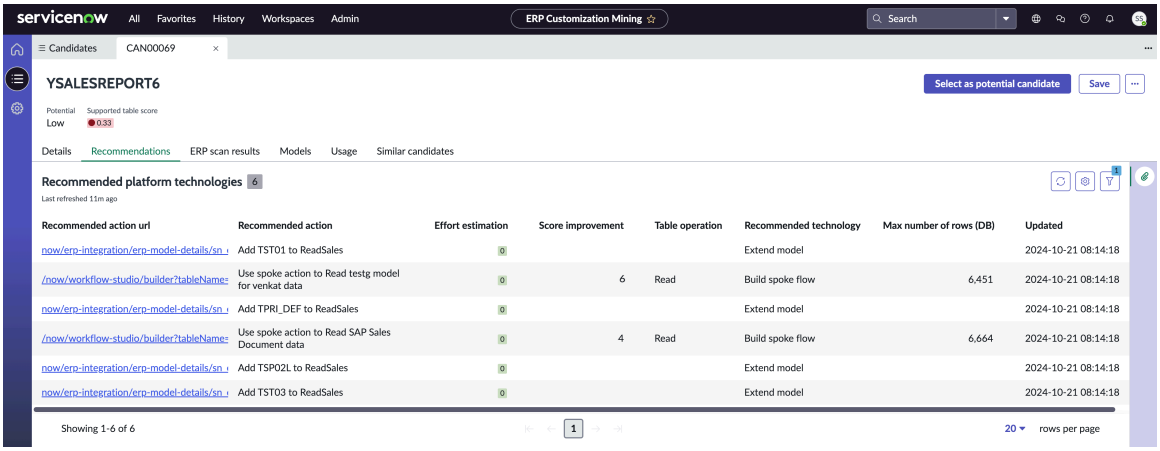
For example, you may need to create a workflow in Workflow Studio for one candidate, update an integration spoke for another, and read an extraction table or remote table for a third candidate.

Procedure

1. Navigate to **All** > **ERP Foundation** > **ERP Customization Mining**.
2. In the side panel, select the candidates icon ( ).
   Alternatively, you can select a candidate directly on the ERP-CM home page. For more information, see Browse an overview of candidates in ERP-CM.
3. Select the candidate that you want to view the recommended next actions for.
4. Select the **Recommendations** tab, which summarizes each suggested action.

   For a description of the field values, see ERP-CM candidate recommendations field descriptions.

Recommendations in ERP-CM



Note:
The score improvement feature is available starting with Xanadu Store Release 2.

5. Select a **Recommended action URL** to open the relevant destination on the ServiceNow AI Platform in a new browser tab.
   For example, if the recommendation is to read an extraction table, select the **Recommended action** link to open the ERP model in ERP Data Hub, where you can add the suggested table.

Save potential candidates to replatform

Use ERP Customization Mining (ERP-CM) to save ERP (Enterprise Resource Planning) app candidates to replatform.

Before you begin

Admins must first configure the connection to the ERP system in ERP Data Hub. For more information, see Working with ERP systems in ERP Data Hub.

Role required: sn_erp_mining.erp_user

About this task

Candidates are custom applications in your ERP system. ERP-CM scans your system of record to build a profile based on application logs and database activity logs. ERP-CM also scans for custom applications based on customized namespaces and other criteria. Replatformed apps use the ERP system of record as the live data source.

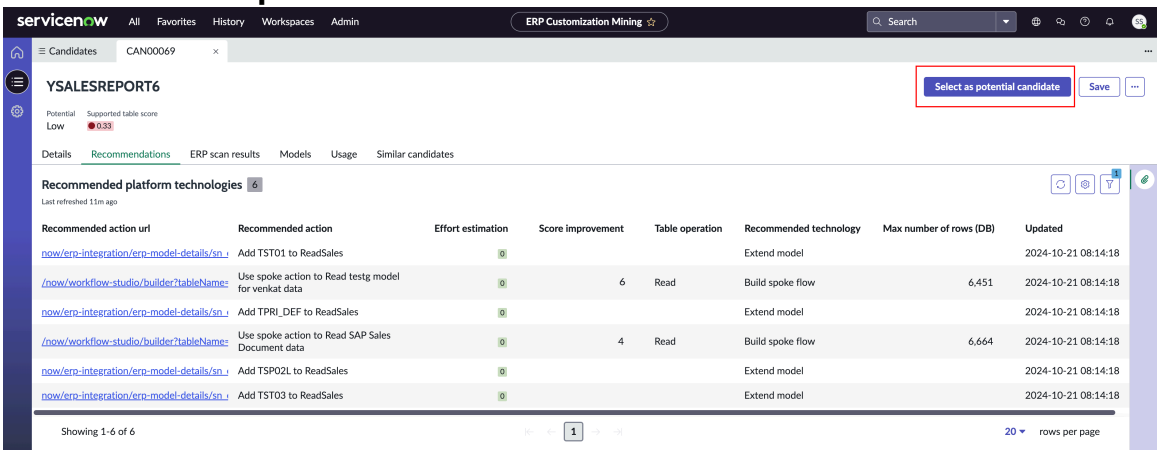
Note:

If you delete a candidate from ERP-CM, it automatically reappears the next time the ERP system is scanned. Instead of deleting candidates, use the **Save as potential candidate** feature to organize your candidates.

Procedure

1. Navigate to **All** > **ERP Foundation** > **ERP Customization Mining**.
   ERP-CM accesses the remote table that calls the system of record instance, and temporarily mirrors that data in the ServiceNow AI Platform.
2. Select the candidates icon ( ) in the side panel.
3. Select the candidate that you want to save as potential.
   Alternatively, you can select a candidate directly on the ERP-CM home page. For more information, see Browse an overview of candidates in ERP-CM.

4. Select the **Select as potential candidate** button.



If you change your mind and want to remove the candidate from the potential candidates list, select the **Change to draft candidate** button.

Note:
The score improvement feature is available starting with Xanadu Store Release 2.

Result

Selecting a candidate as a potential candidate changes the candidate status from Draft to Potential. The selected candidate is added to the Your selected potential candidates list on the home page.

What to do next

After you identify candidates, use ERP Data Hub to view custom fields in remote and extraction tables, and add them to your ERP model. For more information, see Building and managing ERP models to work with ERP data.

ERP-CM also recommends possible next steps for each candidate. For more information, see Check candidate recommendations in ERP-CM.

ERP Customization Mining reference

Find ERP (Enterprise Resource Planning) reference information for ERP Customization Mining (ERP-CM), including table details.

ERP Customization Mining and domain separation

Domain separation is unsupported for ERP Customization Mining (ERP-CM). Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Support level: No support

- The domain field may exist on data tables but there is no business logic to manage the data.
- This level is not considered domain-separated.

For more information on support levels, see Application support for domain separation.

Related topics

Domain separation for service providers

Sample Glide query for ERP data in ERP Customization Mining

Access data from the ERP (Enterprise Resource Planning) system of record through the Glide API.

The following is an example of a Glide query that fetches a customer name.

```
var sap_customer_gr = new
 GlideRecord('sn_erp_integration_st_sap_sales_customer');
sap_customer_gr.get('customer_number', '100032');
sap_customer_gr.getValue('name');
```

ERP-CM system connection tasks

ERP Customization Mining (ERP-CM) runs a number of tasks when connecting to the ERP (Enterprise Resource Planning) system.

- Aggregating daily extract of ERP activity
- Candidate identification
- Carveout Finalize import and prepare viewdata
- Extraction of Collector Directory Data
- Extraction of daily ERP Application activity
- Extraction of daily ERP Application SQL activity
- Extraction of daily ERP Module activity
- Extraction of Namespace Data
- Running ML similarity analysis

ERP Customization Mining snapshot prerequisite check

Before snapshot import and export in (ERP-CM), a check is performed automatically to confirm that other related processes aren't in progress.

Snapshot prerequisite checks

Prerequisite check	Description
System not connected	If you chose to import another snapshot, or attach a system, the 'table flush trigger' flow will immediately start to delete data. Once data is deleted, the selected snapshot will be imported, or the system will be attached.
System Connected and attach new system	The 'table flush trigger' flow will cancel Tasks that are currently executing. After tasks have been stopped' 'table flush trigger' flow will delete data, and attach the new system.
System Connected and import snapshot	The 'table flush trigger' flow will start by scheduling the creation of a snapshot. It will wait for the snapshot to be created. Snapshot is created when all tasks for all Collector entries are completed. Different flows execute subflow 'Check set Snapshot

Snapshot prerequisite checks (continued)

Prerequisite check	Description
	to in progress'. This subflow checks whether a record in the snapshot table should be completed with data. The 'table flush trigger' flow resumes and deletes all data. Subsequently, snapshot is imported.
System Connected and export snapshot triggered	Snapshot is created when all tasks for all Collector entries are completed. Different flows execute subflow 'Check set Snapshot to in progress'. This subflow checks whether a record in the snapshot table should be completed with data. If user wants to delete all data, or attach a new system, 'table flush trigger' flow will wait for the snapshot to be created.

ERP-CM standard tables and fields

Find details on standard ERP (Enterprise Resource Planning) remote tables, extraction tables, and fields in ERP Customization Mining (ERP-CM).

Standard remote tables for ERP-CM

ERP Customization Mining (ERP-CM) accesses several standard remote tables for ERP (Enterprise Resource Planning) data.

The following remote tables are available through ERP Data Hub and ERP-CM.

Remote tables for ERP Data Hub and ERP-CM

Label	Name	ERP module
SAP Company Code	sn_erp_integration_st_sap_company_code	Basis
SAP Country	sn_erp_integration_st_sap_country	Basis
SAP Currency	sn_erp_integration_st_sap_currency	Basis
SAP Language	sn_erp_integration_st_sap_language	Basis
SAP Material Stock	sn_erp_integration_st_sap_material_stock	Plant maintenance
SAP Purchase Document	sn_erp_integration_st_sap_purchase_document	Purchasing
SAP Purchasing Organization	sn_erp_integration_st_sap_purchasing_organization	Purchasing
SAP Customer Invoice	sn_erp_integration_st_sap_customer_invoice	Sales
SAP Distribution Channel	sn_erp_integration_st_sap_distribution_channel	Sales
SAP Division	sn_erp_integration_st_sap_division	Sales
SAP Sales Customer	sn_erp_integration_st_sap_sales_customer	Sales
SAP Sales Delivery	sn_erp_integration_st_sap_sales_delivery	Sales

Remote tables for ERP Data Hub and ERP-CM (continued)

Label	Name	ERP module
SAP Sales Document	sn_erp_integration_st_sap_sales_document	Sales
SAP Sales Organization	sn_erp_integration_st_sap_sales_organization	Sales
SAP Sales Revenue Recognition	sn_erp_integration_st_sap_sales_revenue_recognition	Sales
SAP Vendor	sn_erp_integration_st_sap_vendor	Sales
SAP Vendor Invoice	sn_erp_integration_st_sap_vendor_invoice	Sales
SAP Transport	sn_erp_integration_st_sap_transport	Transport

For more details on working with remote tables, see Remote tables.

You can use any of the standard remote tables as data sources when building apps in ServiceNow products, such as:

- App Engine Studio
- Flows in Workflow Studio
- Playbooks in Workflow Studio
- Table Builder
- UI Builder
- Workspace Builder

You can also access data from the system of record through the Glide API.

The following is an example of a Glide query that fetches a customer name.

```
var sap_customer_gr = new
 GlideRecord('sn_erp_integration_st_sap_sales_customer');
sap_customer_gr.get('customer_number', '100032');
sap_customer_gr.getValue('name');
```

Standard ERP-CM fields within remote tables

The standard ERP (Enterprise Resource Planning) remote tables available for use in ERP Customization Mining (ERP-CM) contain fields from additional SAP tables.

The standard remote tables contain the following additional fields. For details on the standard tables, see Standard remote tables for ERP-CM.

Standard ERP Data Hub and ERP-CM tables within remote tables

Remote table	Source table	ERP field name	Mapped field name
SAP Sales Document	VBAK	VBELN	document_number
SAP Sales Document	VBAK	ERDAT	date_of_document
SAP Sales Document	VBAK	ERZET	time_of_document
SAP Sales Document	VBAK	VKORG	sales_organization

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Sales Document	VBAK	VBTYP	document_category
SAP Sales Document	VBAK	AUART	document_type
SAP Sales Document	VBAK	AUGRU	order_reason
SAP Sales Document	VBAK	LIFSK	delivery_block
SAP Sales Document	VBAK	FAKSK	billing_block
SAP Sales Document	VBAK	KUNNR	customer_number
SAP Sales Document	VBAK	AUFNR	order_number
SAP Sales Document	VBAK	NETWR	document_value
SAP Sales Document	VBAK	WAERK	currency_code
SAP Sales Document	VBUK	LFGSK	delivery_status
SAP Sales Document	VBAP	MATNR	material_number
SAP Sales Document	VBAP	ARKTX	material_description
SAP Sales Document	VBAP	KWMENG	ordered_quantity
SAP Sales Document	VBAP	KLMENG	confirmed_quantity
SAP Sales Document	VBAP	NETWR	item_value
SAP Sales Document	VBAP	VRKME	sales_unit
SAP Sales Document	VBAP	ROUTE	delivery_route
SAP Sales Document	MARA	MTART	material_type
SAP Sales Document	VBUP	GBSTA	overall_item_status
SAP Sales Customer	KNA1	KUNNR	customer_number
SAP Sales Customer	KNA1	LAND1	country_code
SAP Sales Customer	KNA1	NAME1	name
SAP Sales Customer	KNA1	ORT01	city
SAP Sales Customer	KNA1	PSTLZ	postal_code
SAP Sales Customer	KNA1	STRAS	street
SAP Sales Customer	KNA1	STKZU	vat_liable
SAP Sales Customer	KNA1	STCEG	vat_reg_number
SAP Sales Customer	KNVV	VKORG	sales_organization
SAP Sales Customer	KNVV	VTWEG	distribution_channel
SAP Sales Customer	KNVV	SPART	division
SAP Sales Customer	KNVV	INCO1	inco_terms
SAP Customer Invoice	VBRK	VBELN	document_number
SAP Customer Invoice	VBRK	FKART	billing_type

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Customer Invoice	VBRK	WAERK	currency_code
SAP Customer Invoice	VBRK	ZTERM	payment_terms
SAP Customer Invoice	VBRK	NETWR	document_value
SAP Customer Invoice	VBRK	KUNRG	payer
SAP Customer Invoice	VBUK	GBSTK	overall_document_status
SAP Customer Invoice	VBRP	MATNR	material_number
SAP Customer Invoice	VBRP	ARKTX	material_description
SAP Customer Invoice	VBRP	NETWR	item_value
SAP Customer Invoice	VBRP	FKLMG	billing_quantity
SAP Customer Invoice	VBRP	VRKME	sales_unit
SAP Customer Invoice	VBRP	SHKZG	is_returns_item
SAP Customer Invoice	VBUP	GBSTA	overall_item_status
SAP Sales Organization	TVKO	VKORG	sales_organization
SAP Sales Organization	TVKO	EKORG	purchase_organization
SAP Sales Organization	TVKO	BUKRS	company_code
SAP Purchase Document	EKKO	EBELN	document_number
SAP Purchase Document	EKKO	BUKRS	company_code
SAP Purchase Document	EKKO	BSTYP	document_category
SAP Purchase Document	EKKO	LIFNR	vendor_number
SAP Purchase Document	EKKO	ZTERM	payment_terms
SAP Purchase Document	EKKO	EKORG	purchase_organization
SAP Purchase Document	EKKO	WAERS	currency_code
SAP Purchase Document	EKKO	IHREZ	ext_reference
SAP Purchase Document	EKKO	RESWK	supplying_plant
SAP Purchase Document	EKKO	BEDAT	date_of_document
SAP Purchase Document	EKPO	MATNR	material_number
SAP Purchase Document	EKPO	EMATN	supplier_material_number
SAP Purchase Document	EKPO	MENGE	ordered_quantity
SAP Purchase Document	EKPO	MEINS	order_unit
SAP Purchase Document	EKPO	NETPR	item_value
SAP Purchase Document	EKPO	MWSKZ	vat_applicable
SAP Purchase Document	EKPO	ELIKZ	fully_delivered
SAP Purchase Document	EKPO	REPOS	fully_invoiced

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Material Stock	MARA	MATNR	material_number
SAP Material Stock	MARA	MTART	material_type
SAP Material Stock	MARA	MATKL	material_class
SAP Material Stock	MARA	NTGEW	net_weight
SAP Material Stock	MARA	EANNR	ean_number
SAP Material Stock	MARA	EAN11	ean11_number
SAP Material Stock	MARA	MSTAE	material_status
SAP Material Stock	MAKT	MAKTX	material_description
SAP Material Stock	MARD	WERKS	plant
SAP Material Stock	MARD	LGORT	storage_location
SAP Material Stock	MARD	LABST	quantity
SAP Material Stock	MARD	DLINL	date_of_count
SAP Vendor Invoice	RSEG	BELNR	document_number
SAP Vendor Invoice	RSEG	GJAHR	document_year
SAP Vendor Invoice	RSEG	EBELN	purchase_document_number
SAP Vendor Invoice	RSEG	EBELP	purchase_document_item
SAP Vendor Invoice	RSEG	MATNR	material_number
SAP Vendor Invoice	RSEG	BUKRS	company_code
SAP Vendor Invoice	RSEG	WERKS	plant
SAP Vendor Invoice	RSEG	WRBTR	amount
SAP Vendor Invoice	RSEG	SHKZG	credit_debit_indicator
SAP Vendor Invoice	RSEG	MWSKZ	vat_applicable
SAP Vendor Invoice	RSEG	MENGE	quantity
SAP Purchasing Organization	T024E	EKORG	purchase_organization
SAP Purchasing Organization	T024E	EKOTX	name
SAP Purchasing Organization	T024E	BUKRS	company_code
SAP Sales Delivery	LIKP	VBELN	document_number
SAP Sales Delivery	LIKP	KUNNR	customer_number
SAP Sales Delivery	LIKP	ERDAT	date_of_document
SAP Sales Delivery	LIKP	ERZET	time_of_document
SAP Sales Delivery	LIKP	VKORG	sales_organization
SAP Sales Delivery	LIKP	LFART	delivery_type
SAP Sales Delivery	LIKP	ROUTE	delivery_route

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Sales Delivery	VBUK	GBSTK	overall_document_status
SAP Sales Delivery	LIPS	MATNR	material_number
SAP Sales Delivery	LIPS	CHARG	batch_number
SAP Sales Delivery	LIPS	WERKS	plant
SAP Sales Delivery	LIPS	LFIMG	quantity
SAP Sales Delivery	LIPS	NTGEW	net_weight
SAP Sales Delivery	LIPS	GEWEI	delivery_unit
SAP Sales Delivery	LIPS	VOLUM	volume
SAP Sales Delivery	LIPS	VOLEH	volume_unit
SAP Sales Delivery	LIPS	ARKTX	material_description
SAP Vendor	LFA1	LIFNR	vendor_number
SAP Vendor	LFA1	NAME1	name
SAP Vendor	LFA1	ORT01	city
SAP Vendor	LFA1	PSTLZ	postal_code
SAP Vendor	LFA1	STRAS	street
SAP Vendor	LFA1	STCEG	vat_reg_number
SAP Vendor	LFA1	WERKS	plant
SAP Vendor	LFM1	EKORG	purchase_organization
SAP Vendor	LFM1	WEBRE	gr_invoice_indicator
SAP Sales Revenue Recognition	VBREVK	VBELN	document_number
SAP Sales Revenue Recognition	VBREVK	POSNR	document_item
SAP Sales Revenue Recognition	VBREVK	SAKRR	accr_val_clearing_account_number
SAP Sales Revenue Recognition	VBREVK	SAKRRK	offset_clearing_account_number
SAP Sales Revenue Recognition	VBREVK	ACC_VALUE	total_accrued_value
SAP Sales Revenue Recognition	VBREVK	WRBTR	amount
SAP Sales Revenue Recognition	VBREVK	RVAMT	revenue_amount
SAP Sales Revenue Recognition	VBREVK	WAERK	currency_code
SAP Country	T005	LAND1	country_code

Standard ERP Data Hub and ERP-CM tables within remote tables (continued)

Remote table	Source table	ERP field name	Mapped field name
SAP Country	T002	SPRAS	language_code
SAP Country	T002	LAISO	language_iso_code
SAP Country	T005T	LANDX	description
SAP Language	T002	SPRAS	language_code
SAP Language	T002	LAISO	language_iso_code
SAP Currency	TCURC	WAERS	currency_code
SAP Currency	TCURC	ISOCD	currency_iso_code
SAP Currency	TCURT	LTEXT	description
SAP Distribution Channel	TVTW	VTWEG	distribution_channel
SAP Distribution Channel	TVTWT	VTEXT	description
SAP Division	TSPA	SPART	division
SAP Division	TSPAT	VTEXT	description
SAP Company Code	T001	BUKRS	company_code
SAP Company Code	T001	BUTXT	description
SAP Company Code	T001	LAND1	country_code
SAP Company Code	T001	WAERS	currency_code
SAP Company Code	T001	SPRAS	language_code
SAP Transport	E071	AS4DATE	date_of_transport
SAP Transport	E071	AS4TIME	time_of_transport
SAP Transport	E071	AS4USER	user
SAP Transport	E071	TRFUNCTION	type
SAP Transport	E071	TRKORR	number
SAP Transport	E071	TRSTATUS	status
SAP Transport	E071	OBJECT	object_type
SAP Transport	E071	OBJ_NAME	object_name
SAP Transport	E071	PGMID	program_id

ERP-CM field descriptions

Some tables of field descriptions in ERP Customization Mining (ERP-CM) are too large to maintain in task topics. Find information on those large tables in this section.

ERP-CM task list field descriptions

The task list in ERP Customization Mining (ERP-CM) displays information on connection tasks for the ERP (Enterprise Resource Planning) system.

Task list details

Column	Definition
Task	Name of the task in the ServiceNow AI Platform, such as a background script for extraction, an ETL task, or an analysis training script.
Short description	Brief description of the connection task error. **Note:** This column appears on the **Task issues** tab only.
Code no.	Error code number for errors or warnings. **Note:** This column appears on the **Task issues** tab only.
Category	The type of issue, either **Error** or **Warning**.
Status	Status of the task, such as **New** or **Insufficient data**.
Current step	Step of the connection task that ServiceNow AI Platform is being performed.
Total steps	Number of total steps required to complete the connection task.
Task period	Date when the connection task was most recently updated.
Started	Date and time when the connection task started running.
Finished	Date and time when the connection task finished running.

ERP-CM candidate list field descriptions

The candidate list in ERP Customization Mining (ERP-CM) displays information on the basic details for each candidate.

Candidate list details

Column	Description
Name	Name of the application candidate, which is the record number on the ServiceNow AI Platform.
Short description	Brief information on what tables the candidates accesses.
Potential	Evaluation of how well a candidate is suited for replatforming.

Candidate list details (continued)

Column	Description
	The value represents how well the remote tables and extraction tables for the candidate match an application. • A high potential indicates that ERP-CM can immediately use remote tables and extraction tables that match the ERP model for the application candidate without making additional changes. • A low potential indicates that the application candidate matches few of the remote tables and extraction tables in the ERP models in ERP Data Hub. **Note:** This column is available only on the candidates list on the Candidates page, not on the home page.
ERP application	Record for the ERP application of the candidate.
ERP module	Name of the ERP functional area in the system of record. For example, logistics, procurement, or sales.
ERP models	Number of ERP models the candidates belongs to. ERP models are configured in ERP Data Hub. An ERP model functions as a staging area that contains all potential fields you can add to remote and extraction tables, and read and update operations. You can then use the tables and queried data as a data source on the ServiceNow AI Platform. For more information, see Building and managing ERP models to work with ERP data. **Note:** This column is available only on the candidates list on the Candidates page, not on the home page.
Similar candidates	Number of similar candidates. Similarity is based on the remote tables.

ERP-CM candidate scan results field descriptions

The candidate list in ERP Customization Mining (ERP-CM) displays information on the latest scan of the ERP (Enterprise Resource Planning) system.

ERP scan results tab details

Field	Description
ERP model	Relevant ERP model that the candidate uses, which is configured in ERP Data Hub. Select an ERP model to open the model in ERP Data Hub.
Closest ERP model/Extensible model	The suggested closest matching ERP model that you can add the table to in ERP Data Hub. This column is helpful for custom tables, which aren't part of a standard ERP model and therefore won't have a table listed in the **ERP model** column. Select an ERP model to open the model in ERP Data Hub.
Technical cluster	If the field is empty, the table is an ordinary, non-technical table. If the field contains a value, the table belongs to a technical group and is disregarded in the candidate identification analysis.
ERP table name	Name of the table on the ERP system.
Table operation	Action that ERP-CM took on the scanned table: Create, Read, Update, or Delete.
Max number of rows (DB)	Maximum number of rows on the scanned table, which is useful for knowing candidate limitations.
Updated	Date and time the table was most recently scanned.

ERP-CM candidate recommendations field descriptions

The candidate Recommendations list in ERP Customization Mining (ERP-CM) displays information on suggested next actions for replatforming ERP (Enterprise Resource Planning) candidates.

Recommendations list details

Field	Description
Recommended action URL	Link to the recommended action. Select the recommendation link to open the relevant destination on the ServiceNow AI Platform. For example, the **Build a workflow** recommended URL would link to the flow in Workflow Studio.
Recommended action	Brief description of the action that ERP-CM suggests you take on a candidate when replatforming.

Recommendations list details (continued)

Field	Description
Effort estimation	Numerical score for how much effort the recommended action should require. Each recommended action has an estimated effort, which is a numerical score that assesses how well the ServiceNow AI Platform has matching functionality. Green actions require little-to-no effort, while red actions are more difficult.
Score improvement (available in Xanadu Store Release 2)	The possible increase in the overall potential for a candidate if you implement the recommendation. Each time the AI/ML workflow is triggered, either automatically or manually, score improvements are calculated.
Table operation	The action you should take on the table, such as **Read** or **Update**.
Recommended technology	App or product on the ServiceNow AI Platform where you take the recommended action.
Max number of rows (DB)	The maximum number of database rows on the system of record that the recommended action allows.
Updated	Date and time the recommended action was most recently updated.

ERP-CM similar candidates field descriptions

The Similar candidates tab in ERP Customization Mining (ERP-CM) displays information on similar candidates for replatforming ERP (Enterprise Resource Planning) apps.

Similar candidates tab details

Field	Description
Candidate	Name of the application candidate, which is the record number on the ServiceNow AI Platform.
Similar candidate	Record number of the similar candidate, which is automatically generated by the ServiceNow AI Platform.
Short description	Brief description of the similar candidate. For example, the tables the similar candidate reads.
ERP application	Name of the ERP application on the ERP system.

Similar candidates tab details (continued)

Field	Description
ERP module	Functional business area of the ERP system, such as Sales.
Similarity	Numeric score representing how similar the candidates are, from 0 through 1. A higher score, for example 0.94, indicates very similar candidates. For more information, see Recommendations and similar candidates in ERP-CM.
Similar candidates	Number of similar candidates. Select the value in this column to view a list of the additional similar candidates.
Updated	Date and time the similar candidate details were most recently updated.

ERP Customization Mining snapshot field descriptions

The snapshot list in ERP Customization Mining (ERP-CM) displays information about import and export snapshots.

The snapshot features is available starting in Xanadu Store Release 2.

Snapshot list details

Column	Definition
Name	Unique, read-only name of snapshot. The name is created automatically.
Type	Classification of snapshot. The options are **Export** and **Import**.
Status	Current processing state. Options are **pending**, **in progress**, and **complete**.
Error message	If the import or export fails, an error message is displayed to explain the issue.
Created by	User that created the snapshot.
Updated	Date and time the snapshot was most recently updated.

ERP Data Hub and ERP-CM glossary

Learn about ERP (Enterprise Resource Planning) terminology and concepts that apply to ERP Data Hub and ERP Customization Mining (ERP-CM).

BAPI

A Business Application Programming Interface (BAPI) is a standard interface to the business object models in SAP products, similar to an API.

candidate

A candidate is a group of recommended remote tables that you can use to create an application based on a custom transaction (ERP root) in the system. Candidates can also be existing applications that sit on top of the legacy ERP system of record.

Good candidates for replatforming tend to be smaller applications that use data from the system of record.

candidate score

A metric for how well a custom application (ERP root) fits to a remote table in ERP Customization Mining.

connection and credential alias

The credential alias is the system connection access enabler that's maintained in the ServiceNow credential alias. The credential alias is used to access the ERP system of record.

custom fields

Custom fields are values added to a table by the customer.

entity

The table or Business Application Programming Interface (BAPI) function call that ERP Data Hub uses to read or update the system of record.

ERP application

The ERP (Enterprise Resource Planning) custom application that you choose to replatform using ERP Customization Mining is an ERP root in the ERP system.

ERP model

The ERP model represents a distinct set of features and functionalities tailored to address specific business processes or activities. An ERP model encompasses multiple tables from the system of record, as well as APIs and ETL processes, to create a holistic data set. For example, you can have one ERP model for sales orders and another for inventory.

ERP module

The functional business area on the ERP system. The ERP root can only have one selected business area.

ERP system

An ERP system represents a connection to a section of your ERP system of record. The system plays a crucial role in data synchronization, sharing, and collaboration, enabling seamless integration and operation between the ERP model and the connected ERP system.

ERP table

Individual tables (both custom and standard) in the ERP system are part of the remote tables. For example, in sales, ERP tables could include VBAK, VBAP, VBFA, VBPA, VBEP, MKPF, or MSEG.

extraction table

Extraction tables retrieve large amounts of data using a scheduled query, and use transform tables to process data for use on the ServiceNow AI Platform.

mapped value

Value of the field on the ERP system that's being mapped as an operation parameter in ERP Data Hub.

odata

In ERP Data Hub, create an OData connection to link to SAP via HTTP so data can be extracted for use in remote tables and extraction tables.

operation

Individual maintenance task performed to read or update the system of record.

remote table

A remote table is an aggregation of the system of record tables that provide value when gathered in one table. The remote table is the foundation of the ERP model. A candidate can consist of several remote tables.

system of record

The ERP system of record is where the data lives and is distributed from, for example, SAP.

Building applications with ServiceNow Studio

ServiceNow Studio provides a unified experience for all ServiceNow development activities, enabling admins and developers to extend base system solutions and create custom apps with ease. Use ServiceNow Studio to build apps and app files with integrated tools, access and edit app metadata in scoped and global apps, and package app changes for deployment, all in one powerful development tool.

Overview

https://player.vimeo.com/video/1024854672?h=2ba63a9841&badge=0&autopause=0&player_id=0&app_id=58479

Get started

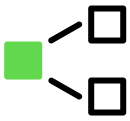
Explore

Learn about ServiceNow Studio concepts and features.

Configure

Configure environments, tools, and user access.

Build



Build applications, flows, and more using ServiceNow Studio.

Reference

Get details about ServiceNow Studio components such as fields, tables, and properties.

Give ServiceNow Studio a try

Ready to give ServiceNow Studio a try? You can test it out using your own Personal Development Instance (PDI), which requires you signing in to the Developer Site. Find out more on PDIs in the Personal developer instance guide.

Select this button to try ServiceNow Studio on a PDI

Try ServiceNow Studio for free

Try installing ServiceNow Studio now on a PDI! Login required.

Troubleshoot and get help

- Contact your company's Customer Admin to unlock or add user accounts, perform restores or zBoots, and more.

  Contact your company's Customer Admin

- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

Exploring ServiceNow Studio

ServiceNow Studio delivers a cohesive, efficient experience for mid-to-high-skilled developers to create, modify, and extend apps. ServiceNow Studio is a single place that you can go to do development work, delivering value to your business faster.

ServiceNow Studio overview

ServiceNow Studio enables you to be more efficient in your work by providing a cohesive and powerful app development experience from creation to deployment.

- Create, edit, maintain, and deploy custom and globally scoped apps and app files in a single, integrated development environment instead of having to switch back and forth between tools and builders. You can also edit base system apps and files and customize store apps.
- Search for and edit any type of metadata record using the Navigator file taxonomy.
- Package app changes for deployment using update sets, pipelines, or the Application Repository.

ServiceNow Studio features and benefits

ServiceNow Studio offers a unified app development experience that can accelerate your time to value in app creation and deployment.

ServiceNow Studio provides the following benefits for admins and delegated developers in a unified development experience.

- App and app file search: Search for any application or metadata record in your instance. For more information, see Find an app or app file using Search and Find an app or app file using the Navigator panel.
- Integrated tabs: Easily switch between apps and app files in different scopes without leaving ServiceNow Studio. For more information about working with different apps and app files, see Open apps and app files across scopes in ServiceNow Studio.
- Accelerated deployment: Create and package apps for deployment via update sets, pipelines, or the Application Repository without leaving ServiceNow Studio. For more information, see Working with update sets in ServiceNow Studio and Publish app changes to the Application Repository from ServiceNow Studio.
- Supercharged app development with Now Assist: Use the Now Assist capabilities to decrease the time it takes to create apps and app files. For more information, see Now Assist for app generation in ServiceNow Studio.

- File creation: Easily create any type of app file in any scope that you have access to edit, including the global scope. For more information, see Create an application in ServiceNow Studio and Create an app file in ServiceNow Studio.
- Taxonomy and collections: Use the ServiceNow Studio metadata taxonomy to easily find the type of files you're searching for. For more information, see ServiceNow Studio Navigator panel taxonomy.
- Integrated builders: Use many of the best low-code builders available in the ServiceNow AI Platform, such as Table Builder and flows in Workflow Studio, along with powerful app development tools. For more information, see Access integrated development tools and builders in ServiceNow Studio.

Use App Engine instead of customizations

App Engine development tools offer an excellent alternative to customizing existing ServiceNow AI Platform applications.

When your company needs to add new functionality to the ServiceNow AI Platform, you can customize existing applications, such as IT Service Management (ITSM), or create a new application using App Engine developer products, such as App Engine Studio, Creator Studio, or ServiceNow Studio. A simple guideline for which path to choose is:

- If the customization extends the intended purpose of the application, it works better to customize. For example, you can add IT functionality to ITSM.
- If the customization doesn’t extend the intended purpose of the application, it works better to create a new application using App Engine developer products. For example, do not repurpose the ITSM workflow to add a travel request workflow.

Examples of when to use App Engine

ServiceNow products work best when they’re used as they were intended. If you find yourself heavily customizing an application to repurpose it, a better plan is to create a new application using App Engine developer products.

The following scenarios demonstrate where creating a new application works better than heavily customizing an existing ServiceNow application:

- Your company has a business process that augments existing product functionality but doesn’t follow the same workflow exactly.
- You have a novel use case for an app that doesn’t align with any product workflow.
- You have a use case that could be built by heavily customizing an out-of-the-box application, but it doesn’t align with what the existing application was intended to do.

Let’s dive deeper into the last use case.

Issues with repurposing existing products

ServiceNow applications come with roles, processes, and flows that are specially tailored to their use case. For example, ITSM apps help with IT users, IT issues, IT reports, and IT cases.

You might have an idea for an app that’s similar to but doesn't exactly align with ITSM. Because ITSM gives you a starting point, you might be tempted to customize ITSM to add the new functionality. For example, ITSM tracks IT issues, and a travel app you want to create might track travel requests. While the workflows sound similar, in fact, they use very different data, different user interfaces, and the details of each workflow vary greatly. Rather than heavily customize ITSM to repurpose it, a better plan is to use App Engine developer products for the following reasons:

- Combining two workflows creates conflicts.
- Customizing applications has implications.

Combining two workflows creates conflicts

In the ITSM example, the repurposing of ITSM to include a travel workflow uses different data, different tables, different roles, and different workflows than ITSM. As ITSM, ITSM customizations, and the travel workflow grow over time:

- Their features will continue to diverge.
- Adding new functionality or fixing problems in one workflow might adversely impact the other.
- The performance of ITSM may suffer.
- The code base will grow and the two purposes of ITSM will make troubleshooting more difficult.
- Quality engineers will require two different testing frameworks.

All these issues may cause unnecessary complications, poorer performance, upgrade delays, and software problems.

Customizing applications has implications

The ServiceNow AI Platform is built to embrace customization and configuration. The ServiceNow AI Platform is flexible enough to fit your company's business needs. How you customize ServiceNow applications, however, can have significant impacts on ServiceNow support, upgrading to future ServiceNow AI Platform versions, and the functionality of the platform.

Let's start by differentiating customization and configuration:

- Customization is any change made to the code that is part of the baseline installation of a ServiceNow instance. You use code to customize applications.
- Configuration is any change you make to the behavior of a product that does not touch the code in the baseline installation of a ServiceNow instance. You can use system properties, ServiceNow products, or code to configure an application.

The following are some of the implications that result from customizing applications:

- If you add code to an application, you own it whether or not it modifies the code in the baseline installation on a ServiceNow instance.
- The platform marks all customizations and skips them when you update to a new version of the platform. That means you are responsible for manually updating the customizations. This can have a significant impact on the time and resources required to update to new platform versions.
- The ServiceNow AI Platform uses a framework that supports applications in how they process tasks, how forms are rendered in multiple browsers, and the overall user experience. Introducing customizations can have unintended consequences on this framework.
- You own the burden of testing custom code and determining if it impacts platform functionality.
- ServiceNow Customer Support cannot troubleshoot custom code or issues caused by custom code.

Customization is one of the key features of the ServiceNow AI Platform. However, over-customizing an application to repurpose it is likely to generate technical debt, lengthen your upgrade cycle, and complicate future platform upgrades because the custom code may not easily migrate to new platform versions.

Conclusion

Customization and configuration are hallmarks of the ServiceNow AI Platform that enable your company to customize workflows to fit its specific needs. Proceed with these tasks in the following order:

1. Configure ServiceNow applications as much as you can before customizing them.
2. Customize an application only when it extends the intent of the application.
3. Use App Engine developer products, such as App Engine Studio, Creator Studio, and ServiceNow Studio, to create new applications rather than customizing an application to create functionality that doesn't align with its original purpose.

For more information, see Customization vs configuration with ServiceNow Studio.

Customization vs configuration with ServiceNow Studio

There are important differences between customizing and configuring ServiceNow applications. The ServiceNow platform is built to embrace customization and configuration, but how you do so can have significant impacts on ServiceNow support, upgrading to future ServiceNow platform versions, and the functionality of the ServiceNow platform.

The general rules around customization are:

- Customize an application only if it extends the original intent of the application. For example, add IT functionality to ITSM but don't add a travel workflow. Instead of over-customizing an application, create a new application using App Engine products, such as Creator Studio or ServiceNow Studio.
- Configure as much as you can before customizing an application.
- If you add code or make other modifications to out-of-the-box functionality, you own them.

What is configuration

Configuration is the process of using ServiceNow built-in tools and features to modify an application's behavior without making changes to flows or the code that is part of the baseline installation on a ServiceNow instance.

Configuration can take the form of using ServiceNow built-in tools to add tables and more, setting instance-wide parameters, as well as using code to extend an application's functionality to meet business needs as long as the code does not modify the baseline code installation. The entire platform is designed for you to add configuration code.

If you add code, such as workflow scripts, you own it, even if it doesn't alter the baseline code installation. That includes owning the impact it has on the entire ServiceNow platform. Issues that arise from added code are beyond the scope of ServiceNow support to debug.

Reverting a configuration should not require any change to the baseline code.

Configuration examples include:

- Forms: Configure tables, fields, data types, default values, and field dependencies to configure the data you capture and display.
- UI elements: Modify layouts, add related lists, add buttons, and change field names.
- Service Catalog: Configure portals where your customers can request catalog items such as service and product offerings.
- ACLs: Restrict unauthorized users from accessing forms and data.
- System property values: Modify the application's experience for all users.

What is customization

Customization is any change made to the flows or the code that is part of the baseline installation on a ServiceNow instance. You use ServiceNow products or code to customize applications.

If you add code, you own it, even if it doesn't alter the baseline installation. That includes owning the impact it has on the entire ServiceNow platform.

Customization examples include:

- Scripting: Customize ServiceNow through scripting using JavaScript. This includes creating client scripts, server-side scripts, and business rules with intricate logic that modify the baseline code.
- Custom tables: Develop custom tables to accommodate specialized data that doesn't fit within standard tables.
- Integration: Customize integration with external systems, such as APIs and web services, for seamless data exchange.
- Widgets and portals: Create custom widgets and portals to provide unique features and user experiences.
- Workflows: Create and modify workflows using Workflow Studio. Create and manage playbooks, flows, actions, decision tables, and integrations from one design environment to automate tasks. Upgrading to a new version of a flow requires reapplying your customizations.

Tools for customization and configuration

ServiceNow offers many tools and features, such as business rules, to modify the out-of-the-box behavior of ServiceNow applications. Whether they customize or configure an application depends on how they're used. Using these tools to modify the installed code base constitutes customization. Using these tools to add code that does not modify the flows, or the installed code base constitutes configuration. In both cases, you own the code you add as well as the impact it has on the ServiceNow platform.

ServiceNow tools include:

- UI Policies: Dynamically modify the visibility of fields and attributes on a form according to user inputs.
- Business rules: Automatically trigger actions based on specified conditions.
- UI Actions: Extend and customize forms and lists by adding buttons, context menu items, or other UI elements that perform specific actions when clicked.
- Client-side scripts: Scripts that execute within the user's browser when certain actions occur on a form or a UI page.
- Server-side scripts: Scripts that execute on the ServiceNow server or database, for example, to update record fields when a database query runs.

What is personalization

Personalization is when users use out-of-the-box application tools to modify an application's look and feel only for themselves. Admins can change the look and feel for all users and that is considered configuration. Personalization examples include a user choosing to use dark mode or choosing which table columns to display.

Personalization does not change the baseline code installation on a ServiceNow instance. So, personalization does not impact customer support or interfere with upgrades to new ServiceNow versions.

Ramifications of customizing ServiceNow products

The ServiceNow platform is extremely flexible and built to embrace customization and configuration to fulfill a wide range of business requirements. How you customize ServiceNow applications, however, can have significant impacts on ServiceNow support, upgrading to future ServiceNow platform versions, and the functionality of the platform. Instead of customizing ServiceNow applications, consider using App Engine development products, such as Creator Studio and ServiceNow Studio to create new applications.

The ServiceNow platform uses a framework that supports applications in how they process tasks, how forms are rendered in multiple browsers, and the overall user experience. ServiceNow relies on the framework's integrity to develop and provide support in a consistent manner. Customizations can impair this framework, change platform functionality, and impair workflows and upgradeability.

Customizations trigger the platform to create sys_update_xml records, which are stored in the Customer Update table. The platform marks all customizations and skips the customized records when you update to a new version of the ServiceNow platform. That means you are responsible for manually updating the customizations. This can have a significant impact on the time and resources required to update to new platform versions.

Note:
The Customer Update table also contains modifications or additions to configuration metadata, for example, creating a new catalog item or a new workflow.

For more information, see the Customer Updates table. Note that the complexity of maintaining customizations dramatically increases as the number of customizations increases.

Customizing the installed code base can be costly, generate technical debt, lengthen your upgrade cycle, and complicate future platform upgrades because the custom code may not easily migrate to new platform versions. Custom code can change the standard functionality of the ServiceNow platform in unintended ways. Evaluate demands for customization carefully and only resort to customization where there is clear business value and no alternative. Wherever possible, avoid customization by using configuration instead.

If you customize a product:

- You are responsible for maintaining the customization going forward.
- Customer Service and Support will not support problems caused by custom code. If it is the cause of the problems, the support team will likely advise you to revert to the out-of-the-box code.

What is the Customer Service and Support stance on supporting customization

The ServiceNow Customer Service and Support stance on customization is if you add code, you own it and its consequences. Why? Customer Support is not privy to your custom business logic, doesn’t know what the expected behavior should be, is unable to reproduce the issue on an out-of-the-box instance, and customer support engineers are not certified implementation specialists, so they are not certified to review customized code logic.

Alternatives to customization

If you have requirements and ideas for enhancements, instead of customizing the installed code base, you can:

- Use configuration instead of customization.
- Submit an Enhancement Request to the ServiceNow development team. Each request is evaluated and, if approved, will be incorporated into a future release.
- Create an app using App Engine developer products to handle desired functionality.

When to use App Engine developer products instead of customizing

When your company needs to add new functionality to the ServiceNow platform, you can customize existing applications, such as ITSM, or create a new application using App Engine developer products, such as Creator Studio or ServiceNow Studio. A simple guideline for which path to choose is:

- If the customization extends the intended purpose of the application, it works better to customize. For example, you can add IT functionality to ITSM.
- If the customization doesn't extend the intended purpose of the application, it works better to create a new application using App Engine developer products. For example, do not repurpose the ITSM workflow to create a travel request workflow.

For example, ITSM is designed to handle IT issues. To customize it to handle travel requests goes beyond the original intention of ITSM. Because IT and travel requests have different workflows, it's better to create a travel request app using App Engine developer tools, such as Creator Studio and ServiceNow Studio, instead of customizing ITSM.

For more information, see Use App Engine instead of customizations.

Examples of when to use App Engine developer products

ServiceNow products work best when they're used as they were intended. If you find yourself heavily customizing an application to repurpose it, a better plan is to create a new application using App Engine developer products.

The following scenarios demonstrate where creating a new application works better than heavily customizing an existing ServiceNow application:

- You have a novel use case for an app that doesn't align with any product workflow.
- You have a use case that could be built by heavily customizing an out-of-the-box application, but it doesn't align with what the existing application was intended to do.
- Your company has a user group or business process that should be separate from the OOTB product workflow.

Guidelines for customizing ServiceNow products

If you must make a customization, consider the following suggestions:

- Maximize configuration options first.
- Avoid copying objects. Instead, update objects in place wherever possible, except for Service Portal widgets and other items designated to be reused.
- Default to "add before edit." This means that you should, for example, add fields to forms rather than change the type of an existing field. Avoid using the same names as out-of-the-box objects, methods, or classes when adding them.
- Minimize the number of fields you add to a form. The more fields you have on a form, the longer it may take to load.
- Export original records as backups before customizing them. Keep track of their out-of-the-box sys_id's in case you must restore them in the future.

- Use scoped applications as your default for any new custom development.
- Document all customizations. Add comments explaining why you customized (including business justification). Review all comments before upgrading to determine if you can revert to out-of-the-box code.
- Create tests for all customizations. Write Automated Test Framework (ATF) tests for all customizations where possible.
- Use HealthScan regularly to identify unnecessary customizations.
- Customizations should be made to baseline objects where necessary so that conflict resolution and decision-making can be appropriately recorded in the updates. Hidden customizations may cause administrators to overlook updates in future assessments in case reverts or merges are necessary.
- Test your customization for all use cases. Include performance testing and the introduction of unintended consequences.
- Administrators are responsible for verifying that their customizations work after a ServiceNow platform upgrade, and for keeping track of what customizations are made.

Handling customizations when you upgrade

Customizations trigger the platform to create sys_update_xml records, which are stored in the Customer Updates table. These records are not updated during platform version upgrades. ServiceNow marks them as skipped records in the ServiceNow Upgrade Monitor. To make sure they're successfully ported to the upgraded instance, you must manually process the skipped changes. For more information, see the Customer Updates table.

Assuming you've documented all your customizations—including the business justification—take your documented inventory and compare it with the skipped records identified in the Upgrade Monitor. After filtering out low-risk changes that have resulted in skipped records (for example, field labels or form layouts), you'll need to decide whether to:

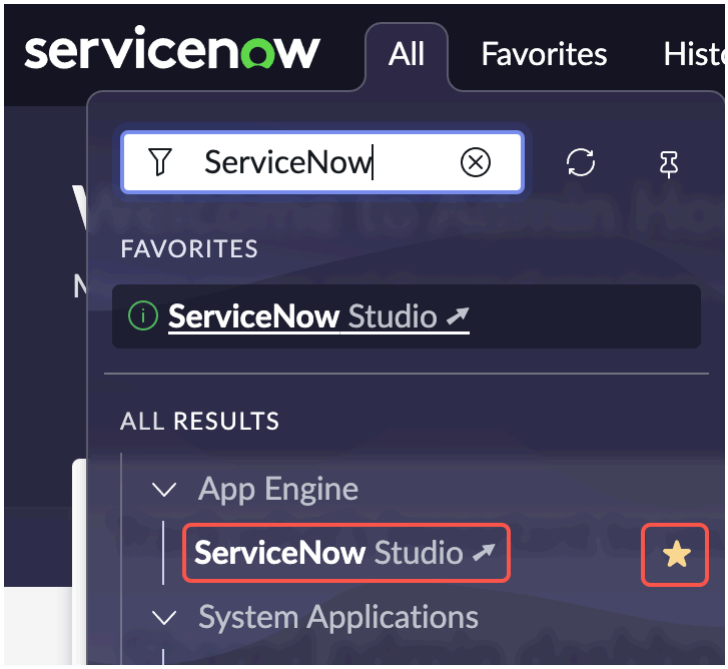
- Retain each customization
- Revert to out-of-the-box
- Merge your customization with the base system to resolve conflicts

Access ServiceNow Studio

Access ServiceNow Studio from the ServiceNow AI Platform All menu or from App Engine Studio.

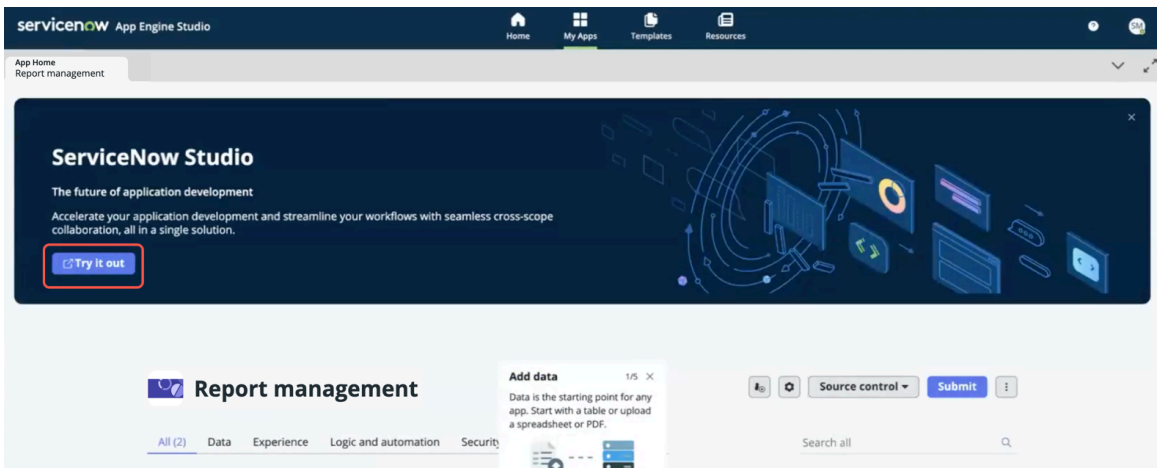
Open ServiceNow Studio through the All menu

On the platform, navigate to **All** > **App Engine** > **ServiceNow Studio** or search in the filter navigator for `ServiceNow Studio`. Open the application directly from the search results. You can also favorite ServiceNow Studio for quick access from the **Favorites** tab.



Open ServiceNow Studio from App Engine Studio

If you're creating apps in App Engine Studio (AES) and want a more advanced app development environment to work in, you can open your app in ServiceNow Studio. When you open your app in AES, a banner appears with information about ServiceNow Studio. Select **Try it out** to open your app in ServiceNow Studio.



ServiceNow Studio quick start

Familiarize yourself with ServiceNow Studio by completing several tasks. These tasks enable you to navigate around the interface and learn where to complete different types of activities.

Quick start tasks

Complete these tasks to familiarize yourself with ServiceNow Studio.

ServiceNow Studio quick start tasks

Task	Description
Find an app or app file using Search	Search from any page in ServiceNow Studio for quick access to your metadata.
Find an app or app file using the Navigator panel	Locate files or metadata items in the taxonomy.
Bookmark apps and files in ServiceNow Studio	Access frequently used apps or app files in one location.
Access recently opened apps and app files in ServiceNow Studio	View and open files or apps that you have been working on recently.
Navigate directly to a table in ServiceNow Studio	Open different table views directly by using shortcut commands like **.list**, **.do**, and **.config** after table names in the Search bar.
Open apps and app files across scopes in ServiceNow Studio	Access global and scoped apps and files in ServiceNow Studio. Scoped apps are color-coded by tab for quick reference.
Access integrated development tools and builders in ServiceNow Studio	Open any file type from the File Categories menu to access a suite of integrated development tools.
Create an update set in ServiceNow Studio	Package changes to your application for deployment.
Publish app changes to the Application Repository from ServiceNow Studio	Publish changes to the Application Repository. If your company has a deployment pipeline set up, you can also deploy through the pipeline in ServiceNow Studio.

Find an app or app file using Search

Search for and open an app or app file in ServiceNow Studio. You can also activate code search to search for snippets of code within select metadata types on your instance.

Before you begin

Role required: admin or delegated developer

Procedure

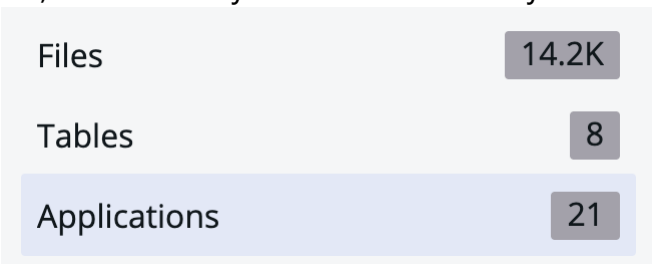
1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Choose from one of the following options to select the Search bar.
   - If you're on the ServiceNow Studio home page, select the Search icon ( ) in the Navigator panel, or select in the main Search bar.

   - If you're on any page other than the home page, such as Tools or Deployment, select the Search icon ( ) in the left side navigation.

3. **Optional:** Activate code search by selecting the switch next to **Activate code search**.

   Note:
   Code search enables you to search for snippets of code within specific application file types. For more information about using code search, see Find an app or app file using code search.

4. Enter your app or app file name.
   If you're familiar with the ServiceNow AI Platform search capabilities and want to navigate directly to a table, follow the instructions in Navigate directly to a table in ServiceNow Studio.
5. Select your app or file from the list that appears.
6. **Optional:** To expand and filter your search results, select **Enter** to open the **Search results** tab, then select any of the filters to narrow your list of search results.



Find an app or app file using the Navigator panel

Use the Navigator panel in ServiceNow Studio to find and open an app or app file.

Before you begin
Role required: admin or delegated developer

About this task
The Navigator panel in ServiceNow Studio is a powerful tool that you can use to find apps and app files. The Navigator panel is organized alphabetically and by metadata type. For example, if you want to find an audio app file, you can locate the file in the Navigator panel under **File Categories** > **Content** > **Audio**.

For more information about the types of files you can find in the Navigator panel, see ServiceNow Studio Navigator panel taxonomy.

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. On the left side of ServiceNow Studio, verify that the Navigator panel is open.
3. Select the icon that corresponds to the type of file that you want to find.
   - To find an app, select the Apps icon ( ).
   - To find an app file, select the File Categories icon ( ).
4. Choose from the following options to refine the list of apps or app files that appear.
5. **Optional:** Expand the list of apps or app files by selecting **Open list** link.
   Selecting **Open list** opens a new tab in ServiceNow Studio that displays a complete list of apps or app files.
6. Select the app or file to open it in a new tab in ServiceNow Studio.

Bookmark apps and files in ServiceNow Studio

Easily bookmark apps and app files from several locations for quick access in ServiceNow Studio.

Before you begin

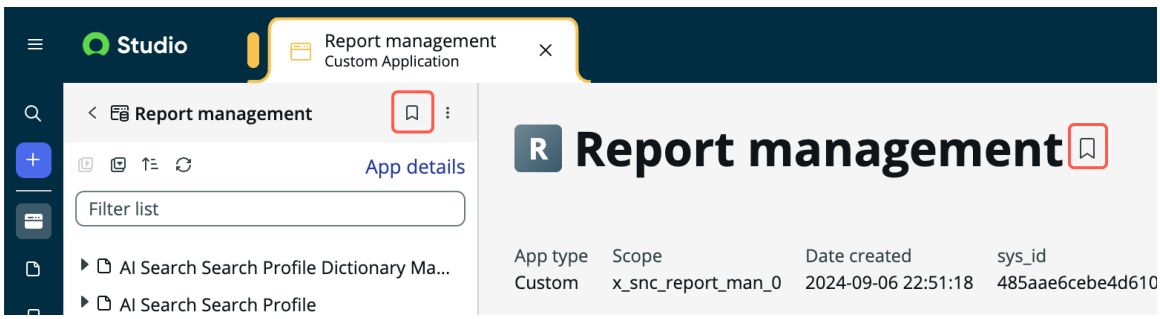
Role required: admin or delegated developer

About this task

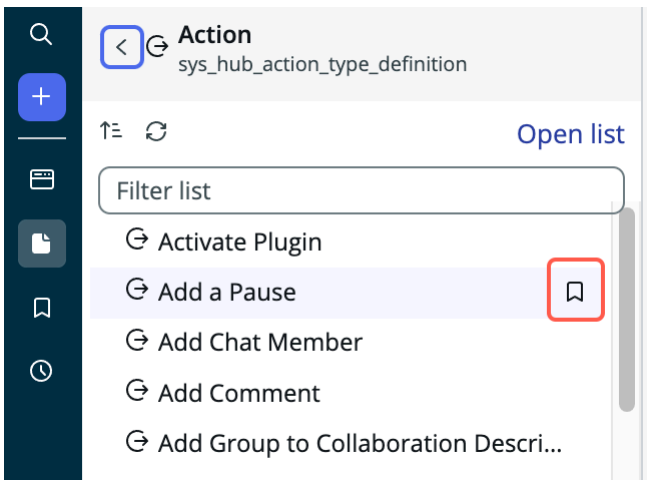
Note:
Bookmarks and favorites are distinct ServiceNow capabilities. Bookmarks exist only within ServiceNow Studio, while favorites are accessible throughout your instance. When you bookmark apps or files in ServiceNow Studio, your bookmarks do not automatically become favorites. Favorites also do not automatically become bookmarks.

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Bookmark your apps and files from different locations in ServiceNow Studio.
   - To bookmark an open app, select the Add to bookmarks icon ( ) next to the app name.



   - To bookmark an app or file in the Navigator panel, hover over it and select the Add to bookmarks icon ( ).



3. View bookmarked apps and files in the **Bookmarks** tab of the Navigator panel.

Access recently opened apps and app files in ServiceNow Studio

Access your recently opened apps and app files from several locations in ServiceNow Studio.

Before you begin
Role required: admin or delegated developer

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Access your recently opened apps and files from either the ServiceNow Studio home page or the Navigator panel.

Navigate directly to a table in ServiceNow Studio

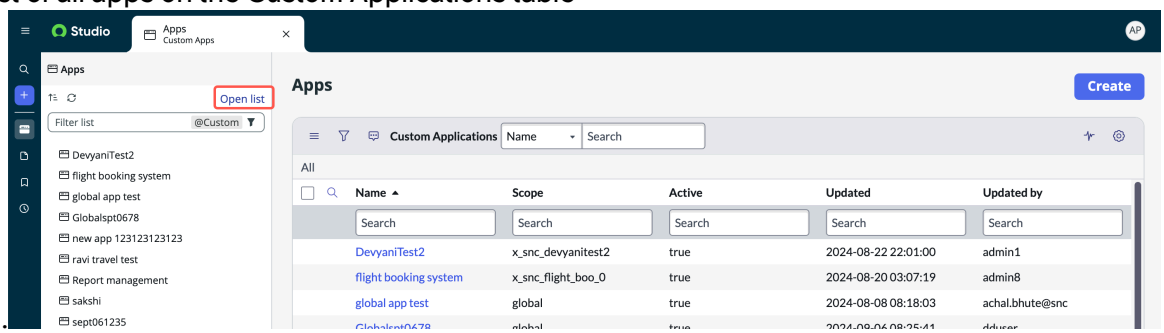
Navigate directly to the list view of a table in ServiceNow Studio by using the **Open list** feature in the Navigator panel or search shortcuts in the main search.

Before you begin
Role required: admin or delegated developer

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Open a table directly in one of two ways.
   - In the Navigator panel, with any application or file category selected, select **Open list**.

     The list opens in a new integrated tab. In the example below, this is a list of all apps on the Custom Applications table [sys_app].



   - In the Search bar on the home page, enter one of the following inputs and press **Enter**.

> **Important:**
> The table name must match the name in the dictionary entry for the table. For more information about the primary table associated with each metadata type, see ServiceNow Studio Navigator panel taxonomy.

When you enter an input, the table view opens in a new integrated tab.

Table opening shortcuts

Input	Behavior
`<table name>.list`	Opens the list view of the table in the same window or tab.
`<table name>.form` or `<table name>.do`	Opens the form view of the table in the same window or tab.

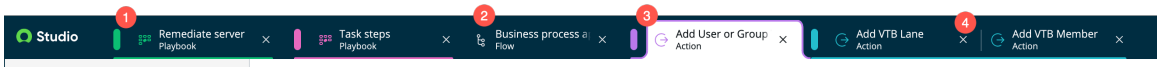
Input	Behavior
`<table name>.config`	Opens the configuration view [personalize_all.do] of the table in the same window or tab.
`<table name>.filter`	Opens an empty list view of the table in the same window or tab, so that you can apply filters without loading the list. This is helpful for large lists that take a long time to load.

For example, if you want to open the list of all users in your organization, you could search for the Users table [sys_user] using the search input `sys_user.list`. To add a new user quickly, search for `sys_user.do` to open a new form view of the Users table. To change the Users table configurations, search for `sys_user.config`. To open the Users table without loading any of the records, search for `sys_user.filter`.

Open apps and app files across scopes in ServiceNow Studio

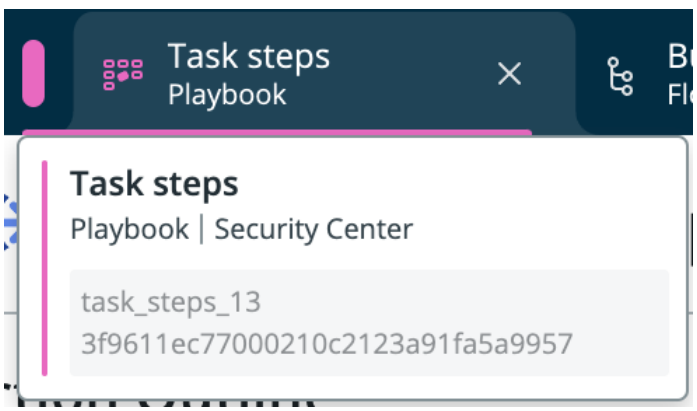
In ServiceNow Studio, you can open and edit apps and app files in any scope, global or custom. The scope that an app is associated with displays at the bottom of the screen and updates automatically when you switch to an app in a different scope.

When you open any app or file in ServiceNow Studio, it may open in an integrated tab. The following diagram and list outline key information about integrated tabs in ServiceNow Studio.

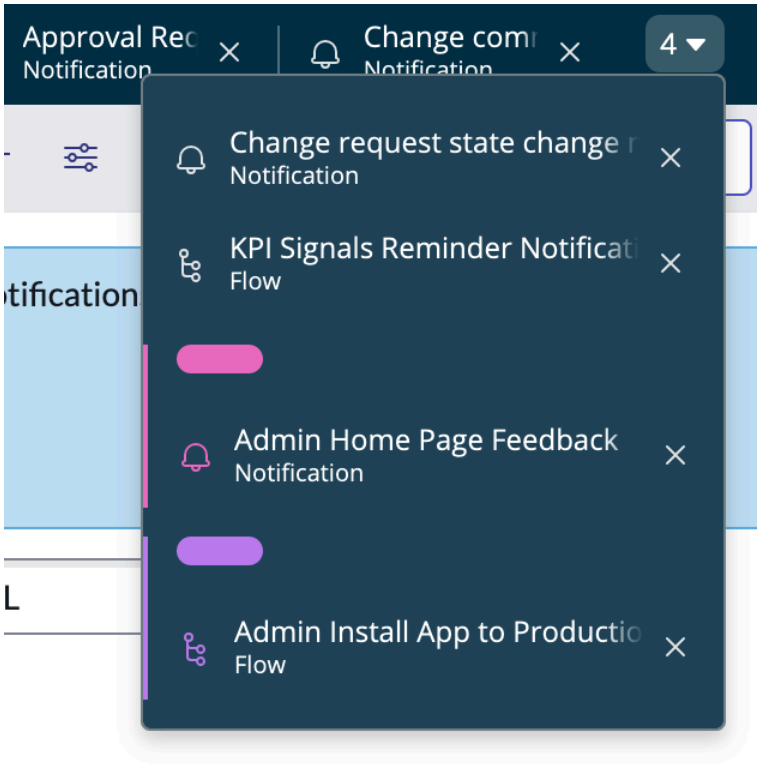


1. Any file in a scoped app opens in a color-coded tab with information about what kind of file it is. In this instance, the playbook opened in an integrated tab, where you can update it using the Workflow Studio interface.
2. Tabs without colors indicate that the file is in the global scope. You can edit global files in ServiceNow Studio.
3. Any tab that's actively open in the canvas shows a contrasting color to indicate the open state.
4. Tabs that are color-coded and grouped are in the same scope. In this example, both actions are in the same application, so they open in the same color tab.

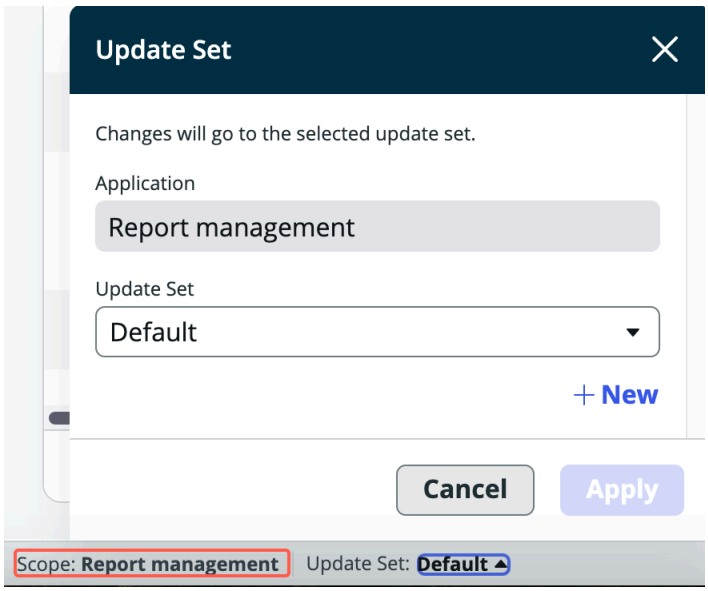
See what scope a file is in by hovering over the tab. In this image, the scope is **Security Center**.



Depending on the width of your browser screen, you may see an overflow tab appear for apps or files you've opened. The scopes for each file are grouped and color coded just like the open tabs. Select any file to open it in the canvas.

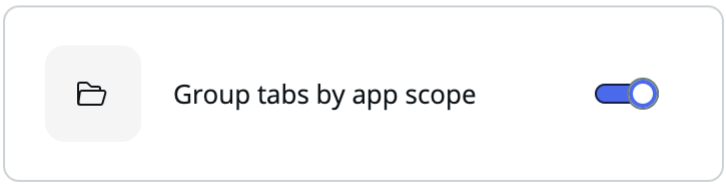


You can also see the scope and update set associated with each file at the bottom of the screen. You can switch update sets here or create one to package your changes.



If you want to ungroup the tabs, select your user preferences on the home page, and select **Studio settings**. Use the **Group tabs by app scope** toggle switch option to group or ungroup the tabs by scope.

Studio settings



Access integrated development tools and builders in ServiceNow Studio

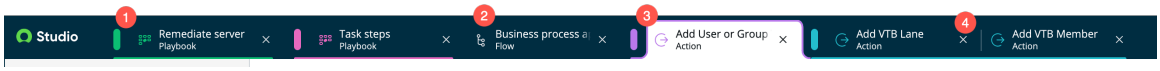
ServiceNow Studio contains several robust development tools and builders to help you expedite the app development process. Open and easily switch between tools and builders as you develop your applications.

Access integrated tools and builders through File Categories

Many of ServiceNow's tools and builders are integrated with ServiceNow Studio, so that all of your app development can happen in one interface.

Most integrated tools and builders, such as Table Builder, Workflow Studio, and flows in Workflow Studio, open in new tabs within ServiceNow Studio and are grouped if they're in the same scope. Some builders, such as Mobile App Builder, open in new browser tabs. Each file type in ServiceNow Studio has either a specified builder that it opens in, or the file opens in the classic UI16 form view. For more information on what tool each file type opens in, see ServiceNow Studio Navigator panel taxonomy.

To access these tools and builders, open ServiceNow Studio, and select the **File Categories** in the Navigator panel. Select the type of file that you'd like to open, and the file opens in either an integrated tab or a new browser tab. The following diagram and list outline key information about integrated tabs in ServiceNow Studio.



1. Any file in a scoped app opens in a color-coded tab with information about what kind of file it is. In this instance, the playbook opened in an integrated tab, where you can update it using the Workflow Studio interface.
2. Tabs without colors indicate that the file is in the global scope. You can edit global files in ServiceNow Studio.
3. Any tab that's actively open in the canvas shows a contrasting color to indicate the open state.
4. Tabs that are color-coded and grouped are in the same scope. In this example, both actions are in the same application, so they open in the same color tab.

For more information about the tools and builders you can use for app development, see Integrated development tools for ServiceNow Studio. For more information about what a builder is, see Builders in ServiceNow Studio.

Switching between tools and builders

During your app development, you may open one or many integrated development tools or builders. For example, you can build a flow or playbook and then use UI Builder to build a custom web experience for your users.

Because ServiceNow Studio seamlessly switches between tools and interfaces, you just have to open the file that you want to edit, and the correct integrated tool or builder opens your file.

Create an update set in ServiceNow Studio

Create an update set as you work in ServiceNow Studio.

Before you begin

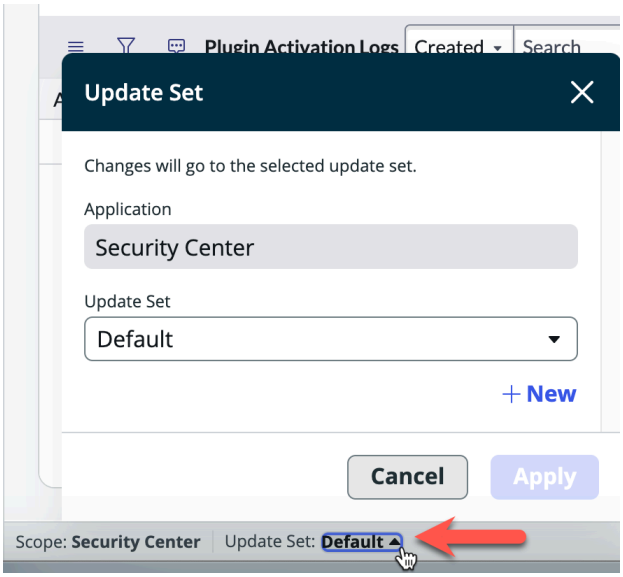
Role required: admin or delegated developer

About this task

Create an update set without having to leave ServiceNow Studio. For more information about creating an update set on the ServiceNow AI Platform, see Get started with update sets.

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Open the application that you have changes for that need a new update set.
3. At the bottom-left corner of the canvas, expand the update set dialog by selecting the current update set from the status bar.



4. Select **New**.
5. Enter a name for the new update set.
6. Select **Save**.
7. Select **Apply**.
   The changes you made in your app go to the new update set.

Publish app changes to the Application Repository from ServiceNow Studio

Publish changes that you have made to an app in ServiceNow Studio to the Application Repository (App Repo).

Before you begin

Role required: admin or delegated developer

About this task

The App Repo is one option ServiceNow offers for app deployment. You can also create update sets to publish changes, or if your company has a deployment pipeline set up, you can deploy

changes through the pipeline in ServiceNow Studio. For more information about the App Repo, see ServiceNow application repository.

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Open the **App details** page for the application that you want to update from either the ServiceNow Studio home page or the Navigator panel.
   - On the ServiceNow Studio home page, select the **Deployment** tab. Then select **Applications** and select your application from the **Name** column.
   - In the Navigator panel, select the Apps icon (). Then select your application and select **App details**.
3. On the App details page, select **Publish**.
4. In the **New version** field, enter a version number for the changes.
5. In the **Release notes** field, add notes about the changes that you're publishing.
6. Select **Publish**.
   You have submitted your changes for review and approval. Once your changes are approved, your updates are accessible to all instances of your application.

ServiceNow Studio user interface

Learn about the ServiceNow Studio user interface and how to customize it to your specifications.

ServiceNow Studio is an application that enables you to build, manage, and deploy applications all in one interface. You must be an admin or a delegated developer with access to an application to access ServiceNow Studio.

ServiceNow Studio home page

From the home page, you can access all of the tools and services in ServiceNow Studio.

- The Navigator panel enables you to see a quick view of all your apps, files, bookmarks, recents, and tools. You can filter or sort the lists, and you can open any list in the platform view by selecting **Open list**.

Navigator panel tools

Navigation item	Description
Menu ()	Option to expand or collapse the Navigator panel.
Search ()	Option to search for any file, app, metadata, or tool in any scope on the platform.
Create ()	Option to create a new app or file.
Apps ()	Option to view all apps.
File Categories ()	Option to view, sort, and filter all files and metadata.
Bookmarks ()	Access point for all your favorite files and apps.

Navigation item	Description
Recent ( )	Access point for all your most recently accessed files.

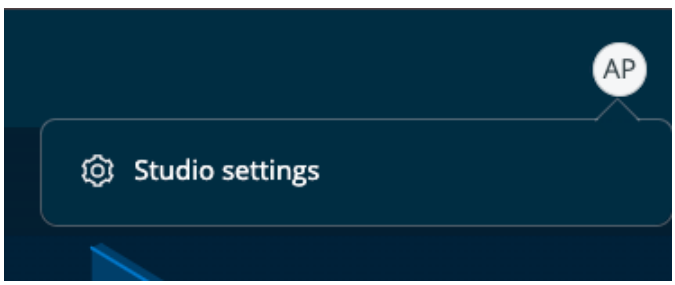
- You can show/hide and pin/unpin the Navigator panel in ServiceNow Studio by using the icons in the side panel. You might want to hide the panel for full-screen editing capabilities when you're working on an app.

- The toolbar provides another access point for the home page, Tools page, and Deployment.

Tool bar

Page	Description
Home	Main landing page for ServiceNow Studio. Access the Navigator panel, recently opened files and apps, and other tools and resources.
Tools	Access point for documentation related to the tools and builders integrated into ServiceNow Studio. For more information, see Integrated development tools for ServiceNow Studio.
Deployment	Landing page for viewing, managing, and deploying apps using update sets and accessing applications for deployment through pipelines or the Application Repository. If you create a new update set, you can see it on the **Deployment** page. Applications can also be deployed directly from the App details page for each app.

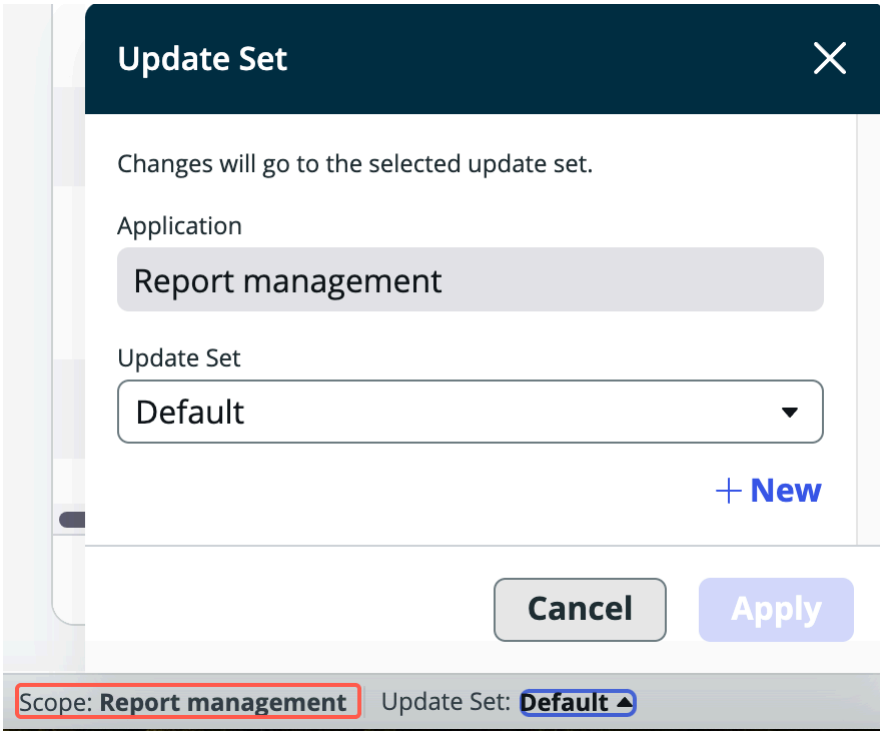
- The user preferences menu provides access to ServiceNow Studio settings.



- The Create button enables you to quickly create apps and app files.

See scope and update set

If you're in an app or file, you can see the scope of the file in the status bar. You can also see the update set associated with the application in the status bar. If you need to change the update set or create a new one, open the default update set and change your selection right within the app.



For more information, see Working with update sets in ServiceNow Studio.

Builders in ServiceNow Studio

Builders are specialized tools that provide an intuitive and efficient interface for editing metadata records primarily associated with tables extending sys_metadata. In ServiceNow Studio, you can use builders to create a variety of file types to support your app development.

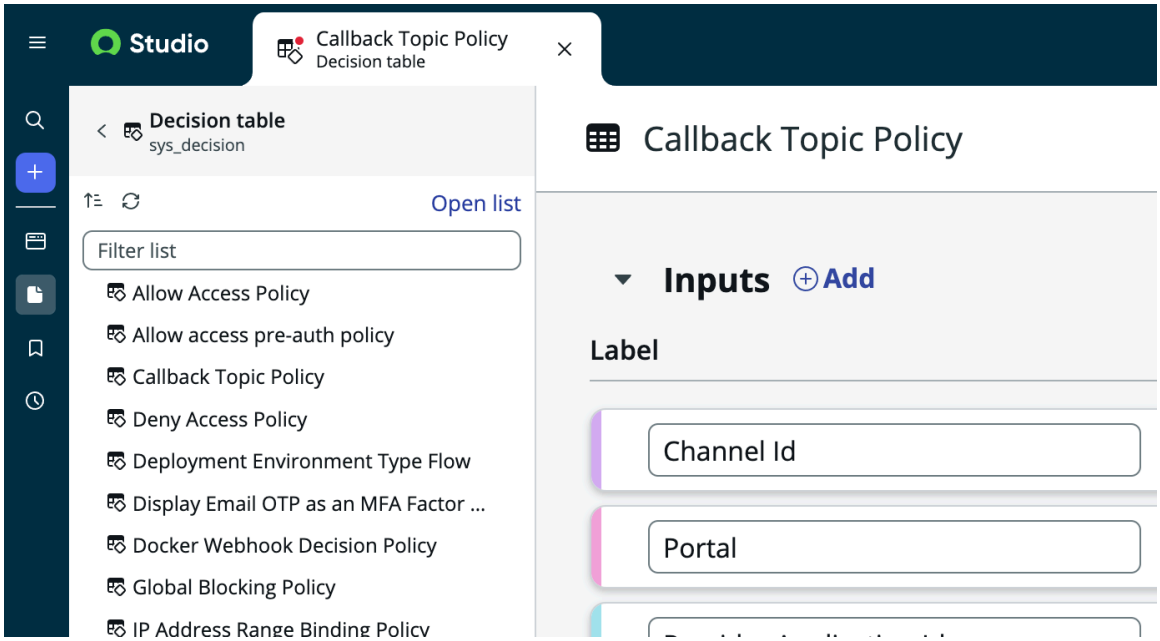
Builders focus on enhancing user experience by offering features such as inline editing, customizable views, and integrated collaboration. Each builder remains agnostic of design libraries or implementations, ensuring adaptability to evolving technologies and frameworks.

When you open files like decision tables, flows, or tables in ServiceNow Studio, the files open in an integrated tab in their designated builder tool. Some other file types like mobile app configurations open in a new browser tab. Note that you must create or open a file or application to access a builder. You cannot open builders on their own in ServiceNow Studio.

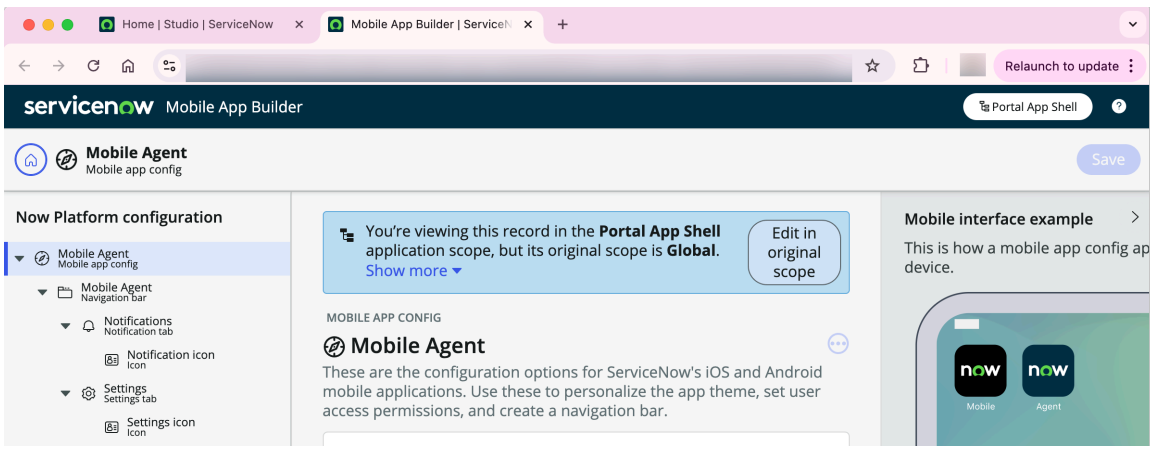
For a full list of the builders integrated in ServiceNow Studio, see Integrated development tools for ServiceNow Studio.

How builders open in ServiceNow Studio

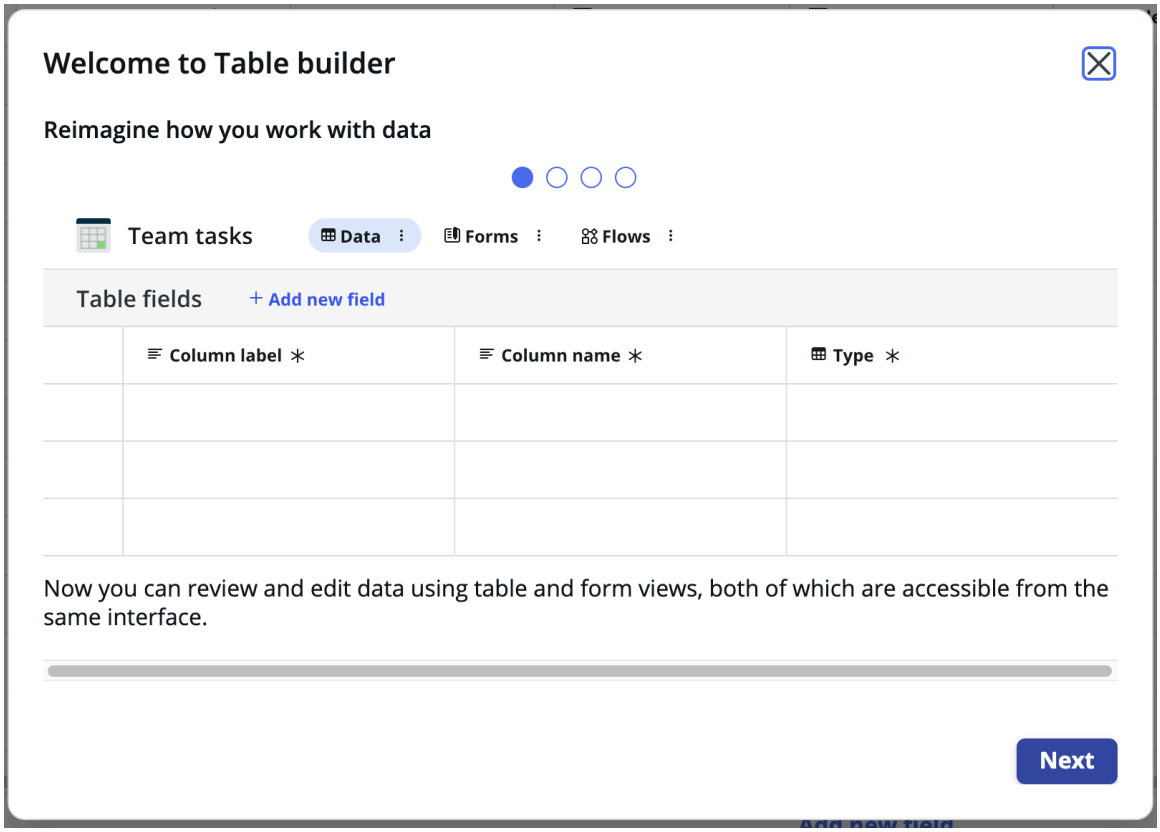
Decision tables and many other types of automations open in integrated tabs.



Mobile App Builder opens in a new tab.



When you open some tools like Table Builder, you may see a short tutorial to help you learn the tool.



Integrated development tools for ServiceNow Studio

ServiceNow Studio provides instant access to several tools available in the ServiceNow AI Platform development suite. As you create apps using ServiceNow Studio, you can extend their capabilities with these powerful development tools.

Development tools

Use the following integrated development tools to build apps, flows, processes, tables, and more in ServiceNow Studio.

While some tools have the word "Builder" attached to their name and others don't, all tools listed below enable you to build different parts of apps. Some builders are automation tools, while others enable you to build your user interface. For more information about builders, see Builders in ServiceNow Studio.

Automation

Builder	Description
Decision tables in Workflow Studio	Decision tables in Workflow Studio embed business logic into a series of if-then decision rules. Decision tables read data from inputs and evaluate the data according to specified conditions. When all the conditions for a decision rule are met, the decision table returns one or more results.
Flows, subflows, and actions in Workflow Studio	Flows, subflows, and actions enable process owners to automate work. Build multi-step

Builder	Description
	flows from reusable components without having to code.
Playbooks in Workflow Studio	Playbooks enable process owners to author cross-enterprise workflows and create a single, unified process. You can also use playbooks to provide end users with a simplified, task-oriented view of your process.

Data

Builder	Description
Form Builder	Customize any ServiceNow AI Platform record or form using Form Builder. Form Builder is accessed most often through Table Builder on the Forms tab.
Table Builder	Design tables, forms, and flows visually using a single user interface.

Integration

Builder	Description
Integration Hub Imports	Integration Hub - Import enables you to configure, run, and schedule your data imports all through a single simplified interface. It also consolidates multiple data integration capabilities into a single environment, eliminating the need to create and manage several forms throughout the platform. Its step-by-step experience guides you through the configuration of data sources, target tables, and data mapping. You can schedule your data imports or run them on demand.

Mobile builder

Builder	Description
Mobile App Builder	Configure how ServiceNow applications display on mobile devices.

User interface

Builder	Description
Catalog Builder	You can create or edit a catalog item (catalog item or record producer) using a visual and

Builder	Description
	guided experience along with specified restrictions. The catalog builder experience enables you to delegate the creation and maintenance of the catalog.
Service Portal	Create or update the portal page layout and add or remove widgets on the page.
Service Catalog	Create or edit a catalog item, which can be a record producer, an order guide, or a content item.
Survey designer	Create survey categories and questions, configure their details, and publish the survey to specific users or groups.
UI Builder	UI Builder is a web user interface builder. Use UI Builder to build pages for App Engine Studio generated workspaces or custom web experiences using Next Experience Components and custom web components.
Workspace Builder	Workspace Builder enables you to automate processes in a single design environment.

Other tools

Builder	Description
Script Debugger	The Script Debugger enables users with the script_debugger role to debug server-side JavaScript.

ServiceNow Studio and legacy products

ServiceNow Studio improves upon the functions of several other tools that you may have used to develop apps on the ServiceNow AI Platform. If you previously used Guided Application Creator or the legacy Studio product to create apps, try using ServiceNow Studio to continue your app development process.

Upgraded features

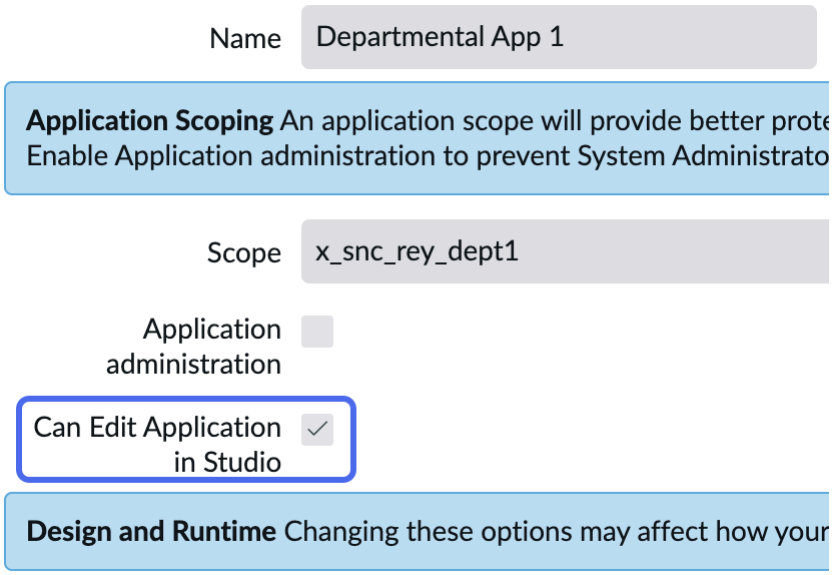
The intuitive navigation and taxonomy of ServiceNow Studio will help you efficiently find, organize, and edit your work. When it comes to creating or editing objects, every development capability you need is accessible in one place, enabling you to access, switch between, and leverage the right tool for the job every time. Once your app is ready to go live, you can effortlessly manage your deployments using any of the app life cycle options ServiceNow offers, such as pipelines or update sets. ServiceNow Studio also offers native support for Now Assist, giving you the ability to leverage the latest GenAI capabilities ServiceNow offers along with innovative low-code tools.

ServiceNow Studio and changed functionality

Some aspects of the upgraded ServiceNow Studio behave differently than legacy Studio. In the record for each app created on the ServiceNow AI Platform, there is an option to allow the app to be edited in legacy Studio. However, this option does not apply to the new ServiceNow Studio.

Any global, scoped, or custom app you have permission to edit can be edited in ServiceNow Studio.

The **Can Edit Application in Studio** option on each app record does not apply to the new ServiceNow Studio.



Using ServiceNow Studio with Guided Application Creator (GAC) roles

When you start using ServiceNow Studio, some features become available only after you create a scoped application. If you're new to using ServiceNow Studio, you may notice that the Search bar and **Create** > **App/File** aren't visible initially. This is expected behavior in ServiceNow Studio.

To get started, select **Create** and continue creating your scoped app. On the App details page for the new application, you can create metadata (or application files) by selecting **Create**. The Create file modal opens, and you can then create any file type you need and associate it to your app.

After you have created your application, return to the ServiceNow Studio home page, and refresh the browser window. You can now use the Search bar and **Create** > **App/File** features during your scoped application development. For more information about roles, see Assign a role to a user.

Configuring ServiceNow Studio

ServiceNow Studio is available from the ServiceNow Store. ServiceNow Studio leverages existing platform development tools. No additional configuration is required to use these tools in ServiceNow Studio.

Domain separation and ServiceNow Studio

Domain separation is not supported for ServiceNow Studio. Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Support level: No support

- The domain field may exist on data tables but there is no business logic to manage the data.
- This level is not considered domain-separated.

For more information on support levels, see Application support for domain separation.

Installing ServiceNow Studio

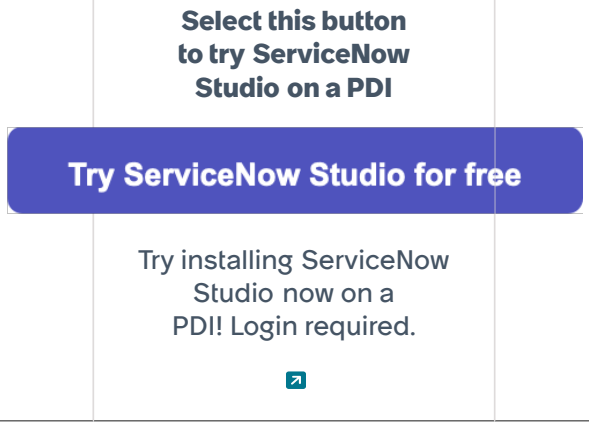
ServiceNow Studio is available for you to use on the ServiceNow AI Platform. You can also access premium features with an App Engine Enterprise license.

ServiceNow Studio is free to install and configure from the ServiceNow Store. If your company has an App Engine Enterprise license, you can install additional plug-ins to use in ServiceNow Studio. For instructions on installing ServiceNow Studio and other products from the ServiceNow Store, see Install a ServiceNow Store application.

After ServiceNow Studio is installed on an instance, no other configurations must happen.

Give ServiceNow Studio a try

Ready to give ServiceNow Studio a try? You can test it out using your own Personal Development Instance (PDI), which requires you signing in to the Developer Site. Find out more on PDIs in the Personal developer instance guide.



Premium features available with an App Engine Enterprise

If your company has an App Engine Enterprise license, you can access these additional premium features in ServiceNow Studio.

- Create a table from a spreadsheet. For more information, see Use a spreadsheet to add data.
- Create a table from a PDF. For more information, see Use a PDF to create data tables.
- Access the UI Template library.
- Access App Engine Studio Catalog templates. For more information, see Create your app using an application template.
- Access Workspace Builder. For more information, see Add a workspace.
- Access Workspace UI templates.
- Deploy scoped apps using the App Engine pipeline. For more information, see Managing app deployments using Pipelines and Deployments.

ServiceNow Studio instance strategy

You should install ServiceNow Studio on all ServiceNow instances where you develop applications.

You'll need to establish your company's instance strategy for ServiceNow Studio. Some of the decisions you need to make include:

- Whether you want to enable anyone within your company to have access to ServiceNow Studio to start building apps.
- Whether you want to grant access to a select group of people.
- Whether you want to grant access on a case-by-case basis by setting up a form where individuals can complete information about the app that they're looking to build. Your admin then decides whether to give those individuals access to build that app.

A non-production instance that is similarly configured to your production instance may be the best candidate for your test environment. You can then more accurately find issues that may arise if the application is deployed to production.

Note:
If you plan on cloning your production instance to one or more non-production instances, you should also install the ServiceNow Studio product on your production instance prior to cloning. For more information, see System clone.

After you have established your instance strategy, you must also complete the following tasks.

1. Establish and automate your approval or review process.
2. Decide which non-production environment ServiceNow Studio will run on. Because ServiceNow Studio is a product that runs on your non-production environment, you need to select one to run it on.
3. Determine which method to use for promoting apps from a particular non-production instance to your test instance, and then finally to production where the app will be running live. You can deploy apps using update sets, pipelines, or the Application Repository in ServiceNow Studio.

Components installed with ServiceNow Studio

When you activate the ServiceNow Studio plugin, various components like tables are automatically installed.

Tables and roles installed with ServiceNow Studio

One table is shipped directly with ServiceNow Studio: Resources [sn_sns_resources]. This table populates the Resources section at the bottom of the home page.

No specific roles are shipped with ServiceNow Studio, but Admin and Delegated Developer roles are used. For more information, see ServiceNow Studio personas and roles.

Plugin installed with ServiceNow Studio

The ServiceNow Studio [sn_sns] plugin is installed with several dependencies.

ServiceNow Studio plugin

Plugin [ID]	Description	Dependencies
ServiceNow Studio [sn_sns]	ServiceNow Studio provides a unified experience for all ServiceNow development activities.	• sn_udc • sn_studio_commons • sn_app_obj_wizards

ServiceNow Studio plugin (continued)

Plugin [ID]	Description	Dependencies
		• sn_deploy_pipeline • sn_workflow_studio

ServiceNow Studio personas and roles

Admins and delegated developers have access to work in ServiceNow Studio.

ServiceNow Studio personas

Admins and delegated developers have different capabilities in ServiceNow Studio. Admins can delegate people to work on certain apps and app files by providing delegated developer permissions.

Roles and capabilities

Role	Permissions	Capabilities
Delegated developer	Delegated developers have access to just the app or apps they're delegated to. These permissions may vary based on your configuration.	• Update apps that they are given delegated development permissions for. • Create an update set to package changes. • Create and update app files and other metadata records. • Submit an app for deployment through update sets, pipelines, or the Application Repository.
Admin	Admins can review and approve tasks related to custom application development.	• Create apps in ServiceNow Studio. • Edit existing apps and app files or delegate development to someone else. • Create, edit, and manage update sets. • Update existing metadata records.

Manage access to ServiceNow Studio

Admins and delegated developers have access to work in ServiceNow Studio. Manage access to develop apps and app files in ServiceNow Studio by assigning delegated development permissions.

Delegated development enables designated users without a system admin role to develop or deploy applications on the ServiceNow AI Platform. If you have the application-specific admin role or the system-level admin role, you can delegate application development to designated developers at the application level.

Note:
Currently, only admins can create apps in ServiceNow Studio, but delegated developers can work on existing apps and app files.

For more information about delegated development, see Delegated development and deployment. If you're an admin, you can also assign collaboration permissions to your development team. For more information, see Collaborating on apps using ServiceNow Studio.

Collaborating on apps using ServiceNow Studio

You can collaborate, or share app development in ServiceNow Studio with other people in your company.

Collaboration is delegated development

Collaboration, also referred to as delegated development, builds on the existing delegated development feature set in the ServiceNow AI Platform. It enables developers to invite other developers into apps so that they can co-create and develop the app together. Depending on your permissions, you can invite others to collaborate on an app with you, or request to join someone else's app. For more information on delegated development, see Delegated development and deployment.

There are two standard types of collaborators when you co-develop an app with other people: owners and editors. Admins can create a custom collaboration role by adjusting permissions.

Requirements for collaboration

ServiceNow Studio supports the collaboration plugin for properly licensed customers.

Note:

1. You must have an App Engine Enterprise license to take full advantage of collaboration.
2. If you already have the collaboration plugin installed, you can continue to use collaboration.
3. Customers that don't have Collaboration installed will not be able to manage delegated development permissions in ServiceNow Studio. Existing delegated development permissions will still be respected within ServiceNow Studio.

Apps that you can access

The two ServiceNow Studio roles that can access ServiceNow Studio, admin and delegated_developer, have different access to apps:

- Users with the admin role automatically have access to all apps in ServiceNow Studio.
- Users with the delegated_developer role have access to:
  - Apps they create
  - Apps they've been invited to edit (as an editor on the app)
  - All apps within the scope you have access to. For more information on scopes, see Application scope.

If you know of an app that you want to work on but don't see it in ServiceNow Studio, contact your admin and ask that they give you permission to work on the app using the Collaboration app. The ServiceNow AI Platform Collaboration app is automatically installed with ServiceNow Studio. For more information about the Collaboration app, see Application collaboration.

What app owners and editors can do

There are two default collaborator roles for working on apps with other developers: owner and editor.

- If you create an app, you're the owner of that app.
- If you see an app in ServiceNow Studio that you've been delegated to work on, you can open it and begin working on it with whatever collaboration role the owner assigned you. That role is usually editor.

Default collaboration descriptors

Descriptor	Description
Owner	Owner of the application. • If you created the app, you're automatically the owner. • Owners can manage other collaborators for the app. • Owners can delete apps because they have the delete app permission. • Owners automatically get the delegated_developer role for the app.
Editor	• Editors can invite collaborators. • Editors have a more limited ability to edit the app.

Note:
For the full list of default owner and editor collaborator collaboration type permissions, see Collaboration permissions for ServiceNow Studio.

Custom collaboration descriptors and permissions

The collaboration descriptor that someone is assigned determines if they can assign, manage, and monitor delegated development permissions. For example, people who are owners can do more than people with the editor collaboration descriptor.

If you want to create a customized collaboration role for help building your app, you can create a custom collaboration descriptor, which is a customized collaboration role. You then use collaboration permissions to control what developers (or users who deploy applications) can do in the app.

If needed, admins can define custom collaboration descriptors to select when managing collaborators using the Collaboration app. For more information on custom descriptors, see Create collaboration descriptors to assign permissions.

Managing collaboration permissions for other developers

If you invite someone to collaborate on an app and they don't have the Delegated developer (delegated_developer) role, an App Engine admin must approve the collaboration request. For more information, see Delegated development and deployment.

When you add a user or group to collaborate on an app, a collaboration task is generated behind the scenes, which initiates an approval flow. If you have App Engine Management Center (AEMC) installed, your admin can review and approve/deny these collaboration request tasks there. The collaboration task that goes to your admin provides information on which app a developer is being added to, and what permissions they get. Admins and approvers sometimes need to review these task records before they add developers to the application.

If you don't have AEMC installed, admins can navigate to **All** > **App Engine** > **Collaboration** > **Collaboration Tasks**.

View collaborators on an app in ServiceNow Studio

View the collaborators for an app to see who is co-developing the app in ServiceNow Studio.

Before you begin

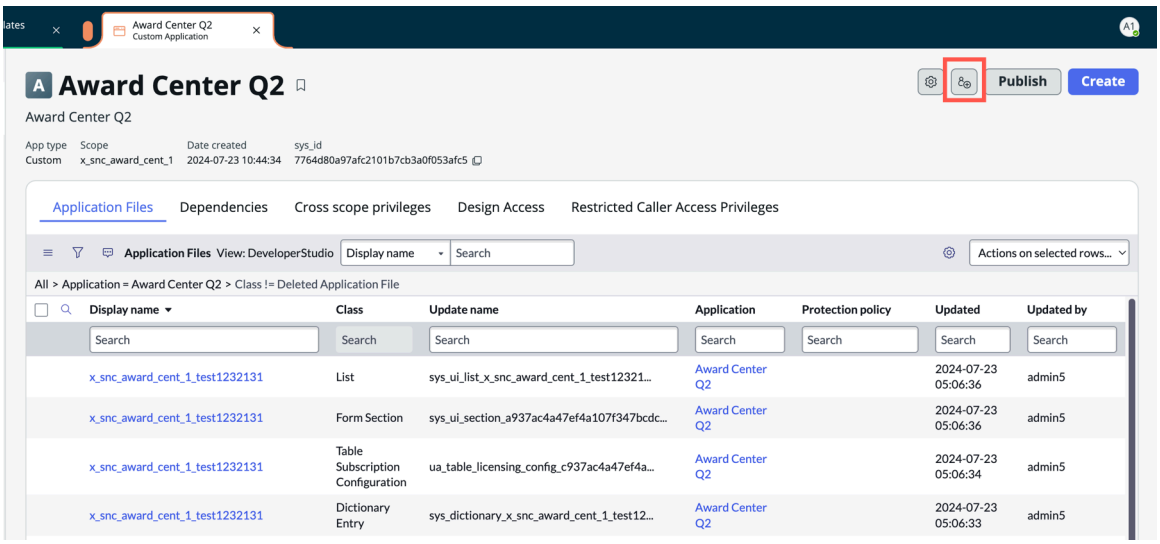
Customers that don't have Collaboration installed will not be able to manage delegated development permissions in ServiceNow Studio. Existing delegated development permissions will still be respected within ServiceNow Studio.

Role required: admin or delegated developer

Procedure

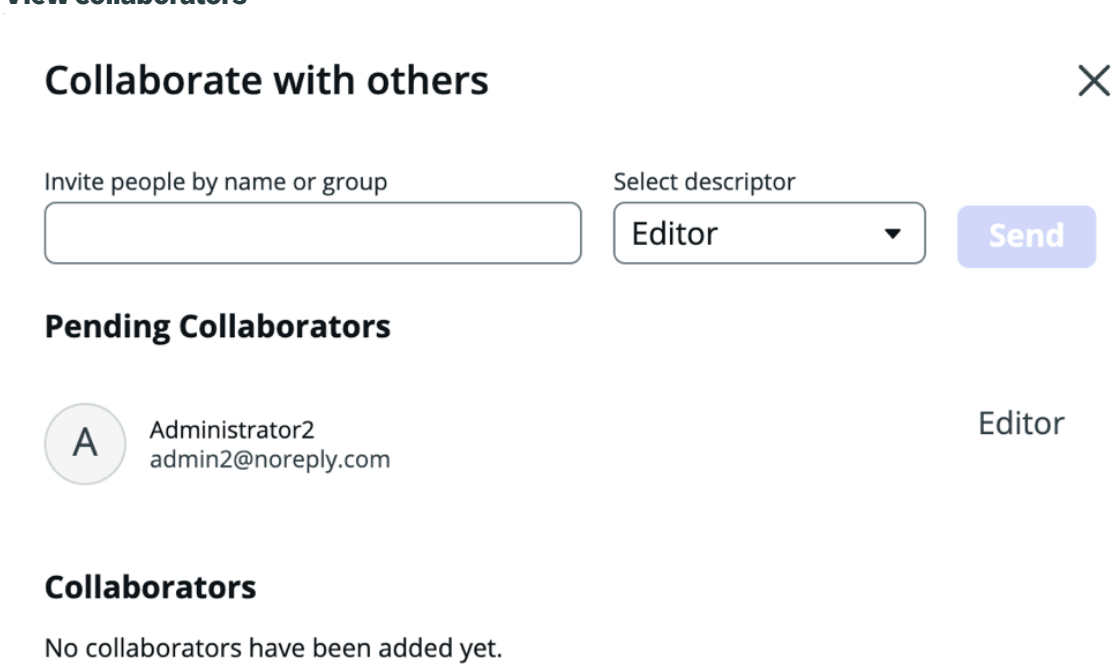
1. Navigate to **All** > *App Engine* > *ServiceNow Studio*.
2. Find and select the app that you want to view collaborators for.
3. Access collaboration settings by selecting the collaborators icon ( ).

Access collaboration in ServiceNow Studio



4. Review the collaborators' names, groups, and descriptors on the Collaborate with others modal.

View collaborators



Result
You can take additional actions when viewing collaborators. For more information, see the following topics:

- Add collaborators to an app in ServiceNow Studio
- Modify or customize collaboration permissions for a user or group in ServiceNow Studio
- Remove collaborators from an app in ServiceNow Studio

Add collaborators to an app in ServiceNow Studio

Invite other people to work on an app with you in ServiceNow Studio, collaborating as co-developers.

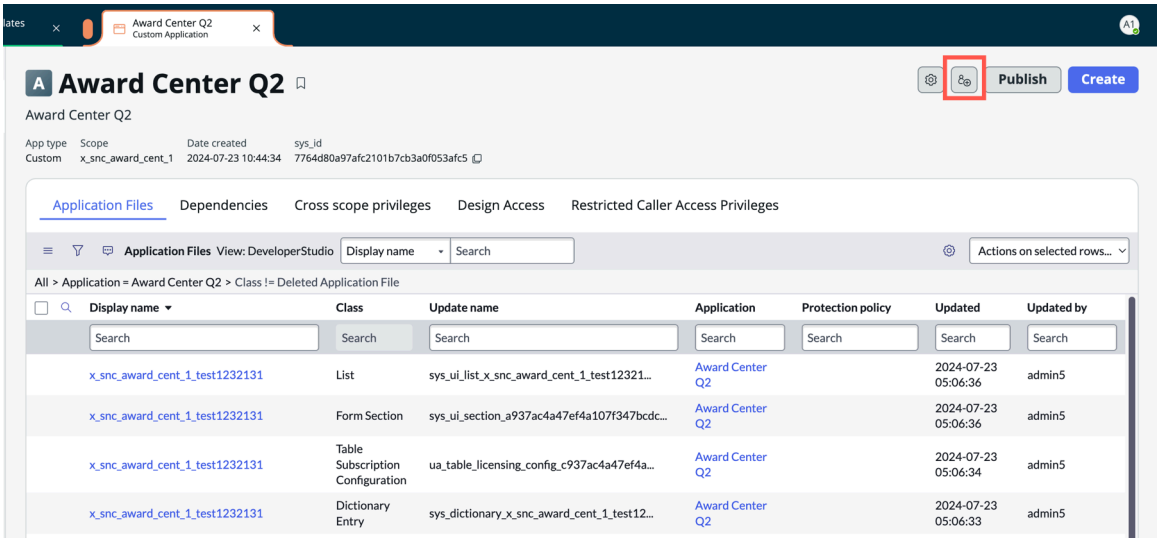
Before you begin
Customers that don't have Collaboration installed will not be able to manage delegated development permissions in ServiceNow Studio. Existing delegated development permissions will still be respected within ServiceNow Studio.

Role required: admin or delegated developer

Procedure

1. Navigate to **All** > *App Engine* > *ServiceNow Studio*.
2. Find and select the app you want to invite people to collaborate on.
3. Open collaboration settings for the app by selecting the collaborators icon ( ).

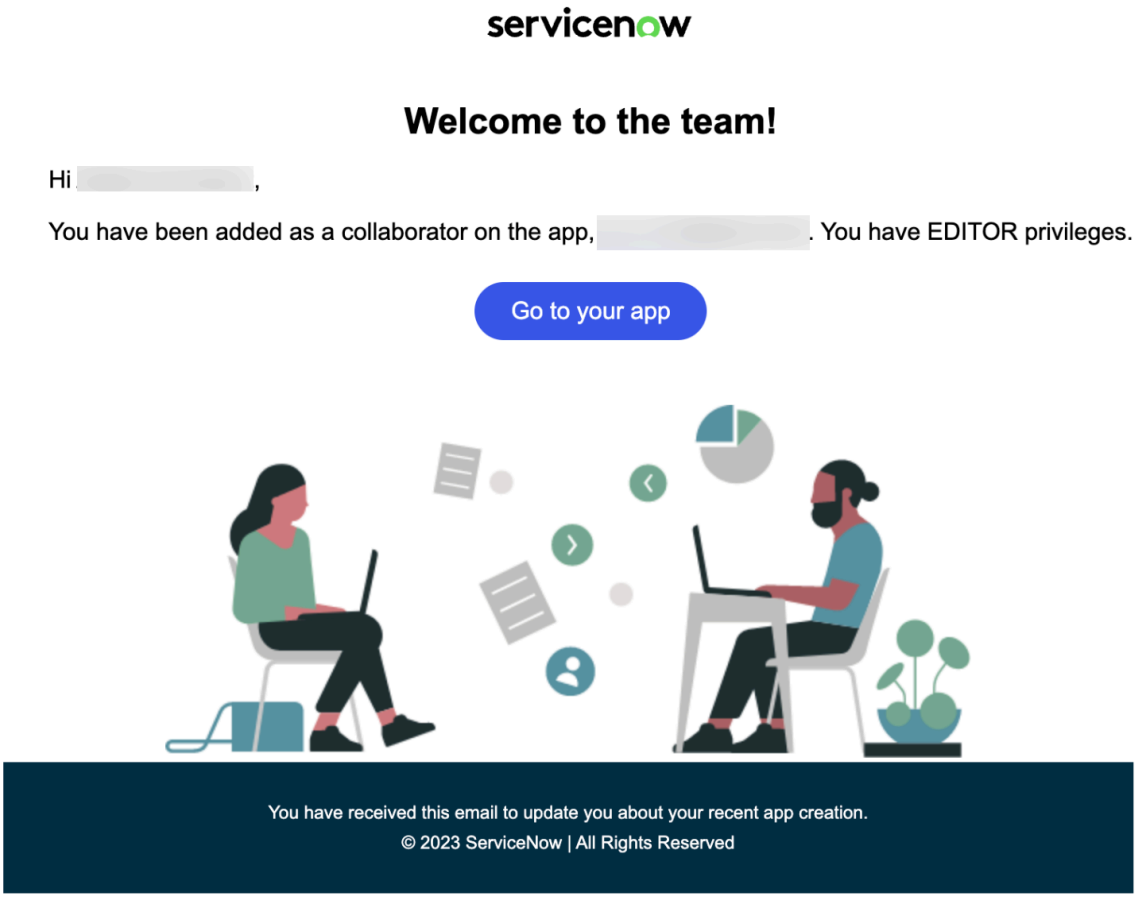
Access collaboration in ServiceNow Studio



4. Search for the user or group that you want to invite to work on the app by entering the name in the **Invite people by name or group** field.
   A drop-down list displays the matching people and groups. If a user or group appears in the drop-down list but you can't select it, it's already been added as a collaborator and can't be re-selected.
5. Select the collaboration role for the user or group that you're adding from the **Select descriptor** field.
   If you're an editor for the app, you can select only the editor descriptor.
   - For more information on collaboration descriptors, see Collaborating on apps using ServiceNow Studio.
   - For a list of all the collaboration permissions, see Collaboration permissions for ServiceNow Studio.
6. Finish inviting the collaborator by selecting **Send**.

Result

- If the user has ServiceNow Studio or delegated developer permissions and is new to the ServiceNow AI Platform, they must be approved by an admin. After the request is approved, both the requester and the user receive an email indicating that the user has been added to the application.



- If the user has ServiceNow Studio or delegated developer permissions and is not new to the ServiceNow AI Platform, the collaboration request is auto-approved. Both the requester and the user receive an email indicating that the user has been added to the application.

Modify or customize collaboration permissions for a user or group in ServiceNow Studio

Change the collaboration access that a user or group has to work on an app in ServiceNow Studio by modifying or customizing their collaboration descriptor.

Before you begin

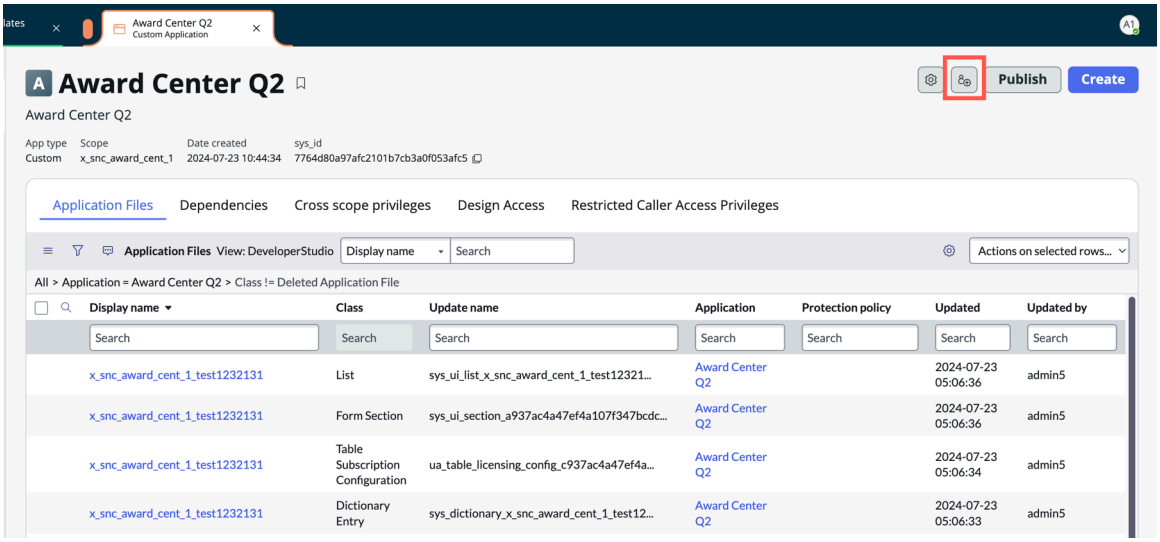
Customers that don't have Collaboration installed will not be able to manage delegated development permissions in ServiceNow Studio. Existing delegated development permissions will still be respected within ServiceNow Studio.

Role required: admin or delegated developer

Procedure

1. Navigate to **All** > *App Engine* > *ServiceNow Studio*.
2. Find and select the app where you want to modify permissions for collaborators.
3. Access collaboration settings by selecting the collaborators icon ( ).

Access collaboration in ServiceNow Studio



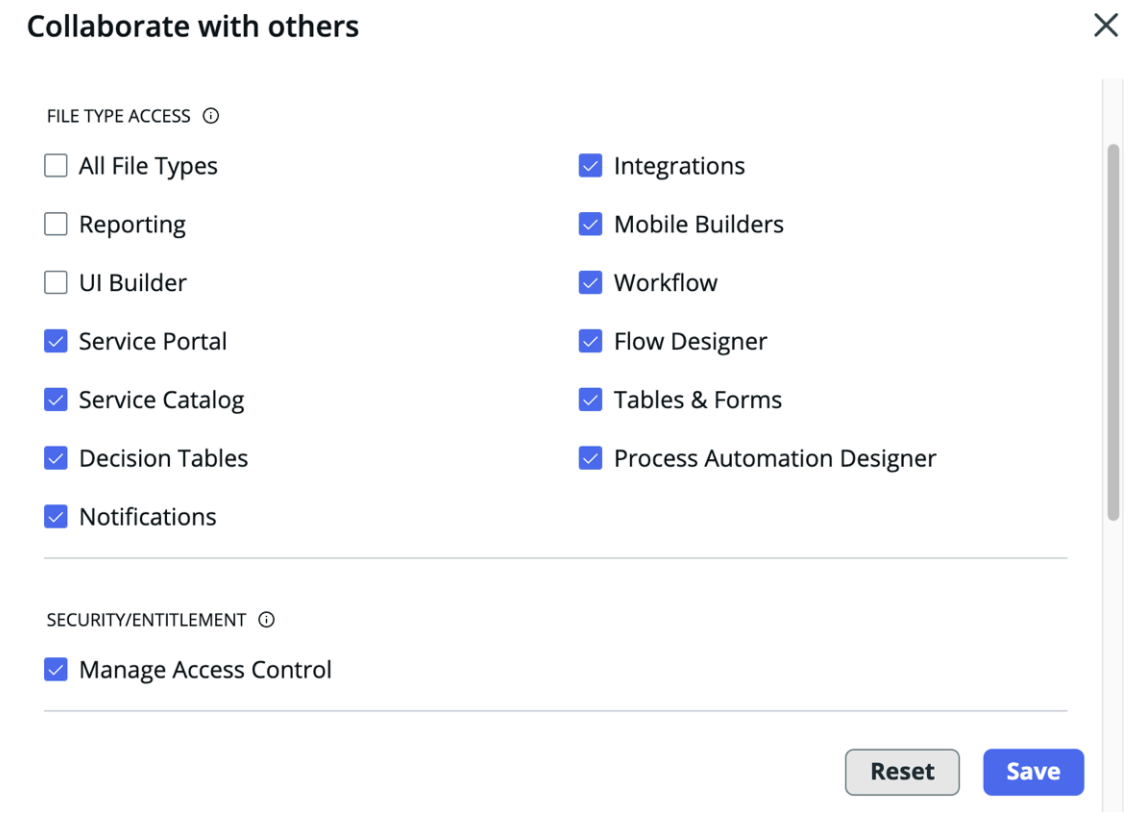
4. Choose the new permission level for the user or group in the Collaborators section of the modal.
5. **Optional:** Customize what the collaborator can do by creating custom permissions.

   a. Select **Customize permissions** for the user or group in the Collaborators section.

Customize collaboration permissions

   b. Choose the permissions you want the group or user to have.
   For more information on permissions and descriptions, see Collaboration permissions for ServiceNow Studio.

Customize collaboration permissions



c. Select **Save**.

Result
Your changes are automatically saved when you close the Collaborate with others modal.

What to do next
Your App Engine admin must then approve the changes to collaborators. For more information, admins should see Approve a collaboration request.

Remove collaborators from an app in ServiceNow Studio

Restrict a user or group from working on an app in ServiceNow Studio by removing them as a collaborator.

Before you begin
Customers that don't have Collaboration installed will not be able to manage delegated development permissions in ServiceNow Studio. Existing delegated development permissions will still be respected within ServiceNow Studio.

Role required: admin or delegated developer

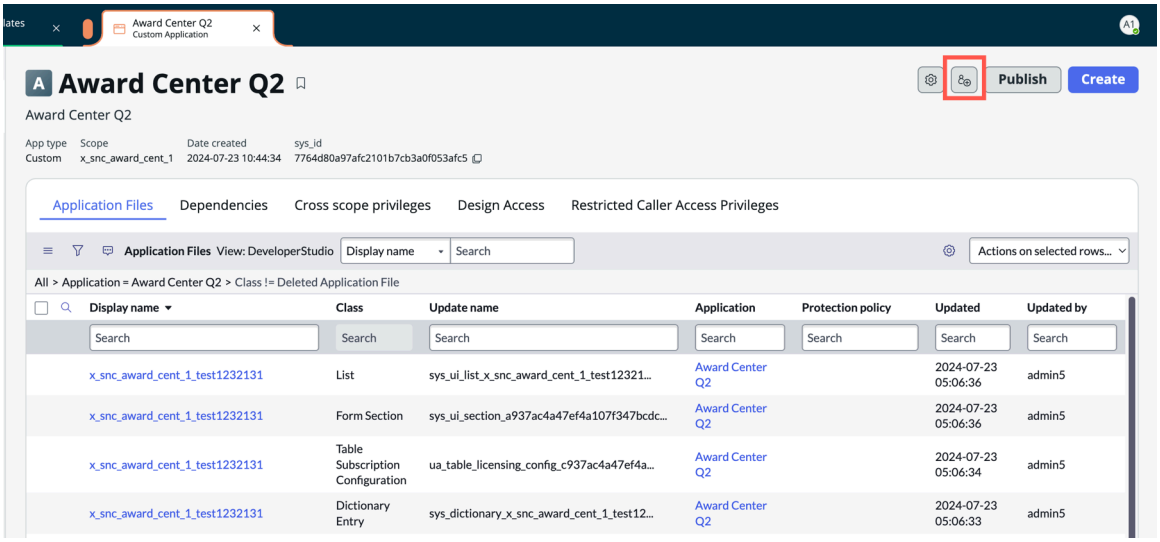
Procedure

1. Navigate to **All** > *App Engine* > *ServiceNow Studio*.

2. Find and select the app you want to remove collaborators from.

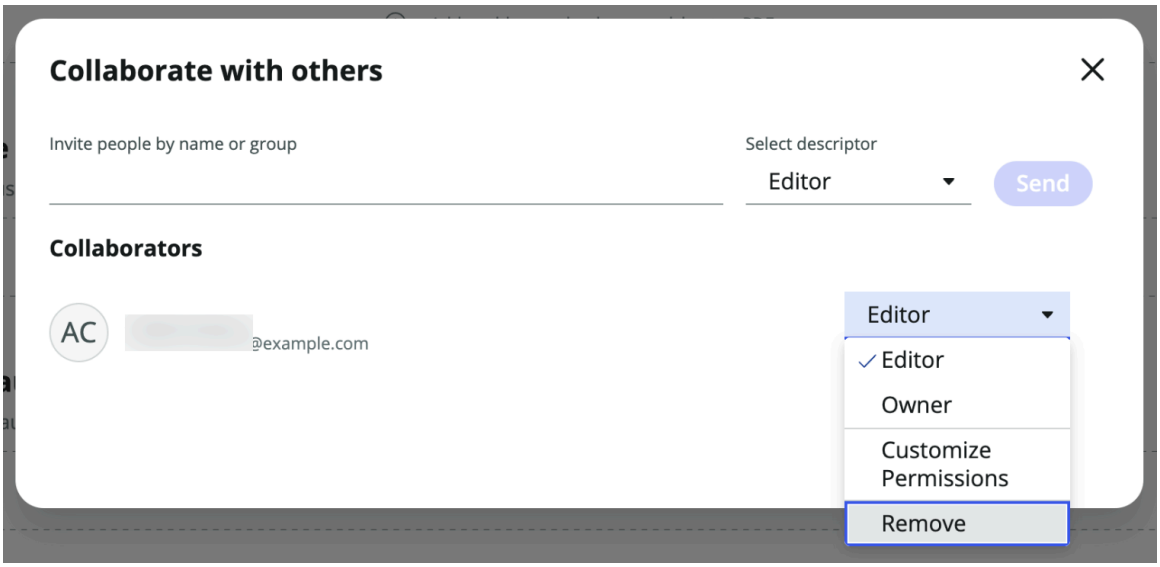
3. Access collaboration settings by selecting the collaborators icon ( ).

Access collaboration in ServiceNow Studio



4. Select **Remove** for the user or group in the Collaborators section of the modal.

Remove a collaborator



Result
Your changes are automatically saved when you close the Collaborate with others modal.

Using ServiceNow Studio

Accelerate your app development with ServiceNow Studio. Many ServiceNow AI Platform features are integrated with a suite of development tools and builders to provide an efficient and powerful app development and deployment experience.

Aims of ServiceNow Studio

ServiceNow Studio improves upon a lot of great features in the legacy Studio product, as well as integrates new features that can speed up your app development experience. ServiceNow Studio helps enable your app development experience by making it faster and easier to do the following:

- Access, switch between, and adopt the right tools you need for development.
- Find, organize, and edit your work in apps and metadata records.
- Track, package, and deploy your work in applications through update sets, pipelines, or the Application Repository.

Create an application in ServiceNow Studio

Create a custom application in ServiceNow Studio. After creating the foundations of your app, you can add data, automations, or many other types of app files using integrated development tools and builders in ServiceNow Studio.

Before you begin

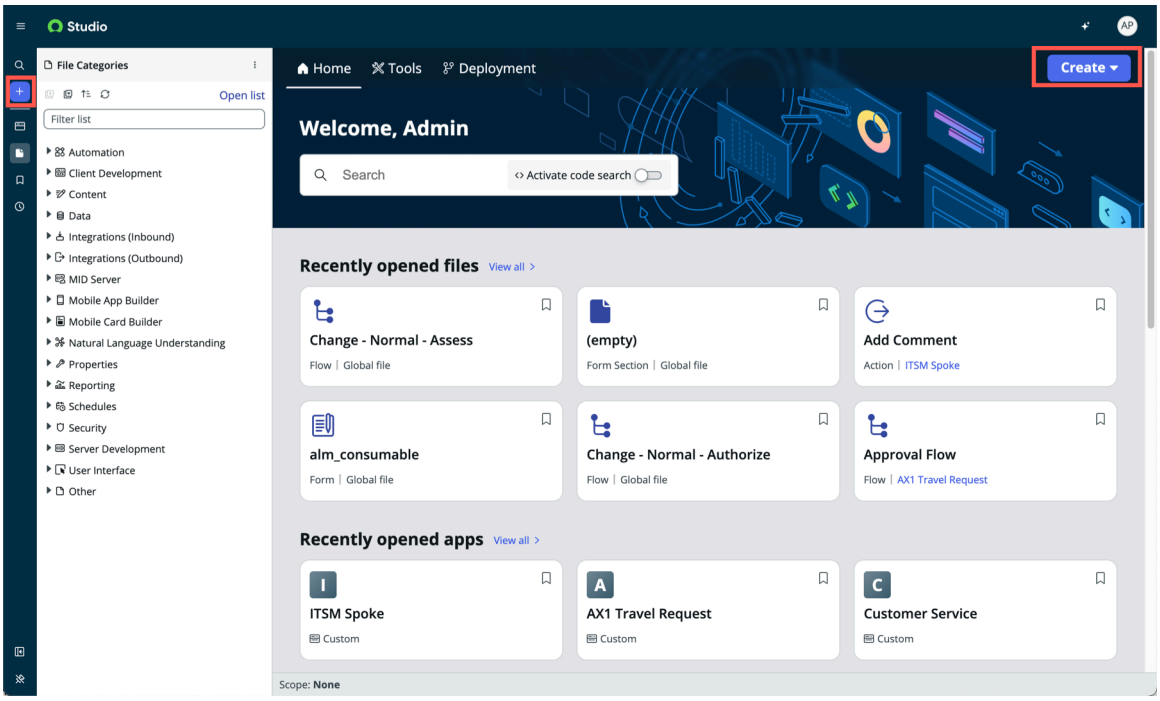
Only admins can create an app in ServiceNow Studio. If you need an app created for you, contact your admin.

Role required: admin

Procedure

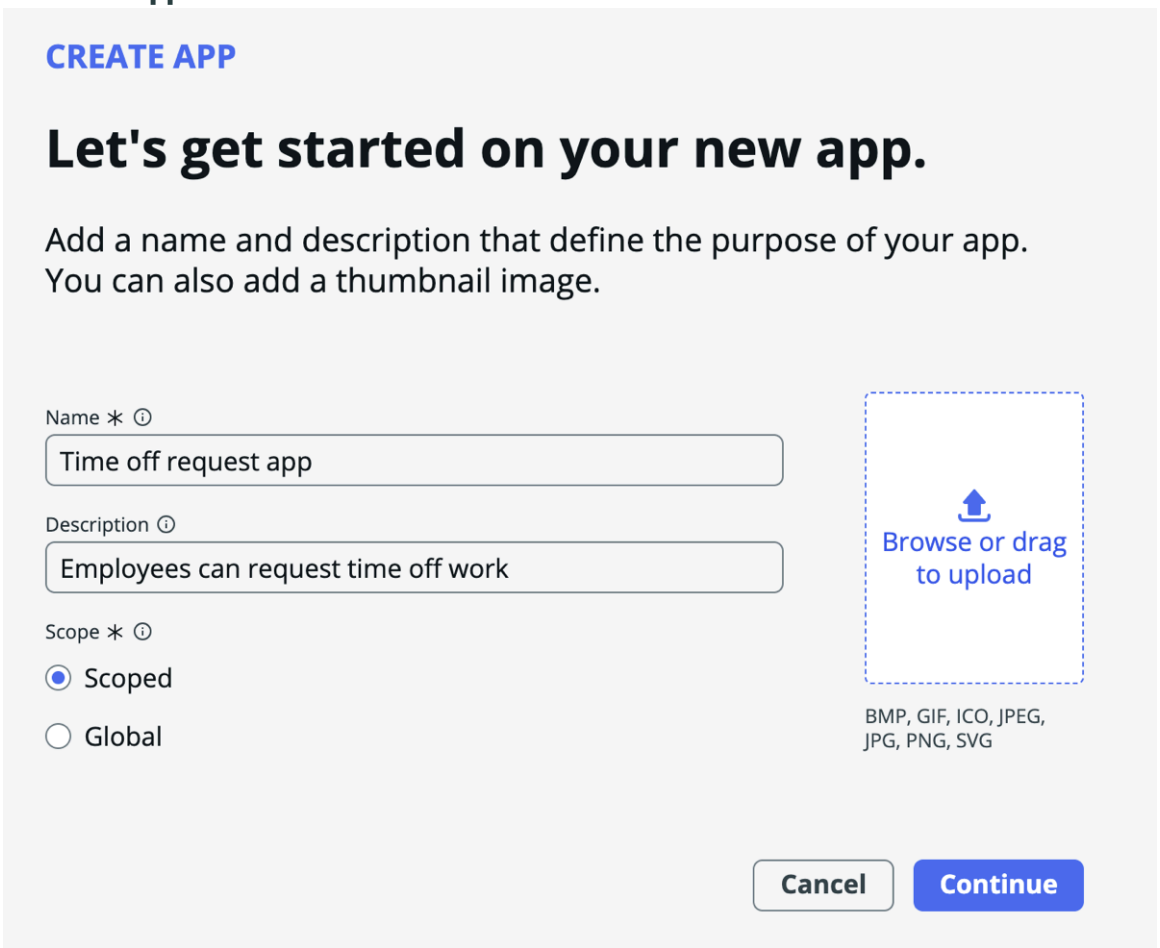
1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Select either the Create icon ( ) next to the Navigator panel or the **Create** button.

Select either Create button



3. Select **App** from the options that appear.
4. Enter the basic information for the app on the form.

Create an app basic details



a. Enter a **Name** and **Description** for your application.

b. **Optional:** Add a logo by either dragging the image to the **Browse or drag to upload** field or selecting the field, selecting the image from your file directory, and selecting **Open**.

c. Specify the **Scope**.

- **Scoped** means that the app doesn't interact with any other data on the instance. By default, the app can access and change its own tables and business logic, but other apps can't unless you give them explicit permission.
- **Global** means that all tables and business logic on the instance can interact with the app.

For more information about working with scopes, see Application scope.

d. Select **Continue**.

5. Define user access to the app by adding roles.
ServiceNow Studio automatically defines default admin and user roles. You can remove the pre-defined roles or add more roles. For more information about roles, see Managing roles.

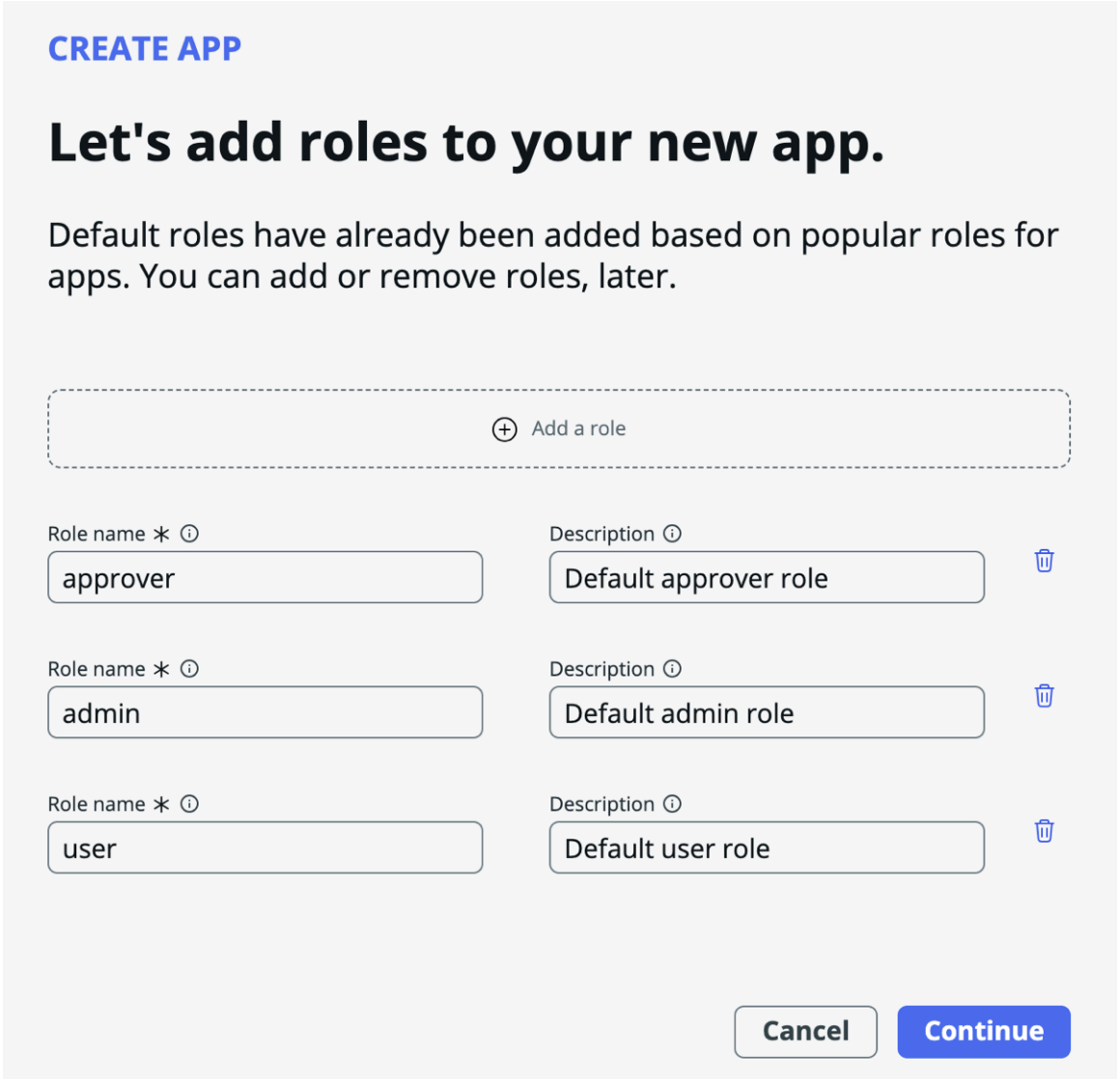
a. Select **Add a role**.

b. Enter the **Role name** and **Description**.

c. Repeat this process for as many roles that you need to add.

d. Select **Continue**.

Roles create user access to the app



6. Open the app dashboard when ServiceNow Studio is done creating the foundation of the app by selecting **Go to app dashboard**.

Result

The app dashboard opens within ServiceNow Studio. From there, you can add other content, such as application files, dependencies, and cross-scope privileges. For more information, see Create an app file in ServiceNow Studio, Working with metadata app file categories in the ServiceNow Studio Navigator, and Access integrated development tools and builders in ServiceNow Studio.

Create an app file in ServiceNow Studio

Create an app file to define an aspect of how an application functions. For example, you could add a security file using ServiceNow Studio to define which users can access the app.

Before you begin

Role required: admin or delegated developer

About this task

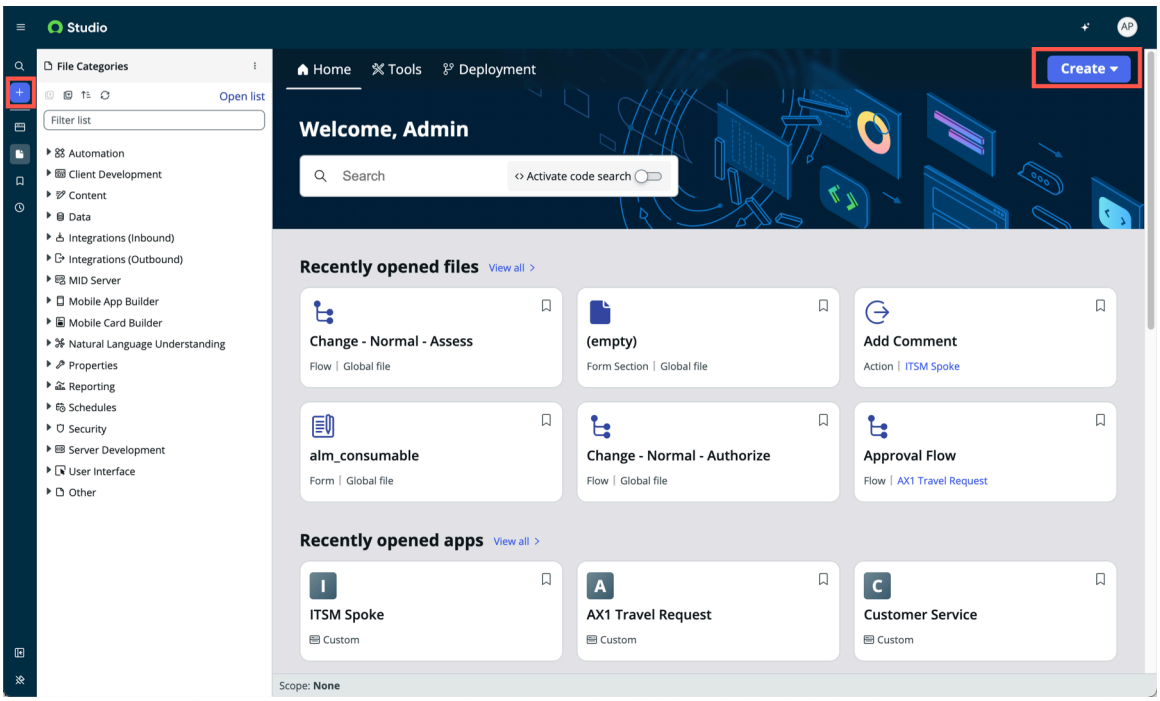
Application files are essentially metadata records for application logic such as business rules, workflows, and script includes. For more information on app files, see Application files and Working with metadata app file categories in the ServiceNow Studio Navigator.

Delegated developers have access to create app files for apps they have access to. If you need an app created, contact your admin.

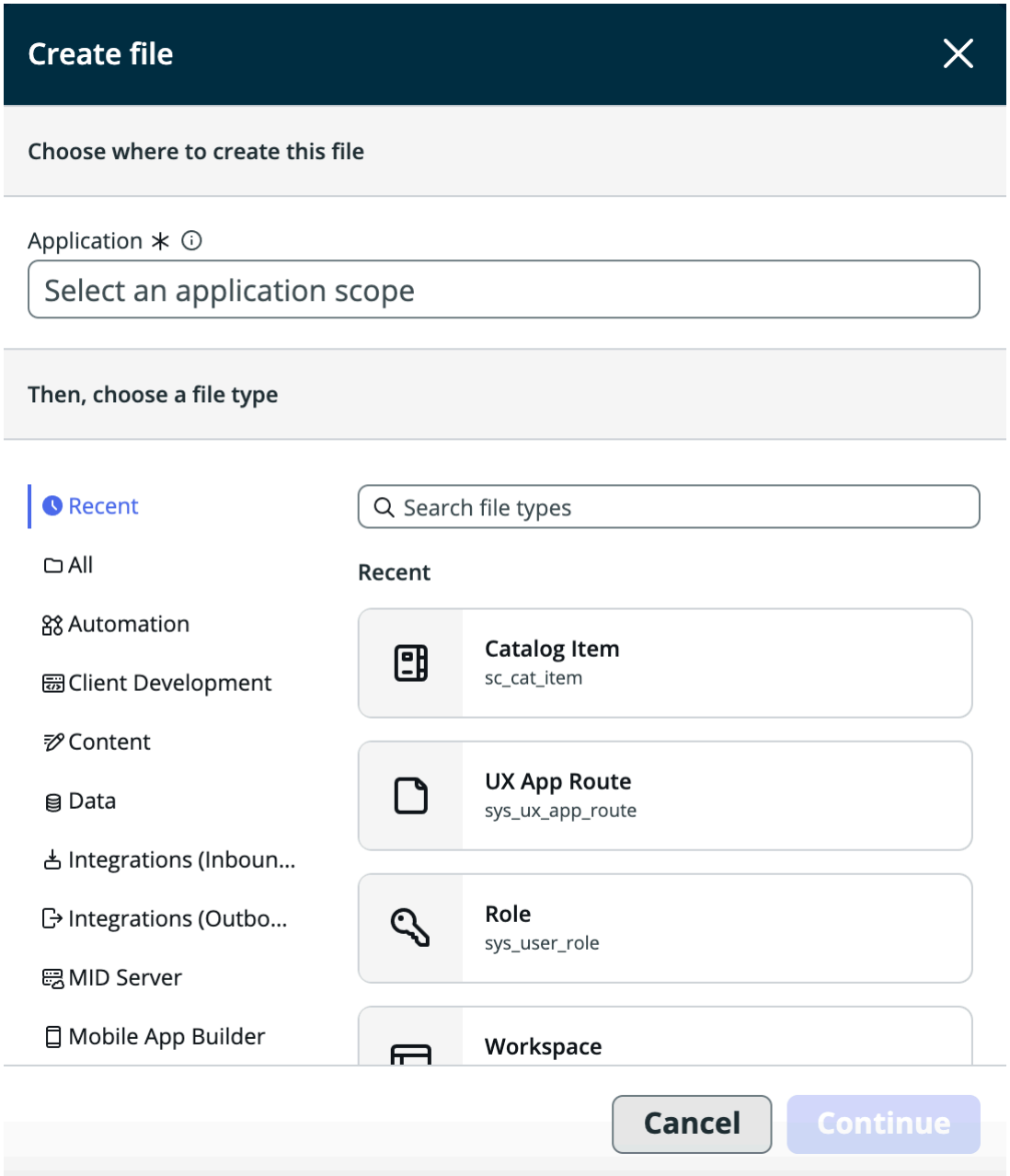
Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Select either the Create icon ( ) next to the Navigator panel or the **Create** button.

Select either Create button



3. Select **File** from the options that appear.



4. Specify which app the file belongs to by entering it in the **Application** field.
   If you want the app file to be available to all apps, select the **Global** scope.
5. Select the card for the type of file you want to create.
   You can find the file type in several ways:
   - Filter the available types by selecting one of the metadata taxonomies in the left of the modal.
   - Enter the type of file you're looking for in the search bar.
   - Choose from the list of Recent file types.

   For a list of all types of metadata that you can use to create app files, see ServiceNow Studio Navigator panel taxonomy. For more information about each file type, see Working with metadata app file categories in the ServiceNow Studio Navigator.
6. Select **Continue**.
   The record for the app file appears in a new browser tab in ServiceNow Studio.
7. Fill in the form with the details for the new app file.

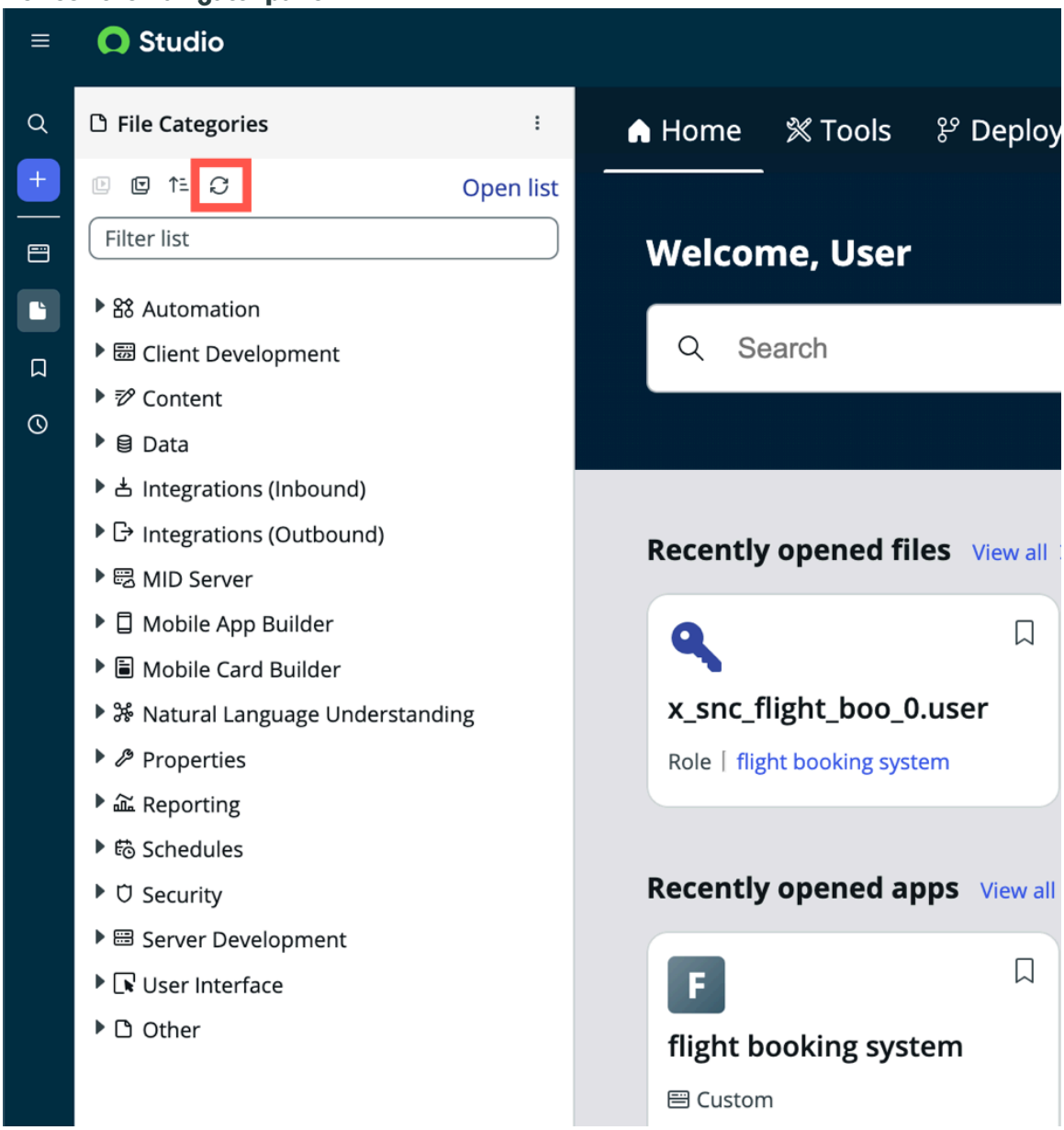
The form fields vary depending on the type of file you created.

8. Select **Submit**.

What to do next

After you finish creating the app file, you must select the refresh icon in the Navigator panel for it to appear in the list of app files.

Refresh the Navigator panel



Now Assist for app generation in ServiceNow Studio

Use the ServiceNow® Now Assist for Creator application to use generative AI for simplifying app creation. After having a conversation about the app that you want to create, Now Assist for Creator generates the app that you can then modify in ServiceNow Studio (Xanadu Store release 2).

Get started

Explore  Learn how to create apps through conversation	Configure  Set up app generation
Generate  Build apps through conversation	Reference  Get details about properties, roles, and more

Important:

- Not all model providers are available for customers with in-country SKUs, and some Now Assist products/features are currently unavailable for in-country customers. For more information, see the KB1584492 article in the Now Support Knowledge Base. Be sure to check for model provider availability updates in future releases.
- Some Now Assist products/features are currently unavailable for customers in the FedRAMP, NSC DOD IL5, or Australia IRAP-Protected data centers, self-hosted customers, or in other restricted environments. For more information, see the KB0743854 article in the Now Support Knowledge Base . Be sure to check for availability updates in future releases.
- Some Now Assist products/features are currently available only for customers in some regions. Be sure to check for availability updates in future releases.
- Some AI products and skills are not available in Regulated Markets. For more information, see KB2593939: Regulated Markets AI Products/Skills Not Available. Be sure to check for availability updates in future releases.

Troubleshoot and get help

AI limitations

This application uses artificial intelligence (AI) and machine learning, which are rapidly evolving fields of study that generate predictions based on patterns in data. As a result, this application may not always produce accurate, complete, or appropriate information. Further, there is no guarantee that this application has been fully trained or tested for your use case. To mitigate these issues, it is your responsibility to test and evaluate your use of this application for accuracy, harm, and appropriateness for your use case, employ human oversight of output, and refrain from relying solely on AI-generated outputs for decision-making purposes. This is especially important if you choose to deploy this application in areas with consequential impacts such as healthcare, finance, legal, employment, security, or infrastructure. You agree to abide by ServiceNow's AI Acceptable Use Policy, which may be updated by ServiceNow.

Data processing

This application requires data to be transferred from ServiceNow customers' individual instances to a centralized ServiceNow environment, which may be located in a different data center region from the one where your instance is, and potentially to a third-party cloud provider, such as Microsoft Azure. This data is handled per ServiceNow's internal policies and procedures, including our policies available through our CORE Compliance Portal.

Data collection

ServiceNow collects and uses the inputs, outputs, and edits to outputs of this application to develop and improve ServiceNow technologies including ServiceNow models and AI products. In addition, this application will collect information about applications (and associated application files) in which App generation was utilized. Customers can opt out of future data collection at any time, as described in the Now Assist Opt-Out page.

Exploring app generation in ServiceNow Studio

App generation overview

Administrators and developers must be assigned the admin and now.assist.creator roles to create and edit apps using app generation. In Xanadu Store Release 2 there is a second option to assign users the delegated developer, now_assist_panel_user, and now.assist.creator roles to edit apps (not create) using app generation. For more information about using app generation, see Generate apps with Now Assist for Creator for ServiceNow Studio

Conversation

Chat with Now Assist for Creator to specify the business processes you want in the application including the details on the objectives, users, workflows, and experiences.

Refinement in Xanadu Store Release 2

Now Assist for Creator provides a summary of the application requirements based on the information collected during the conversation. Preview the application and check that it's accurate. If you want to make changes, stay in the conversation and keep editing. Now Assist continues to update the preview tab based on your comments and provides summaries until you choose to proceed with generating the application.

Generation

Now Assist for Creator generates the application and associated components, including the tables, roles, access control lists (ACLs), and record producers.

Open and verify everything that is generated. In Xanadu Store Release 2, verify in ServiceNow Studio. Modify the app to make it suit your organization's needs. For example, the app's functionality can be extended by adding flows and automation, script includes, business rules, and other features.

App generation benefits

Benefit	Feature	Users
Quickly reach a starting point and establish a base for further development.	Generate apps with Now Assist for Creator for ServiceNow Studio	Developer

General guidelines for using app generation in ServiceNow Studio

Learn about general guidelines for using app generation.

Configuring app generation in ServiceNow Studio

Install Now Assist for app generation in ServiceNow Studio

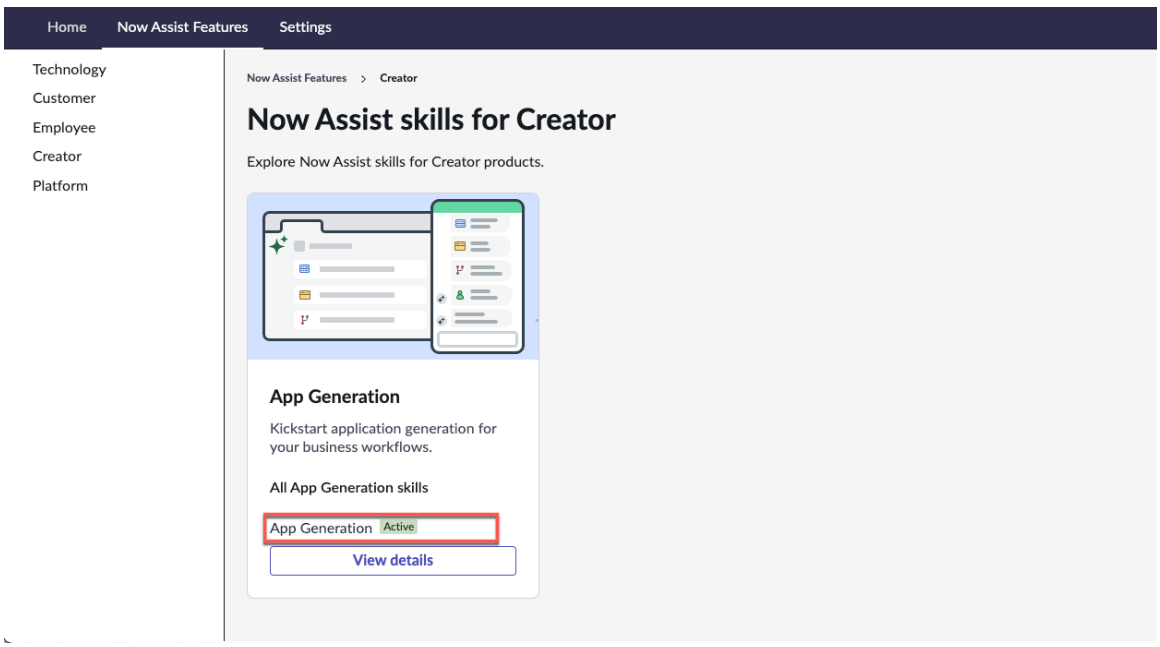
Before you begin

Review the Now Assist for Creator application listing in the ServiceNow Store to learn about dependencies, licensing, and subscription requirements, and release compatibility. Now Assist for Creator installs the Now Assist for app generation application.

Role required: admin

Procedure

1. Navigate to the Now Assist for Creator application on the ServiceNow Store .
2. On the Now Assist for Creator application page, select **Request App**.
3. After your request is approved, navigate to **All** > **System Applications** > **All Available Applications** > **All**.
4. Find the Now Assist for Creator application (sn_now_creator) by using the filter criteria and search bar.
5. Select **Install**.
6. Verify that Now Assist for Creator is installed.
   a. Navigate to **All** > *Now Assist Admin* > **Features**.
   b. In the workflow list, select **Creator**.
   c. On the App Generation card, verify that the app generation skill is active.



For more information about using the Now Assist Admin console to access information about setting up, configuring, and monitoring Now Assist applications, see Now Assist Admin console .

What to do next

Grant the admin and now.assist.creator roles to each user that you want to create and edit applications using app generation.

As of Xanadu Store Release 2, users that only need to edit (not create) applications using app generation can be granted the delegated_developer, now_assist_panel_user, and now.assist.creator roles. For more information, see Delegated development and deployment.

To begin generating applications, see Generate apps with Now Assist for Creator for ServiceNow Studio.

Related topics

Install Now Assist for Creator

Generate apps with Now Assist for Creator for ServiceNow Studio

Have a conversation with the Now Assist for Creator application to start building applications.

This video shows you how to perform the following procedure.

https://player.vimeo.com/video/1040832044?h=b2b8dc6b7c&badge=0&autopause=0&player_id=0&app_id=58479

Before you begin

To use app generation, you must activate the skill in Now Assist for Creator. For more information, see Install Now Assist for app generation in ServiceNow Studio.

Role required: admin and now.assist.creator

Note:
As of Xanadu Store Release 2, if you have users that only need to edit (not create) apps, they can be assigned the delegated_developer, now_assist_panel_user, and now.assist.creator roles.

About this task

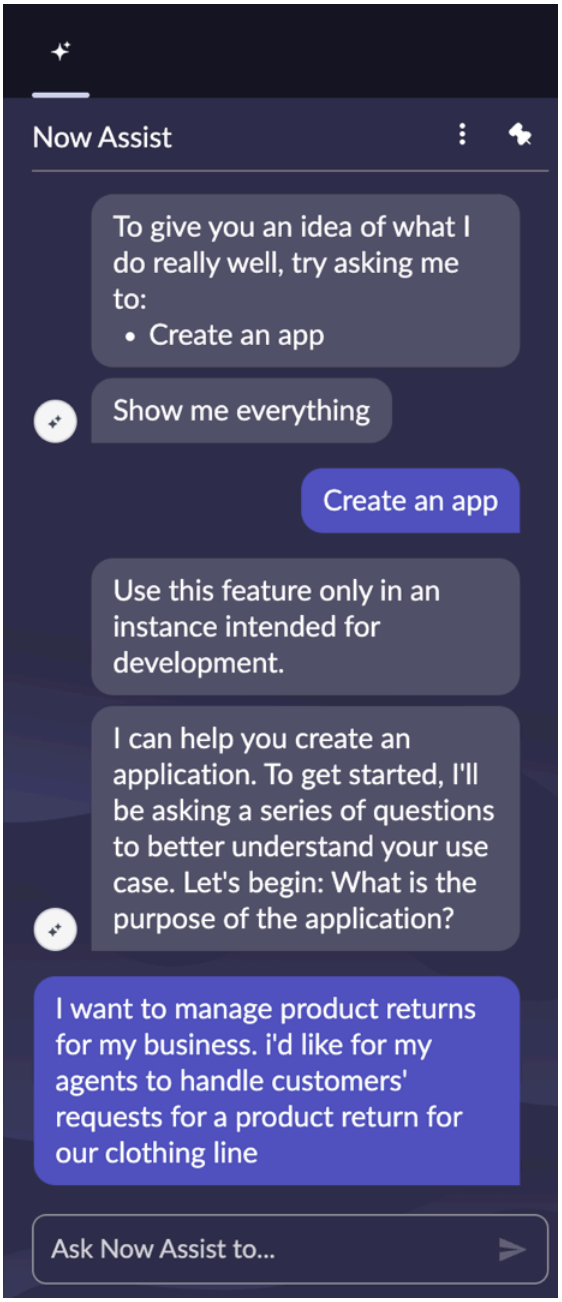
When app generation is enabled on an instance, the Now Assist icon (✨) appears on the home page banner.

Procedure

1. Open the Now Assist panel by navigating to **App Engine** > **ServiceNow Studio** and selecting the Now Assist icon.

2. In the Now Assist panel, select **Create an app**.
   Now Assist asks you to describe your application.
3. Summarize your current business process.

   Describe your application's use case so Now Assist understands the purpose of your application. You can be highly detailed if you have a clear idea of what you want. If you don't, start with a broad description. Now Assist narrows down your app's requirements.



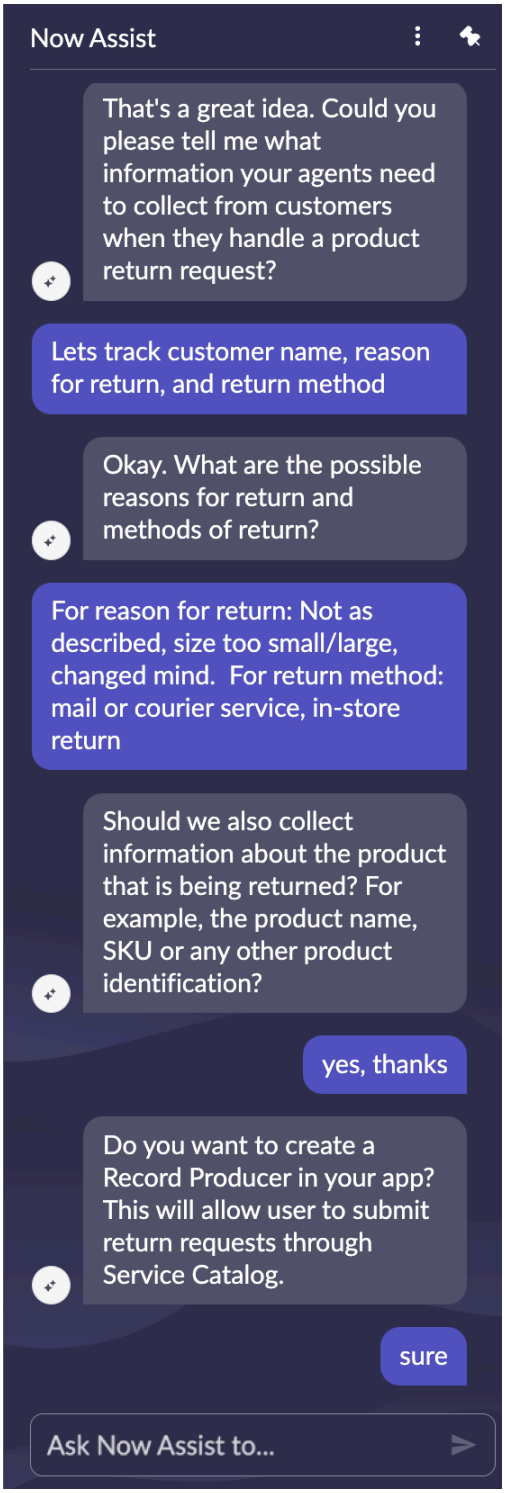
4. Continue the conversation with Now Assist by providing details about how you want the app to work.

   Example

   Now Assist asks you questions to understand the data to be collected, the users involved and their permissions, and the desired interface. Your answers help Now Assist create the correct tables, roles, access control lists (ACLs), forms, and record producers for your application.

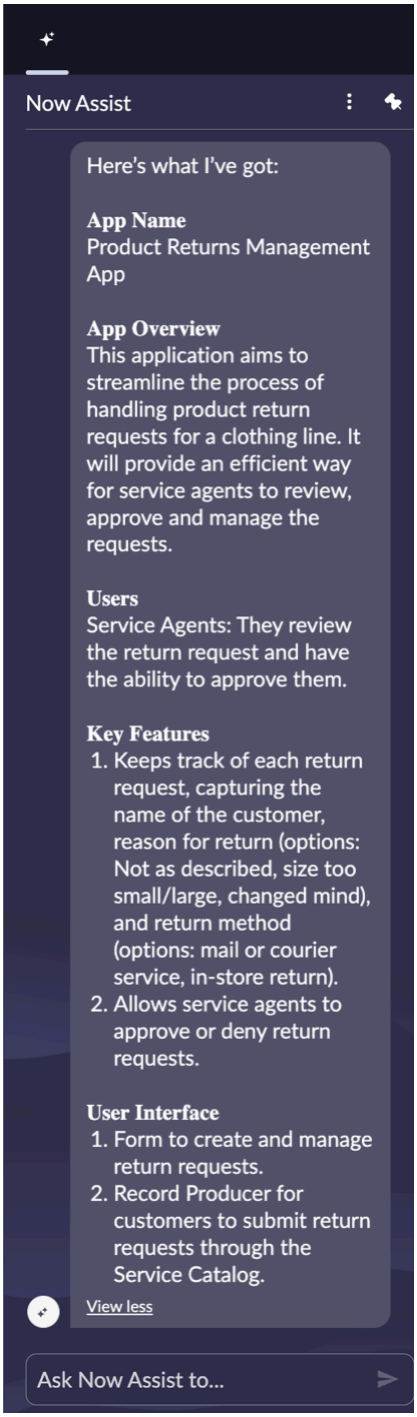
   If you know exactly how you want your app to work, be specific about its functionality. If you don't, interact with Now Assist to determine the correct application requirements. For more information, see General guidelines for using app generation in ServiceNow Studio.

   When Now Assist collects enough information, it provides a detailed summary of the application requirements.



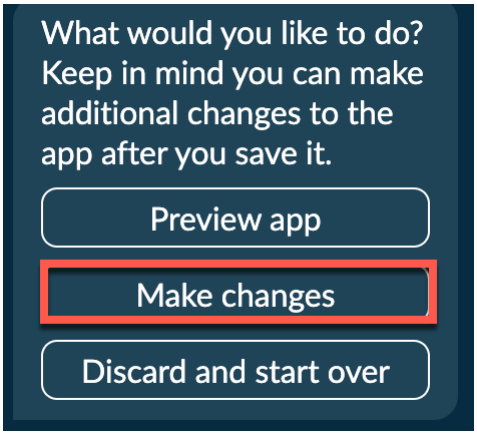
5. Review the summary of your application requirements.

Note:
Because the information is generated by AI, review the text to make sure it's accurate.



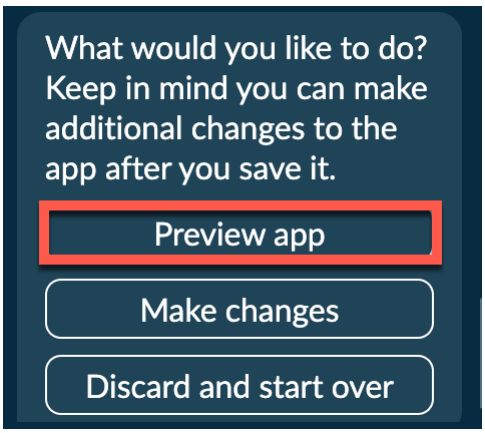
6. In Xanadu Store Release 2, preview, finalize, and then generate the app.

a. If necessary, select **Make changes** to continue editing.



Continue to interact with Now Assist until the summary it produces matches your app's requirements.

b. After you're satisfied with the application requirements, select **Preview app**.



The preview opens and displays a list of app files. Select any of the app files to view details. For example, select a role to see if it grants users create, read, write, or delete permissions. If your app contains a record producer, select it to see what the form and fields look like.

Note:

If the Now Assist panel is covering information, select the Now Assist icon to close the panel. Select the icon again to open the panel and continue the conversation.

When you are finished previewing, select an option:

- **Save and open app** generates the app and opens it in ServiceNow Studio for you to review and edit as needed.
- **Make changes** enables you to continue the conversation with Now Assist to refine and edit the app. After Now Assist makes any changes you request, the app preview is also updated.
- **Discard and start over** deletes the current app and resets the conversation in the Now Assist panel.

For more information about ServiceNow Studio, see Building applications with ServiceNow Studio.

Result
In Xanadu Store Release 2, use the tools in ServiceNow Studio to add more features and enhance your app further.

Review the apps generated by Now Assist for Creator in ServiceNow Studio

In Xanadu Store Release 2, use the ServiceNow Studio development environment to verify and modify the application that you created with Now Assist for Creator.

This video shows you how to perform the following procedure.

https://player.vimeo.com/video/1040832738?h=0b12434687&badge=0&autopause=0&player_id=0&app_id=58479

Before you begin
You must have created an application using Now Assist for Creator.

Role required: admin and now.assist.creator, or sn_g_app_creator.app_creator and now.assist.creator (If you have users that only need to edit, not create, apps, they can be assigned the delegated_developer, now_assist_panel_user, and now.assist.creator roles.)

About this task
The application generated by Now Assist for Creator provides a starting point for further development. Examine the contents of the application and confirm that they align with your business process. Enhance the application by adding functionality as needed.

Procedure
1. If the application isn't already open, navigate to App Engine > ServiceNow Studio and select the application tile.

   Notice the AI indicator AI at the top showing that the application was generated with Now Assist for Creator.

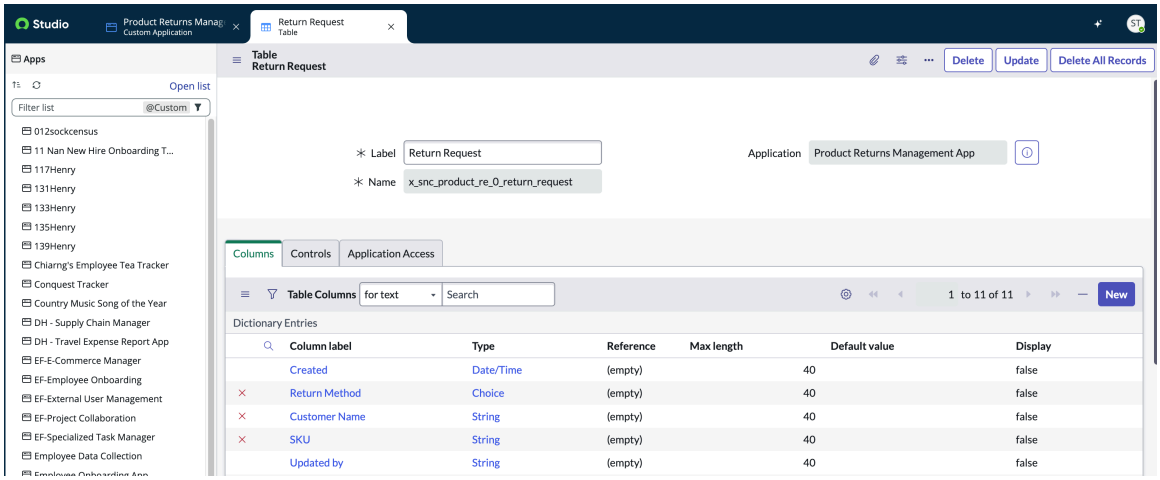
   To learn more about the ServiceNow Studio home page, see Building applications with ServiceNow Studio.

2. Review the generated tables.

   a. On the **Application Files** tab, in the **Display name** column, select an existing table in your application.
   The table record opens in a new tab.

   Note:

   If the Now Assist panel is covering information, select the Now Assist icon to close the panel. Select the icon again to open the panel and continue the conversation.



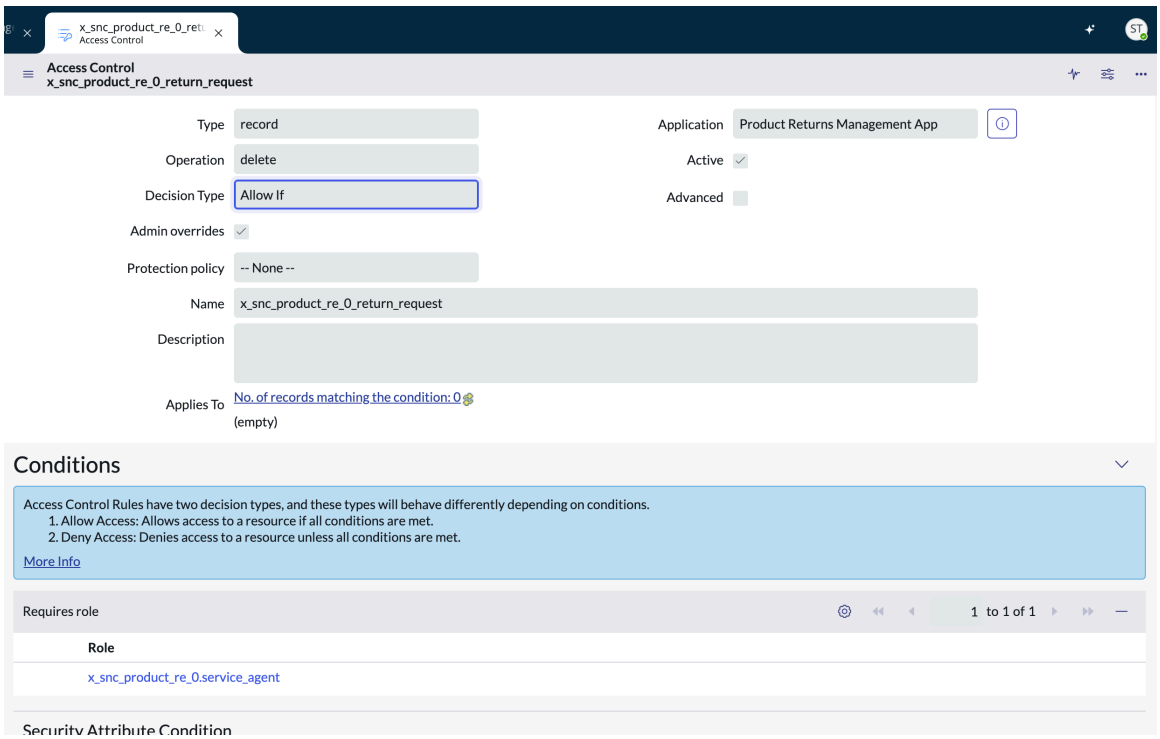
b. Now Assist created the table based on your instructions; confirm that the table is accurate and captures the correct data.

- Manage table fields.
- Edit the table and field column properties.
- Verify that the generated choices are implemented correctly.

3. Close the table tab.

4. Review the generated access control lists (ACLs).

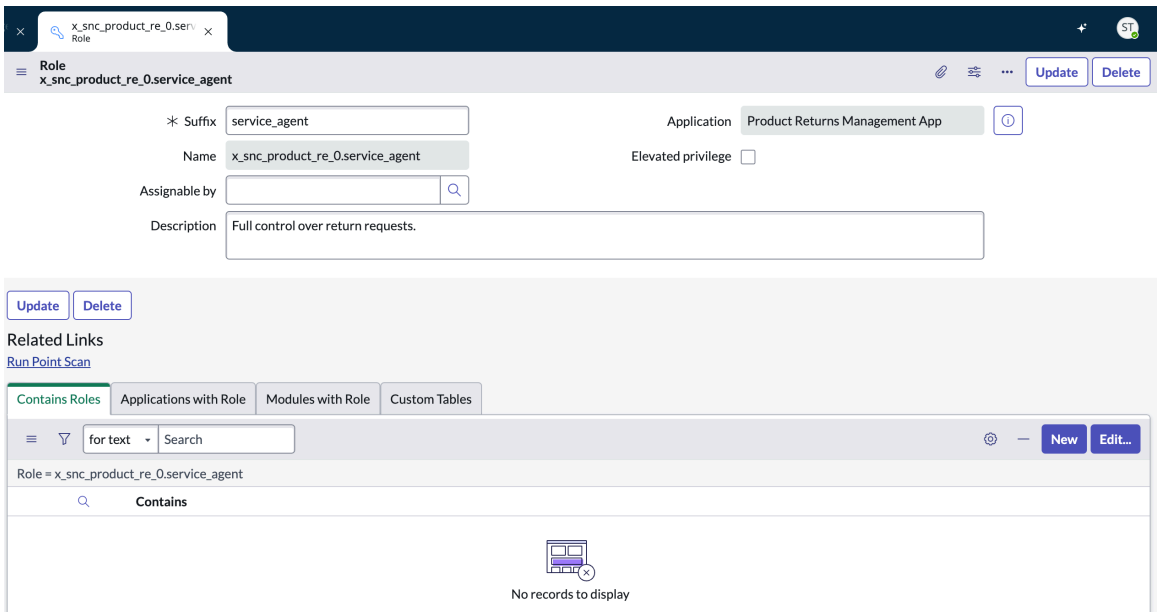
a. On the **Application Files** tab, in the **Display name** column, select an existing access control in your application.
The access control record opens in a new tab.



b. Verify that the created ACL is accurate and requires the correct role or roles.

5. Close the access control tab.
6. Review the generated roles.

   a. On the **Application Files** tab, in the **Display name** column, select an existing role in your application.
   The role record opens in a new tab.



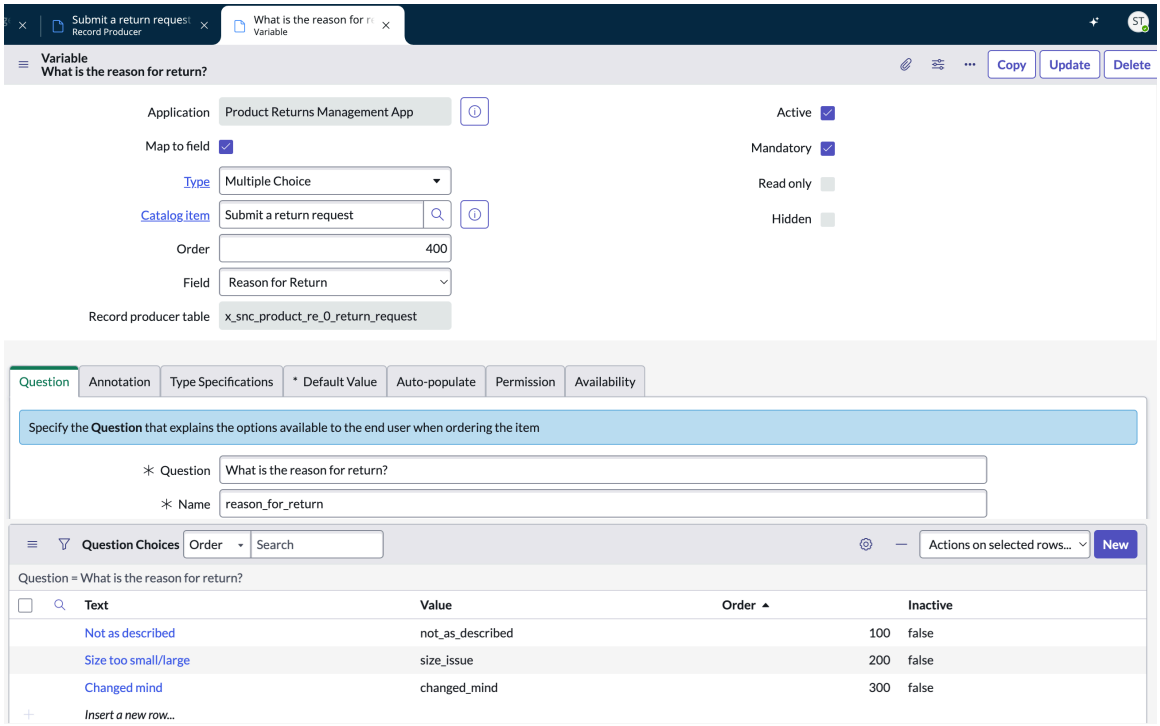
   b. Verify that the created role and its assigned permissions accurately represent the users of your application.

7. Close the role tab.
8. Review the generated record producers.

   a. On the **Application Files** tab, in the **Display name** column, select an existing record producer in your application.
   The record producer record opens in a new tab.

   b. Review and modify the record producer if necessary.

   For example, select the **Variables** tab and then select a **Question** to confirm that the question options are correct.



For more information about record producers, see Record Producer.

9. Close the record producer tab.

What to do next

Continue opening, verifying, and modifying items in the application list as needed. For more information, see Building applications with ServiceNow Studio.

Edit applications in Now Assist for App generation with ServiceNow Studio

In Xanadu Store Release 2, use the generative AI capabilities in Now Assist for Creator to edit applications.

Before you begin

You have created an application using Now Assist for Creator.

Role required: admin and now.assist.creator (as of Xanadu Store Release 2, if you have users that only need to edit, not create, apps, they can be assigned the delegated_developer, now_assist_panel_user, and now.assist.creator roles)

Procedure

1. Open the Now Assist panel by navigating to App Engine > ServiceNow Studio and selecting the Now Assist icon.

2. In the Now Assist panel, select **Update an app**.
3. Select the app to edit.

   The apps that appear in the Now Assist panel for editing are custom applications you have created or to which you have access.

Users with the admin and now.assist.creator roles can create and edit applications. Users with the delegated_developer, now_assist_panel_user, and now.assist.creator roles can edit applications.

If you have the delegated_developer, now_assist_panel_user, and now.assist.creator roles, but do not see the app that you need, contact your App Engine admin. Ask them to add you to the app as a delegated developer. For more information, see Delegated development and deployment.

To edit with Now Assist, ensure that your scope is set to the same scope as the app. If you aren't in the correct scope, return to the browser tab you used to launch ServiceNow Studio and use the scope picker to change the scope. (For more information, see Application picker.) When you're finished, select the tab with ServiceNow Studio open and select **Done**.

4. Interact with Now Assist to edit the app.

   For example, request new metadata such as a form or request a new table that is extended from an existing table.

   Note:
   Now Assist may encounter issues while updating a large app. Consider updating the app manually in ServiceNow Studio instead.

What to do next

Continue interacting with Now Assist to edit and refine the app. For more information, see General guidelines for using app generation in ServiceNow Studio.

App generation in ServiceNow Studio reference

Reference topics provide additional information about configuration properties, roles, and more.

App generation roles for ServiceNow Studio

The following roles are installed for use with the Now Assist for Creator app generation skill.

As of Xanadu Store Release 2, users that only need to edit (not create) applications using app generation can be granted the delegated_developer, now_assist_panel_user, and now.assist.creator roles. For more information, see Delegated development and deployment.

Administrator [admin] role for Now Assist with ServiceNow Studio

Access all system features, functions, and data, regardless of security constraints.

Now Assist Creator [now.assist.creator] role for Now Assist with ServiceNow Studio

Create applications through a conversation with generative AI.

Contains Roles

List of roles contained within the role.

None.

Groups

List of groups this role is assigned to by default.

None.

Special considerations

None.

Bookmark apps and app files in ServiceNow Studio

Easily bookmark apps and app files from several locations for quick access in ServiceNow Studio.

Before you begin
Role required: admin or delegated developer

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Bookmark your apps and files from different locations in ServiceNow Studio.
3. **Optional:** To view your bookmarked apps and files, ensure that the Navigator panel is open and select the bookmarks icon ().
   Your bookmarked apps and files appear in the bookmarks tab under the appropriate file type.

Find an app or app file using code search

Find an app or app file using code search in ServiceNow Studio. Code search enables you to search through all applications and tables on your instance to find what you're looking for.

About this task
You can use code search for troubleshooting issues on your instance. For example, if a client script automatically updates a record incorrectly and displays an error message, you can perform a code search for the error message to find out where it's being generated. For a detailed example of using code search to troubleshoot, see the Community article on Using Code Search to make developing on the platform easier.

Before you begin
Role required: admin or delegated developer

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. In the search bar, select the **Activate code search** toggle.
3. Enter the code snippet you're looking for in the search bar.
4. Specify whether to search all apps or a specific app.
   - Choose **Select all** to search all apps on the instance for the code snippet.
   - Choose **Select specific app** and then enter the name of the app in the search bar that appears.
5. **Optional:** Limit the search to a table by entering the name of the table in the **File types / tables** field.
   Only tables in the same scope are available. For a list of supported file types, see ServiceNow Studio supported file types using code search.
6. Run the search by selecting **View results**.
   The results open in a new **Code search** tab inside ServiceNow Studio.
7. Expand the type of script that you want to review for the code snippet.

For example, you can expand the **Notification** section to view all code search results in notification scripts, or the **Access control** section to view all instances of code related to roles and security.

For a list of types of scripts, see Available script types.

8. View the code you searched for and its corresponding line number by expanding the name of the individual script within the section you're viewing.
   Each script that contains the code snippet appears in an expandable section that displays the number of times the code appears in the script.
9. **Optional:** View the whole script that contains the code by selecting **open file**.
   The complete script appears in a new tab in ServiceNow Studio.

Debug a script in ServiceNow Studio

Use the Script Debugger in ServiceNow Studio to debug business rules and other synchronous server-side scripts. Access the Script Debugger during the course of your app development in ServiceNow Studio.

Before you begin

Role required: admin or delegated developer

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. On the home page, scroll down to the bottom of the screen, and select *`Script Debugger`* from the **Resources** section.
3. Use the Script Debugger features you use in the normal course of your app development work.
   The Script Debugger opens in a new tab outside of ServiceNow Studio. Some tabs and features are role-dependent. For more information about using Script Debugger, see Script Debugger and Session Log.

Modify an app's settings in ServiceNow Studio

Adjust an app's settings in ServiceNow Studio to meet your needs. Change the name, description, or image associated with the app, or delete the app if it's no longer needed.

Before you begin

Role required: admin or delegated developer

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. In the Navigator panel, select the **Apps** icon ( ).
3. Select the app that you want to open.
4. Select **App details** to open the app in the canvas.
5. In the top-right corner, select the App settings icon ( ).
6. Edit the app's settings.
7. Select **Save**.

Working with metadata app file categories in the ServiceNow Studio Navigator

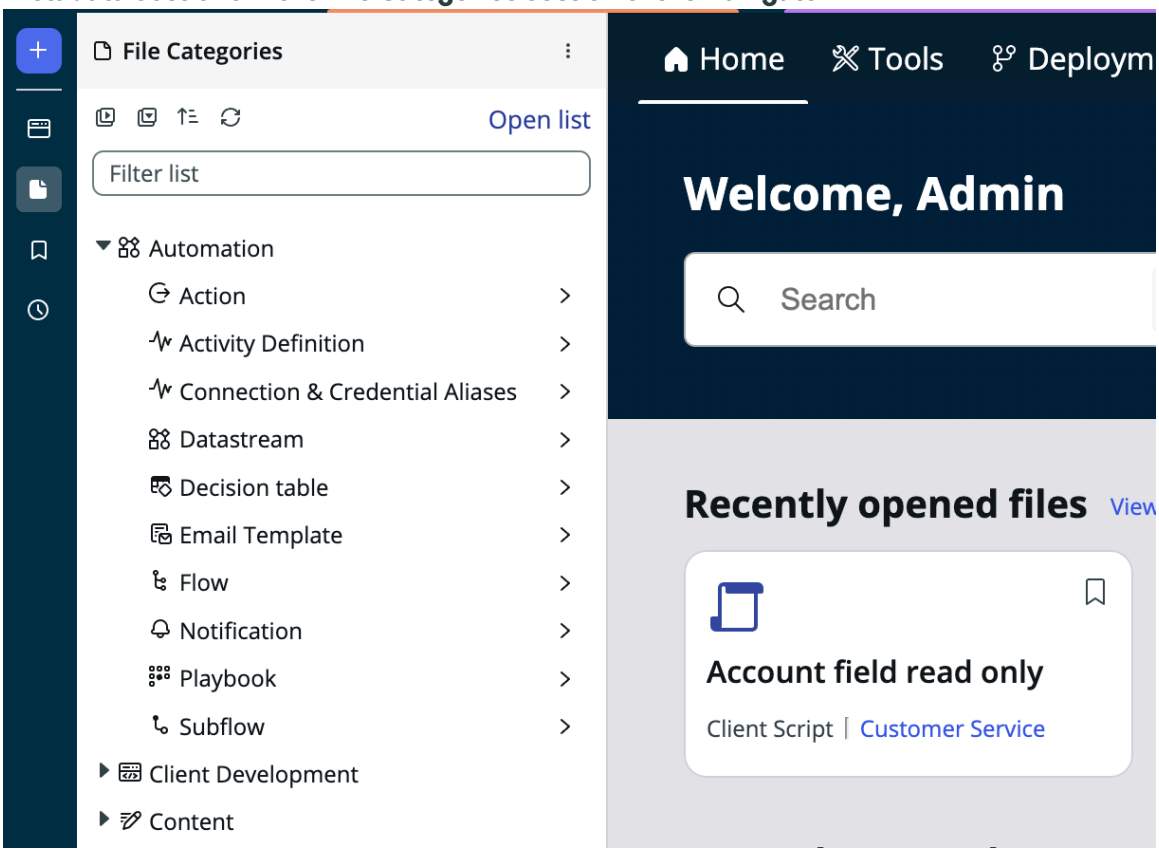
ServiceNow Studio enables you to work with all types of metadata that the ServiceNow AI Platform supports, from automations and integrations to user interface files.

Accessing metadata for your ServiceNow Studio apps

The **File Categories** section of the Navigator panel enables you to quickly view all types of metadata that you have access to, arranged by taxonomy category. Expanding each metadata file category shows its subcategories. For example, expanding the **Automation** category provides access to view all of the sub-types of automation metadata, such as actions, flows, and playbooks.

Note:
To see a complete list of each category's file types and information about each file, see ServiceNow Studio Navigator panel taxonomy.

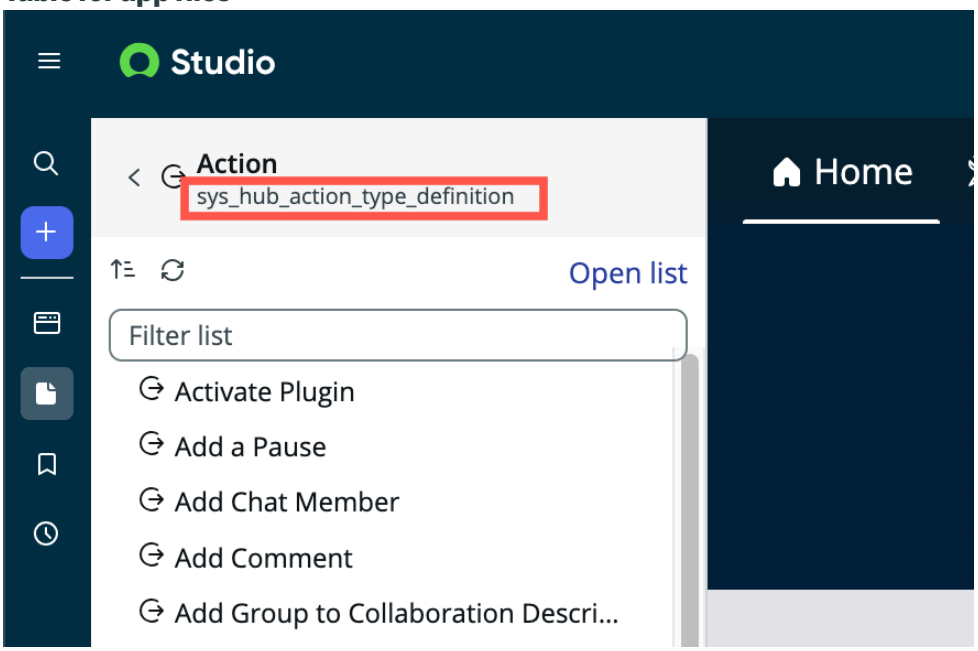
Metadata sections in the File Categories section of the Navigator



When you select a sub-type of an expanded file category, ServiceNow Studio displays all of the metadata of that type, which you can select to view. For example, selecting **Flow** under the **Automation** file category displays a list of all flows that you have access to. Select one to open it in Workflow Studio within a tab in ServiceNow Studio.

Note:
The source table for each sub-category of metadata file type appears when you select the category. Select **Open list** to see a complete list of all the files of that type.

Table for app files



For more information on using the Navigator panel to find app files by their metadata category, see Find an app or app file using the Navigator panel.

Metadata table and its extended tables

Metadata lives as application files in their corresponding tables on the ServiceNow AI Platform. The main metadata table is sys_metadata, but specific types of metadata extend the sys_metadata table. For example, flows and sub-flows use the extended sys_hub_flow table.

See ServiceNow Studio Navigator panel taxonomy for a list of available metadata and corresponding tables that extend sys_metadata.

Metadata by category

Each file category has different types of metadata available for automations, data, integration, mobile app builder, UI, and other categories. For a list of each type of file, its primary table, primary builder experience, and more, see ServiceNow Studio Navigator panel taxonomy.

Categories of metadata

Category	Metadata sections
Automation	• Automation • Schedules
Data	• Client development • Data • MID Server • Server development

Categories of metadata (continued)

Category	Metadata sections
Integrations	• Inbound integration • Outbound integration • Properties
Mobile	• Mobile App Builder • Mobile Card Builder
UI	• Content • User interface
Other	• Natural Language Understanding (NLU) • Reporting • Security

Working with update sets in ServiceNow Studio

An update set is a group of configuration changes that can be moved from one instance to another. This feature allows administrators to group a series of changes into a named set and then move them as a unit to other systems for testing or deployment.

If you're an admin, you might work with update sets in ServiceNow Studio to move changes through your instances. Access update sets from within an individual application or from the **Deployment** tab. Most of the ways you can interact with update sets in ServiceNow Studio are very similar to how you would use them on the ServiceNow AI Platform. For more information about tasks you can accomplish with update sets, see System update sets.

Administrators have the following options with update sets.

- Create an update set to store local changes.
- Select the current update set to store local changes.
- Commit an update set to prepare it for distribution.
- Report on the contents of update sets.
- Compare update sets to determine what differences they contain.
- Merge separate update sets into a single update set.
- Create an external file from an update set.
- Retrieve update sets from remote instances.
- Apply retrieved update sets.
- Back out changes applied from an update set.
- Set system properties related to update sets.

Delegated developers have the following options with update sets in ServiceNow Studio.

- Create an update set to store local changes.
- Switch between update sets easily using the update set picker.
- Package changes for deployment using the **Publish** button in any app.

Create an update set

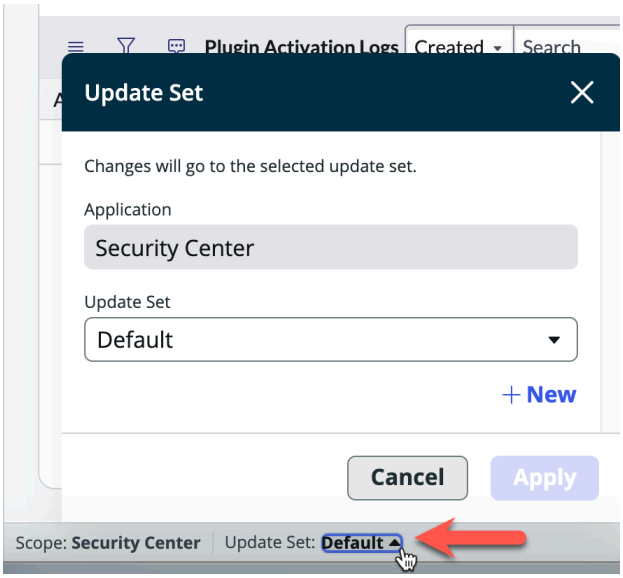
Create an update set to package app changes for deployment in ServiceNow Studio.

Before you begin

Role required: admin or delegated developer

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Open the application that you have changes for that need a new update set.
3. At the bottom-left corner of the canvas, expand the update set dialog by selecting the current update set from the status bar.



4. Select **New**.
5. Enter a name for the new update set.
6. Select **Save**.
7. Select **Apply**.
   The changes you made in your app go to the new update set.

Mark an update set complete in ServiceNow Studio

Mark an update set as **Complete** in ServiceNow Studio to enable your changes to be retrieved by other instances.

Before you begin

Role required: admin or delegated developer

About this task

Only mark an update set as **Complete** when it's ready to migrate. Once an update set is **Complete**, don't change it back to **In progress**. Instead, create another update set for the rest of the changes.

Procedure

1. Navigate to **All** > **App Engine** > **ServiceNow Studio**.
2. Select the **Deployment** tab.
3. Open the update set record you want to change.
4. Change the **State** of the update set from **In progress** to **Complete**.

Working with pipelines in ServiceNow Studio

Pipelines enable you to automate the propagation and installation of your applications from one instance to another. As an admin, you can manage app deployment requests coming from ServiceNow Studio in the App Engine Management Center (AEMC).

Pipelines are powered by the ServiceNow CICD spoke, which enables you to automate processes such as publishing applications to the Application Repository, installing them on target instances, running ATF tests, and running instance scans.

If you're an admin, you might work with pipelines to move changes to your apps through your instances. For more information about configuring pipelines, see Configure Pipelines and Deployments.

If you have the App Engine Management Center (AEMC) installed, you can manage deployment requests that are created when a developer selects **Deploy** in ServiceNow Studio. For more information, see Managing app deployments using Pipelines and Deployments.

Working with the Application Repository in ServiceNow Studio

After you develop and test a custom application in ServiceNow Studio, you can make the application available to company instances by publishing it to the ServiceNow Application Repository. The Application Repository is a central repository for all scoped applications that are published by all ServiceNow customers.

The Application Repository enables you to upload and distribute applications between your instances. When you access the Application Repository, you can see and manage only the applications that are published by your own organization. You can't see or manage applications that are published by other organizations.

After you have developed your app using ServiceNow Studio, successfully tested your app, and selected **Deploy**, you can publish your application to the Application Repository to share it to other instances in your company. For more information about the publishing process, see Publish app changes to the Application Repository from ServiceNow Studio.

For more information, see ServiceNow application repository.

ServiceNow Studio reference

Reference topics provide additional information about the Navigator panel taxonomy, supported file types, properties, and collaboration permissions for ServiceNow Studio.

ServiceNow Studio Navigator panel taxonomy

Learn more about each metadata file type and its corresponding primary table in the File categories tab of the ServiceNow Studio Navigator panel.

Note:
Any additional extensions of the sys_metadata table are automatically included in the Navigator panel in the appropriate section.

For each file category, see the file types that you can create and their corresponding primary table. For more information about navigating to a file's primary table, see Navigate directly to a table in ServiceNow Studio.

Automation files

Automations provide ways to manage recurring tasks efficiently, with fewer steps, reducing human input and effort. For example, you can configure an email notification to be sent automatically when a record is approved. Create automations to streamline and simplify how you work by adding automations to an app, such as actions, flows, and other automatic tasks.

Automation file types

File type	Description	Primary table	Primary editing experience
Action	Actions automate a repeatable task or operation within a flow as a sequence of related steps. Actions run a sequence of steps to complete the task, and pass data to the flow as outputs. For more information, see Exploring actions.	sys_hub_action_type_definition	Workflow Studio
Activity definition	Activity definitions describe how the activities in your playbook get the data that they need when your playbook runs. Each activity definition contains some basic configuration details, as well as an automation plan and activity experience. For more information, see Activity definitions.	sys_pd_activity_definition	UI16
Connection & Credential Aliases	A Connection and Credential alias defines an alias that labels a credential or connection record, enabling an app to connect to another system or component.	sys_alias	UI16

Automation file types (continued)

File type	Description	Primary table	Primary editing experience
	For more information, see Connections and Credentials.		
Datastream	A datastream is a reusable action that processes a stream of response data within a flow. For example, you can create a data stream action to import a large quantity of employee data from a third-party HR site. For more information, see Data Stream actions and pagination.	sys_hub_action_type_definition	Workflow Studio
Decision table	Decision tables decouple decision logic from your code by creating and maintaining decision rules. Decision tables provide a single point where you can create, view, and modify decisions. For more information, see Exploring decision tables.	sys_decision	Workflow Studio
Email Template	Email templates enable administrators to create reusable content for the subject line and message body of email notifications. Admins can add rich text and other items, such as images, to email templates. For more information, see Email templates.	sysevent_email_template	UI16

Automation file types (continued)

File type	Description	Primary table	Primary editing experience
Flow	Flows are automated processes that consist of a trigger and a sequence of reusable actions and flow logic. The trigger specifies when to run the flow. The actions perform a sequence of operations on your data. For example, the Visual Task Boards (VTB) Sample Flow creates and assigns a VTB card whenever a priority 1 incident is created. For more information, see Exploring flows.	sys_hub_flow	Workflow Studio
Notification	Notifications alert users when a record changes. For example, you could get a push notification when a request is rejected. For more information, see System notifications.	sysevent_email_action	UI16
Playbook	Playbooks are sets of automated activities that occur based on a trigger. For example, you can create a playbook for your app to send an email when a request is approved. For more information, see Getting started with Playbooks.	sys_pd_process_definition	Workflow Studio
Subflow	Subflows are processes that consist of a sequence of reusable actions	sys_hub_flow	Workflow Studio

Automation file types (continued)

File type	Description	Primary table	Primary editing experience
	and flow logic, data inputs, and outputs. In contrast to flows, subflows don't have a trigger but instead run when called from a playbook, flow, another subflow, or a script. For more information, see Exploring subflows.		

Client development files

Client development files define operations that occur based on actions you can take when working with an app built in ServiceNow Studio.

Client development file types

File type	Description	Primary table	Primary editing experience
Assignment Data Lookup	Use assignment data lookups to assign a record automatically using Data Lookup and Record Matching. For example, you can automatically set a value in the assigned_to and assignment_group fields for a record when a set of conditions occurs, such as assigning approvals to a group of managers for users below a certain level. For more information, see Defining assignment rules.	dl_u_assignment	UI16
Client Extension Instance	Use a client extension integration to a registered instance of a client extension	sys_client_extension_instance	UI16

Client development file types (continued)

File type	Description	Primary table	Primary editing experience
	point that links a UI script with a client extension point. The UI script is included on pages that invoke the client extension point. For more information, see Using extension points to extend application functionality.		
Client Extension Point	Client extension points extend the functionality of an application without altering the original application code. Extension points can help prevent your custom code interactions from breaking, which often occurs after an upgrade if you directly embed the custom code into the application code. For more information, see Using extension points to extend application functionality.	sys_client_extension_point	UI16
Client Script	Client scripts enable apps on the ServiceNow AI Platform to run JavaScript on the client (web browser) when client-based events occur. For example, it could run when a form loads, after form submission, or when a field changes value.	sys_script_client	UI16

Client development file types (continued)

File type	Description	Primary table	Primary editing experience
	For more information, see Client scripts.		
Data Lookup Definitions	Data lookup definitions are no-code solutions that enable you get attributes from a record on the same table. For example, you can create a data lookup definition to populate an email field automatically when you enter your name. For more information, see Create a catalog lookup definition.	dl_definition	UI16
Priority Data Lookup	Priority data lookups enable you to define the impact and urgency of an incident to calculate how it should be prioritized. For more information, see Define priority lookup rules.	dl_u_priority	UI16
UI Extension Instance	Use a UI extension instance to create a registered instance of a UI extension point that links a UI macro with a UI extension point. The macro can be called whenever the UI extension point is invoked. For more information, see Creating and adding a UI extension point.	sys_ui_extension_instance	UI16

Client development file types (continued)

File type	Description	Primary table	Primary editing experience
UI Extension Point	UI extension points enable you to add custom content to a UI page without having to modify the page directly. You must first create the UI extension points, and then add them to the UI macros in the base application code. For more information, see Using UI extension points in server-side UI macros.	sys_ui_extension_point	UI16
UI Policy	UI policies dynamically change the behavior of information on a form and control custom process flows for tasks. For example, you can use UI policies to make the number field on a form read only, make the short description field required, and hide other fields. For more information, see UI policies.	sys_ui_policy	UI16
UI Script	UI scripts provide a way to package client-side JavaScript into a reusable form, similar to how script includes store server-side JavaScript. For more information, see UI scripts.	sys_ui_script	UI16

Content files

Content files define ways to provide users and systems with information by referring to knowledge articles and other forms of communication. For example, content files can reference knowledge articles, static blocks of text, or external web-based content. Use content files to provide information to apps you're building in ServiceNow Studio.

Content file types

File type	Description	Primary table	Primary editing experience
Audio	Use audio files to upload sounds and recordings that your app can use on the ServiceNow AI Platform. For more information, see Manage audio files.	db_audio	UI16
Detailed content	Use a detailed content block to display the content of an existing document, such as an incident, knowledge article, or service management request. For more information, see Configure a detailed content block.	content_block_detail	UI16
Dynamic content	Dynamic content uses scripting or pulls information from the ServiceNow AI Platform into an app. For example, use dynamic content to create a job posting, where the postings are stored in knowledge articles and displayed in the app with a dynamic block. For more information, see Configure dynamic blocks.	content_block_programmatic	UI16

Content file types (continued)

File type	Description	Primary table	Primary editing experience
iFrames	Use iFrames to embed a URL on a page within a frame. You can embed external pages or render ServiceNow content. For more information, see Configure iFrames.	content_block_iframe	UI16
Images	Upload and store images on the ServiceNow AI Platform to be used in apps and forms. You can then reference images from HTML fields by appending the name of the image to the URL of the instance. For more information, see Storing images in the database.	db_image	UI16
Static content	Use static blocks for text that doesn't change. For example, use a static block for a site footer with only the company or organization name. For more information, see Configure a static HTML block.	content_block_static	UI16

Data files

Data files enable you to use information stored in or referenced by an application in apps built in ServiceNow Studio. Data is the foundation of any app. You can start with a table, or upload spreadsheets and PDFs to get data into the ServiceNow AI Platform for apps to consume.

Data file types

File type	Description	Primary table	Primary editing experience
Form	A form is a content page that displays fields and values for a single record from a database table. For more information, see Forms in Table Builder.	sys_ui_form	Form Builder
Form section	Use sections to design the layout of a form. For example, you can have one column or two. For more information, see Customize your form layout in Table Builder.	sys_ui_section	Form Builder
Many to Many Definition	Use a many to many task to define relationships between different tasks. You can implement one-to-one, one-to-many, and many-to-many relationships. For example, users and roles are a many-to-many relationship because a user can have multiple roles, and multiple users can have any given role. For more information, see Creating many-to-many task relations.	sys_m2m	UI16
Relationship	Use relationships to define how tables interact with each other. You can create relationships between tables by extending tables, referencing	sys_relationship	UI16

Data file types (continued)

File type	Description	Primary table	Primary editing experience
	records in another table, creating many-to-many relationships, and joining tables in a database view. For more information, see Table relationships.		
Table	Tables are the foundation of how the ServiceNow AI Platform stores data. When you view a table as a list, each row is a record, and each column is a field from the record. For example, the Incident table has a record for every customer interaction, or incident. For more information, see ServiceNow AI Platform tables and data and Table Builder.	sys_db_object	Table Builder
Table Column	A table column represents a field from a record. For example, a user record might have a column for first name and a separate column for family name. For more information, see Table properties in Table Builder.	sys_dictionary	UI16

Inbound integration files

Inbound integrations enable you to get information and data onto the ServiceNow AI Platform from another system or source. Inbound integrations enable seamless data flow into ServiceNow from external systems, facilitating efficient communication and integration between systems.

Inbound integration file types

File type	Description	Primary table	Primary editing experience
Data Import	Use data imports to view the all the records that are being processed for an import job and also the import jobs that are awaiting approvals. For more information, see Importing data using import sets and Integration Hub - Import.	sn_ihub_integration_instance	Integration Hub
Data Source	A data source specifies how and where to get the data you want to import. For more information, see Data sources and Configure a data source.	sys_data_source	UI16
Scheduled Data Import	Scheduled data imports specify to import data from data sources using import sets. Transform maps are applied to the imported data before writing the data to the target table. For more information, see Run or schedule a data import.	scheduled_import_set	UI16
Scheduled Data Import	Scheduled data imports specify to import data from data sources. Transform maps are applied to the imported data before writing the data to the target table.	scheduled_data_import	UI16

Inbound integration file types (continued)

File type	Description	Primary table	Primary editing experience
	For more information, see Run or schedule a data import.		
Scripted REST API	Use scripted REST APIs to build custom web service APIs for your application. You can define service endpoints, query parameters, and headers for a scripted REST API, as well as scripts to manage the request and response. For more information, see Scripted REST APIs.	sys_ws_definition	UI16
Scripted Web Service	Scripted web services enable developers to create their own APIs on the ServiceNow AI Platform. Third-party applications use scripted web services to access records in ServiceNow tables. For more information, see Web services.	sys_web_service	UI16
Table Transform Map	Transform maps contain a set of field maps that determine the relationships between fields in an import set and fields in an existing ServiceNow table, such as Incident [incident] or User [sys_user]. After creating a transform map, you can reuse it to map data from another import set to the same table.	sys_transform_map	UI16

Inbound integration file types (continued)

File type	Description	Primary table	Primary editing experience
	For more information, see Transform maps.		

Outbound integration files

Outbound integrations enable you to work with data that comes from the ServiceNow AI Platform and is sent to an external system or source.

Outbound integration file types

File type	Description	Primary table	Primary editing experience
Export Definition	Export definitions determine the data to include in an export set. For more information, see Create an export definition.	sys_export_definition	UI16
Export Set	Export sets define the data to export, as well as the export target to use when exporting data. For example, you can push data from an instance to an external file. For more information, see Create an export set.	sys_export_set	UI16
Export Target	Export targets specify the target file on a MID Server to which the export set data will be written. For more information, see Create an export target.	sys_export_target	UI16
REST Message	A REST message is a record that stores details on how to	sys_rest_message	UI16

Outbound integration file types (continued)

File type	Description	Primary table	Primary editing experience
	interact with an external web service through REST. Use REST messages to send requests to a REST web service endpoint by creating a REST message record. For more information, see Create a REST message.		
Scheduled Data Export	Scheduled data exports specify a schedule when export sets will be run. A single export can be scheduled or regular intervals can be scheduled with support for including only delta records. For more information, see Schedule an export.	scheduled_data_export	UI16
SOAP Message	A SOAP message is a record that stores details on how to interact with an external web service through SOAP. SOAP messages define the remote endpoint, web services description language (WSDL), and authentication settings. For more information, see SOAP message.	sys_soap_message	UI16

MID Server files

Work with management, instrumentation, and discovery (MID) Server files to facilitate communication and data movement between a single ServiceNow instance and external applications, data sources, and services.

MID Server file types

File type	Description	Primary table	Primary editing experience
MID Server Application	The ServiceNow MID Server is a Java application that runs as a Windows service or UNIX daemon on a server in your local network. For more information, see MID Server.	ecc_agent_application	UI16
MID Server Capability Value Test	MID Server capabilities define the specific functions of a MID Server within an IP address range. The capability value can be empty, a single value, or a * (wildcard). You can use value tests to create capabilities that find devices using values without requiring exact string matching. For more information, see MID Server capabilities.	ecc_agent_capability_value_test	UI16
MID Server IP Range	Use MID Server to specify an IP range or the specific IP address of a target. For more information, see Configure an IP address range for the MID Server.	ecc_agent_ip_range	UI16
MID Server Property	Use MID Server properties to define the behavior of one or more MID Servers.	ecc_agent_property	UI16

MID Server file types (continued)

File type	Description	Primary table	Primary editing experience
	For more information, see MID Server properties.		
MID Server File	Use a MID Server script file to synchronize to a connected MID Server. For more information, see Attach a script file to a file synchronized MID Server.	ecc_agent_script_file	UI16
MID Server Script Include	Use MID Server script includes to make REST calls to cloud providers. For more information, see CAPI classes in MID Server script includes.	ecc_agent_script_include	UI16

Mobile App Builder files

Mobile App Builder is a configuration tool to build and manage screens and records that make up workflows within ServiceNow mobile apps. The organizational layout and navigation options in the Mobile App Builder facilitate a faster and more intuitive creation of ServiceNow mobile applications.

Mobile App Builder file types

File type	Description	Primary table	Primary editing experience
Analytics preview	Analytics previews display previews of data visualization charts and single score reports in your launcher screen's analytics section. Analytics previews enable you to verify that your data is tailored for mobile use and communicates	sys_sg_chart	Mobile App Builder

Mobile App Builder file types (continued)

File type	Description	Primary table	Primary editing experience
	the appropriate information for users. For more information, see Create a mobile analytics preview.		
Calendar screen	Calendar screens display a calendar interface and records associated with the selected date. You can use a calendar screen to display dates that are relevant to application records. For example, you can display when tasks are due or when important events take place. For more information, see Calendar screen.	sys_sg_calendar_screen	Mobile App Builder
Chart screen	Chart screens display data visualizations created in the Analytics Center and are displayed in the analytics section of your launcher screen. Adding data visualizations helps you identify trends and turning points through indicator scores and visual representation. For more information, see Chart screen.	sys_sg_chart_screen	Mobile App Builder
Custom map screen	Custom map screens enable you to create maps that display content for specific records.	sys_sg_custom_map_screen	Mobile App Builder

Mobile App Builder file types (continued)

File type	Description	Primary table	Primary editing experience
	For more information, see Configure a map screen.		
Function	Functions determine what actions users can perform in your mobile application. For example, you can create a navigation function that enables users to open a record from a list, or move from an employee user profile screen to a manager user profile screen. For more information, see Mobile functions.	sys_sg_button	Mobile App Builder
Input form screen	Input form screens provide interfaces for users to enter information in mobile applications. For example, you can use input form screens to create or edit records, complete surveys, or any other situation where your users must enter information. For more information, see Input form screen.	sys_sg_parameter_screen	Mobile App Builder
Launcher screen	Launcher screens serve as landing pages or home pages. Using a launcher screen, you can access screens in various formats, search, perform quick actions, and find user information.	sys_sg_applet_launcher	Mobile App Builder

Mobile App Builder file types (continued)

File type	Description	Primary table	Primary editing experience
	For more information, see Launcher screens.		
List screen	List screens display a list of records. Records in list screens appear in a card format, showing a limited selection of the information for the record. For more information, see List screen.	sys_sg_list_screen	Mobile App Builder
Map screen	Map screens display a map with locations that are associated with the records in a data item. For example, map screens can show your users where their assets are located, or which job locations they must travel to. For more information, see Map screen.	sys_sg_map_screen	Mobile App Builder
Mobile App config	Mobile app configs enable you to create customized mobile experiences for the Now Mobile app and Mobile Agent app. For more information, see Configuring the Mobile Platform.	sys_sg_native_client	Mobile App Builder
Mobile web screen	Mobile web screens open an external URL or a relative URL within your instance.	sys_sg_browser_screen	Mobile App Builder

Mobile App Builder file types (continued)

File type	Description	Primary table	Primary editing experience
	For more information, see Mobile web screen.		
Record screen	Record screens display content for a specific single record. You can configure functions on record screens to enable users to make edits and perform actions. For more information, see Record screen.	sys_sg_form_screen	Mobile App Builder

Mobile Card Builder files

Mobile Card Builder files enable you to edit the cards and templates used in applications for iOS and Android.

Mobile Card Builder file types

File type	Description	Primary table	Primary editing experience
Card	Cards are predetermined layouts that can show visuals, text, and data in mobile applications. You can define card elements and specify how elements are arranged within a card. For more information, see Cards and icons.	sys_sg_view_config	Mobile Card Builder
Card template	Card templates are preconfigured layouts or frameworks that determine how information is displayed in mobile application cards. You can use the existing	sys_sg_view_template	Mobile Card Builder

Mobile Card Builder file types (continued)

File type	Description	Primary table	Primary editing experience
	Mobile Card Builder card templates or create your own templates. For more information, see Create a card template with Mobile Card Builder.		

Natural Language Understanding (NLU) files

Natural Language Understanding (NLU) is a model that enables a computer to interpret, analyze, and derive meaning from human language. By creating NLU files, you build and train your application's NLU model to help it recognize user input, or utterances, and determine the corresponding user or system actions.

NLU file types

File type	Description	Primary table	Primary editing experience
NLU Model	A Natural Language Understanding (NLU) model is the collection of utterance examples and their associated intents and entities that an application uses as a reference to infer intents and entities in a new utterance. For more information, see Natural Language Understanding.	sys_nlu_model	UI16

Properties files

Properties files are configurable parameters that enable you to change the behavior of an application without hard-coding the values directly into scripts.

Properties file types

File type	Description	Primary table	Primary editing experience
Message	Messages are the text values used in informational messages, confirmation messages, error messages, and other types of system messages in your application. For more information, see Message table.	sys_ui_message	UI16
System Property	A system property is a way to store important values for an application, like settings or configurations, that you might need in different scripts. Instead of writing these values directly into the script (hard-coding), you can store them as system properties. Creating system properties enables you to update and edit properties in one place, without having to change each script that references the values manually. For more information, see What are application properties? and Available system properties.	sys_properties	UI16
System Property Category	A system property category creates the page layout for an application's system properties.	sys_properties_category	UI16

Properties file types (continued)

File type	Description	Primary table	Primary editing experience
	The system property category page contains all the application properties in a single location. For more information, see Create a system property category.		

Reporting files

Reporting files enable you to create and distribute reports that show the current state of instance data, such as the number of open incidents of each priority.

Reporting file types

File type	Description	Primary table	Primary editing experience
Chart Colors	Chart colors assign a consistent color to a grouping or stacking value in reports and dashboards. The color stays the same across reports regardless of the order of the values. For more information, see Chart colors.	sys_report_chart_color	UI16
Color Definition	Color definitions enable you to maintain consistency in the platform's look and feel by applying defined colors to various UI components, such as buttons, backgrounds, text, and other elements. For more information, see Define system colors for analytics.	sys_report_color	UI16

Reporting file types (continued)

File type	Description	Primary table	Primary editing experience
Dashboard	Dashboards enable you to display performance analytics, reporting, and other widgets on a single screen. You can use dashboards to create a story with data that you can share with other users. For more information, see Create and use dashboards.	pa_dashboards	Dashboard Builder
Metric Definition	A metric measures and evaluates the effectiveness of an application process. For example, a metric could measure the effectiveness of the incident resolution process by calculating how long it takes to resolve an incident. You can define metrics and create reports and dashboards using your metrics definitions. For more information, see Metrics and Define a metric.	metric_definition	UI16
Range	Ranges are defined data intervals that are used in bar and pie charts to segment the data into logical groups. For example, you might create a range to see how many tasks were completed well within the service level	sys_report_range	UI16

Reporting file types (continued)

File type	Description	Primary table	Primary editing experience
	agreement (SLA) and how many tasks elapsed during the SLA. For more information, see Report ranges.		
Report	Reports are tools used to display data visually, enabling you to gain insights, track performance, and make data-driven decisions. For example, you can create and distribute reports that show the current state of instance data, such as the number of open incidents of each priority. For more information, see Exploring reporting.	sys_report	UI16
Scheduled Email of Report	Scheduled emails of reports enable you to generate and distribute scheduled reports via email. For more information, see Schedule emails of reports.	sysauto_report	UI16

Schedules files

Schedules can help you manage the entire life cycle of your application, from setting up maintenance schedules to defining durations and risks for operations.

Schedules file types

File type	Description	Primary table	Primary editing experience
Blackout Schedule	A blackout schedule is a time during which certain activities, such as changes or updates, are restricted to avoid disruptions. You can set up blackout schedules to confirm that critical business operations remain unaffected during high-impact or sensitive times, such as holidays, end-of-quarter financial processing, or other key business events. For more information, see Create blackout and maintenance schedules in Change Management.	cmn_schedule_blackout	UI16
Maintenance Schedule	A maintenance schedule is a time during which planned maintenance activities, such as changes and updates, should take place. Maintenance schedules usually occur during low-impact times to minimize disruptions to business operations. For more information, see Create blackout and maintenance schedules in Change Management.	cmn_schedule_maintenance	UI16
Relative Duration	Relative durations are a duration type available in ServiceNow Studio	cmn_relative_duration	UI16

Schedules file types (continued)

File type	Description	Primary table	Primary editing experience
	that you can select when defining service level agreements (SLAs). Relative durations enable you to calculate how much time you have to work on an SLA by defining the amount of time to wait. For example, you can define a relative duration as three business days by 4pm. For more information, see Define a relative duration and Use a relative duration.		
Risk Conditions	A risk condition is a set of rules or criteria that evaluate the potential risks associated with scheduling activities, such as changes, updates, or maintenance tasks. You can define risk conditions and run risk calculations using The Best Practice - Change Risk Calculator. For more information, see Add or modify risk and impact conditions and Risk conditions and calculation.	risk_conditions	UI16
Schedule	Schedules are rules that include or exclude time for various actions or tasks.	cmn_schedule	UI16

Schedules file types (continued)

File type	Description	Primary table	Primary editing experience
	For more information, see Schedules and Define a schedule.		

Security files

Security files enable you to control who has access to application data and help prevent accidental modification or deletion of data.

Security file types

File type	Description	Primary table	Primary editing experience
Access Control	Access control, or access control lists (ACLs), restrict access to data by requiring users to pass a set of requirements before they can interact with application content. For more information, see Exploring Access Control Lists.	sys_security_acl	UI16
Public Pages	Public pages enable users to see the application content without logging in. For more information, see Make UI pages public or private.	sys_public	UI16
Role	Roles determine what application access is granted to which users. For more information, see Managing roles and Determine What Roles to Create.	sys_user_role	UI16

Server development files

Server development files manage back-end processes in ServiceNow Studio to verify that data is handled appropriately and securely.

Server development file types

File type	Description	Primary table	Primary editing experience
Business Rule	A business rule is a server-side script that runs when a record is displayed, inserted, updated, or deleted, or when a table is queried. You can establish server-side conditions to determine when a business rule script should run and what record operations the business rule applies to. For more information, see Classic Business rules.	sys_script	UI16
Data Policy	Data policies enable you to enforce data consistency by setting mandatory and read-only states for fields. Data policies are similar to UI policies, but UI policies only apply to data entered on a form through the standard browser. Data policies can apply rules to all data entered into the system, including data brought in through import sets or web services and data entered through the mobile UI. For more information, see Data policy.	sys_data_policy2	UI16

Server development file types (continued)

File type	Description	Primary table	Primary editing experience
Event Registration	Events are special records that the system uses to log when certain conditions occur and to take some kind of action in response to the conditions. By registering an event, you can define properties about the event and associate the event with the business rule that fires the event. For more information, see Register an event.	sysevent_register	UI16
Extension Instance	An extension instance is a registered instance of a scripted extension point that links a script include with a scripted extension point. When you want to define custom logic or methods without affecting your original code, you may consider using an extension instance. An extension instance enables you to encapsulate specific logic and functions, making it easier for you to manage, update, and debug your code. For more information, see Using extension points to extend application functionality.	sys_extension_instance	UI16

Server development file types (continued)

File type	Description	Primary table	Primary editing experience
Extension Point	An extension point designates where custom script logic can be incorporated into your code, so that you can integrate customizations and new features without altering the existing code for your application. Data or objects returned by an extension point must conform to requirements that are specified by the application creator. For more information, see Using extension points to extend application functionality.	sys_extension_point	UI16
Fix Script	Fix scripts are server-side JavaScript that run after a custom application is installed or upgraded. You can include fix scripts to make changes that are necessary for the data integrity or product stability of an application. For more information, see Fix scripts and Create a fix script.	sys_script_fix	UI16
Scheduled Script Execution	Scheduled script executions, also known as scheduled jobs, are automated, server-side script logic that execute at a specific time or on a recurring basis. Use scheduled script executions when	sysauto_script	UI16

Server development file types (continued)

File type	Description	Primary table	Primary editing experience
	application processes require script logic to be executed based on a time schedule. For more information, see What is a Scheduled Script Execution? and Creating a Scheduled Script Execution.		
Script Action	A script action is server-side JavaScript that is executed when a particular event is generated. For more information, see Script actions	sysevent_script_action	UI16
Script Include	Script includes are reusable, server-side JavaScript that define a function or class and execute only when explicitly called. For more information, see Script includes and Script Includes.	sys_script_include	UI16
UI Action	UI actions are configurations that define the behavior of buttons, links, or context menu items in your application, specifying how they interact with the server-side database. For more information, see UI actions and Create a UI action.	sys_ui_action	UI16

User interface files

User interface files define layouts for pages, modules, and tools that users interact with. Some examples of user interface files include catalog items, guided tours, and themes.

User interface file types

File type	Description	Primary table	Primary editing experience
Application Menu	An application menu is a grouping of modules as they appear in the application navigator (UI16) or **All** menu (Next Experience). You can refer to an Application menu as simply an application. For more information, see Enable or disable an application menu or module.	sys_app_application	UI16
Assessment Metric	In the Assessments application, a metric is a trait or value used to evaluate assessable records. For more information, see Assessment metrics.	asmt_metric	Survey Designer
Catalog	A catalog is a section of the Service Catalog where users can order items and services. A catalog is like a portal where your users can request catalog items such as service and product offerings. For example, a hardware catalog may have items to request a new keyboard or a new mouse device. For more information, see Exploring Service Catalog.	sc_catalog	Catalog Builder

User interface file types (continued)

File type	Description	Primary table	Primary editing experience
Catalog Item	A catalog item is essentially a form that describes a good or service you can order in the service catalog. For example, if you're requesting time off using a catalog item, you may enter your name and requested dates off on the form. For more information, see Service Catalog items.	sc_cat_item	UI16
Context Menu	The context menu for a form provides controls for a list or form based on the table and user access rights. For more information, see Form context menu.	sys_ui_context_menu	UI16
Embedded Help	Embedded Help provides targeted help content to a user in a UI page, based on their role. Some embedded help content comes with the base instance. Your organization can add or replace embedded help content. For more information, see Embedded Help.	sys_embedded_help_content	UI16
Embedded Help Qualifier	An Embedded Help qualifier is an identifier that helps a ServiceNow instance identify the correct	sys_embedded_help_qualifier	UI16

User interface file types (continued)

File type	Description	Primary table	Primary editing experience
	Embedded Help topic when there could be more than one topic for a UI page. For more information, see Use qualifiers in Embedded Help.		
Guided Tour	Guided Tours contain a series of interactive steps that help users complete online tasks within their browser window to help train and onboard users working in a ServiceNow app. For more information, see Guided tours.	sys_embedded_tour_guide	UI16
List	Lists display a set of records from a table, and can be filtered to refine the contents. For example, you can filter the Task list to show only Unassigned tasks. Each row in a list is a record, and each column is a field from the record. For more information, see Lists in the classic environment and List administration.	sys_ui_list	UI16
List Control	List controls are settings that specify which features are available on a list, such as the **New** and **Edit** buttons. For more information, see Configure list controls.	sys_ui_list_control	UI16

User interface file types (continued)

File type	Description	Primary table	Primary editing experience
Map Page	Map pages display ServiceNow data graphically on a Google map page based on location data that you provide. For more information, see Map pages.	cmn_map_page	UI16
Module	A module is any link in the application navigator (UI16) or **All** menu (Next Experience) that opens a page in the content frame or in a separate tab or window. For more information, see Modules on the Developer Site.	sys_app_module	UI16
Page Collection	Page Collections are groups of pages that can be reused across multiple experiences. For more information, see Page collections.	sys_ux_extension_point	UI Builder
Portal	Portals provide users with access to the services, information, and resources they need to get their work done quickly and efficiently. Work with portals in UI Builder. For more information, see Configure UI Builder portal experiences.	sys_ux_page_registry	UI Builder
Record Producer	Record producers are catalog items that	sc_cat_item_producer	Catalog Builder - Record producer

User interface file types (continued)

File type	Description	Primary table	Primary editing experience
	enable users to create task-based records, such as incident records, from the service catalog. For more information, see Record Producer.		
Related List	Related lists appear on forms and show records in tables that have relationships to the current record. For more information, see Related lists.	sys_ui_related_list	UI16
Schedule Page	A schedule page is a record that contains a collection of scripts that enable custom generation of a calendar or timeline display. For more information, see Schedule Pages.	cmn_schedule_page	UI16
Service Portal	Service Portal enables you to build a mobile-friendly self-service portal experience for your employees or customers. For more information, see Service Portal.	sp_portal	Service Portal
Style	Styles define properties such as font size, border, and alignment for text that appears in your app. For more information, see Create a Next Experience style	sys_ui_style	UI16

User interface file types (continued)

File type	Description	Primary table	Primary editing experience
	and Style - Scoped, Global.		
Template	Templates enable administrators to create reusable content. For example, an email template could have a reusable subject line and message body for email notifications. Form templates simplify the process of submitting new records by populating fields automatically. For more information, see Email templates, Form templates, and Page templates.	sys_template	UI16
Theme	Themes enable you to tailor the visual experience for your users, helping to update the look and feel to be more like your brand. For more information, see Working with themes in Next Experience.	sys_ui_theme	UI16
Timeline Page	Use timeline pages to track any activity bounded by two dates, such as change request start and end dates, or incident open and close dates. For more information, see Timeline pages.	cmn_timeline_page	UI16
UX App Configuration	UX app configuration files store	sys_ux_app_config	UI16

User interface file types (continued)

File type	Description	Primary table	Primary editing experience
	configuration settings for specific applications built with UI Builder. For more information, see UI Builder.		
UX Application	Use UX application files to register and manage pages in UI Builder. For more information, see UI Builder.	sys_ux_page_registry	UI16
View Rule	Use view rules to force a specified view when users access a page or application. For more information, see Create a view rule.	sysrule_view	UI16
Workspace	Workspaces are spaces that provide agents and managers with tools to help answer customer questions and resolve customer problems. Workspaces are primarily used for request and fulfillment processes, such as a service desk to manage tickets. For more information, see Configurable Workspace UI and Add a workspace.	sys_ux_page_registry	Workspace Builder

ServiceNow Studio properties

System properties control system behavior. The properties in this section are specific to the ServiceNow Studio application and delegated development deployment. You can access system properties for ServiceNow Studio by navigating to **All** > **sys_properties.list**.

System properties

System property	Description
com.snc.dd.manage_update_set_enabled	Enables or disables display of the **Manage Update Set** permission. • Type: true | false • Default value: false • To enable the display of this permission, set this value to **true**. Delegated developers cannot see the update sets list unless granted permissions via the system property shown.
com.snc.dd.publish_to_app_repo_enabled	Enables or disables display of the **Publish To App Repo** permission. • Type: true | false • Default value: true • To disable display of this permission, set this value to **false**.
com.snc.dd.publish_to_app_store_enabled	Enables or disables display of the **Publish To App Store** permission. • Type: true | false • Default value: true • To disable display of this permission, set this value to **false**.
com.snc.dd.publish_to_update_set_enabled	Enables or disables display of the **Publish To Update Set** permission. • Type: true | false • Default value: false, which disables display of this permission. • To enable the display of this permission, set this value to **true**.
com.snc.dd.upgrade_app_enabled	Enables or disables display of the **Upgrade App** permission. • Type: true | false • Default value: true • To disable display of this permission, set this value to **false**.
sn_devstudio.servicenow_studio_banner.enable	Shows the banner for navigating to ServiceNow Studio from App Engine Studio or

System properties (continued)

System property	Description
	legacy Studio. Change this property to false to hide the banner. • Type: true | false • Default value: true • Location: System Properties [sys_properties] table

ServiceNow Studio supported file types using code search

ServiceNow Studio supports the following file types when using code search.

When using code search in ServiceNow Studio, the application uses the default code search group. The following table outlines the file types that are supported when using the default ServiceNow Studio code search.

Tip:
If your company wants to expand the list of supported file types used in code search, an administrator can create a new code search group that includes additional file types. For more information about expanding code search, see Expand code search tables.

Supported file types for code search

File type	Code search table name	Description
Access control	sys_security_acl	Determines whether access is granted for a specified operation to a specific entity, such as: • Type of entity being secured • Operation being secured • Unique identifier describing the object
Business rules	sys_script	Customize system behavior. • Business Rules run when a database action occurs (query, insert, update, or delete). • The script can run: ○ before or after the database action is performed (runs as part of the database operation).

Supported file types for code search (continued)

File type	Code search table name	Description
		○ asynchronously (at some point after the database operation). ○ on display (when displaying the data in a form).
Client scripts	sys_script_client	Used for changing the appearance of forms, displaying different fields based on values that are entered or other custom display options. • onLoad means that the Client Script runs when the form or page is loaded • onChange means that the Client Script runs when something specific gets changed AND also when the form or page loads • onSubmit means that the Client Script runs when the form is submitted Client Scripts can also be called by other scripts or modules, including UI policies.
Email templates	sysevent_email_template	Enable administrators to create reusable content for the subject line and message body of email notifications.
Inbound email actions	sysevent_in_email_action	Script how the system responds to inbound emails.
Map pages	cmn_map_page	Display ServiceNow data graphically on a Google map page based on location data that you provide.
Map transforms	sys_transform_map	Used for importing data. Transform maps:

Supported file types for code search (continued)

File type	Code search table name	Description
		• Define mapping relationships between tables. • Can use business rules, other scripts, or other options to import data.
Notifications	sysevent_email_action	Determine how an application communicates with users and alerts them about important application-related events.
Processors	sys_processor	Provide customizable URL endpoints that can execute arbitrary server-side JavaScript code and produce output such as TEXT or JSON.
Relationships	sys_relationship	Used to extend tables, reference records in another table, create many-to-many relationships, and join tables in a database view.
Scheduled script executions	sysauto_script	Define automated, server-side script logic that executes at a specific time or on a recurring basis. Scheduled script executions are also referred to as scheduled jobs.
Script actions	sysevent_script_action	Contain scripts that run when an event occurs, for example: • Approval is canceled • Change is approved • Problem is assigned
Script includes	sys_script_include	Contain scripts that can be functions or classes. These scripts run only when called by other scripts, often business rules. Any server script that is complicated or reusable should be a Script Include, especially complicated business rules.
Schedule items	sys_trigger	Contain the back-end data for the System scheduler, where scheduled jobs are created,

Supported file types for code search (continued)

File type	Code search table name	Description
		queued up, and executed. Schedule items can execute scheduled jobs, business rules, inactivity monitors, service level agreement (SLA) calculations, metric events, upgrades, and more. You can access schedule item records to troubleshoot scheduling.
UI actions	sys_ui_action	Include the buttons, links, and context menu items on forms and lists. Configure UI actions to make the UI more interactive, customized, and specific to user activities.
UI macros	sys_ui_macro	Contain discrete scripted components that administrators can add to the user interface. UI macros are typically controls that provide inputs or information that is not provided by existing field types. By default, the system provides UI macros for a variety of user interface elements.
UI pages	sys_ui_page	Used to create and display forms, dialogs, lists, and other UI components.
UI policies	sys_ui_policy	Define the behavior and visibility of fields on a form. Fields can be one of the following: • Mandatory • Visible • Read only The following apply to usage: • UI policies are always attached to one table. • UI policies often have a condition that must be true for them to run.

Supported file types for code search (continued)

File type	Code search table name	Description
UI scripts	sys_ui_script	Contains client scripts stored for reuse. Only used when called from other scripts.
UI style	sys_ui_style	Enables you to declare individual CSS styles for a field in a list or form.
Widgets	sp_widget	Describe objects that contain content and can be added to or embedded in portal pages. You can use the base system widgets provided with the Service Portal, clone and modify widgets, or develop custom widgets to fit your own needs.

Collaboration permissions for ServiceNow Studio

Collaboration permissions determine what delegated developers can do when working on an app in ServiceNow Studio. They can be set by an admin or the app owner.

For more information, see Collaborating on apps using ServiceNow Studio.

File types custom collaboration permissions

File type collaboration permissions

Permission	Description	Owner default setting	Editor default setting
All File Types	Grants access to work with all types of files, including additional file types not granted by the other options.	Yes	Yes
Integrations	Grants access to web service APIs, REST APIs, data sources, and Integration Hub - Import.	Yes	Yes
Reporting	Grants access to reports and scheduled reports.	Yes	Yes
Mobile builders	Grants access to build mobile experiences, such as with Mobile App Builder.	Yes	Yes
UI Builder	Grants access to work with UI Builder to build	Yes	Yes

File type collaboration permissions (continued)

Permission	Description	Owner default setting	Editor default setting
	out more complex interfaces.		
Workflow	Grants access to Workflow Editor and Activity Creator.	Yes	Yes
Service Portal	Grants access to work with Service Portal editors and tools.	Yes	Yes
Workflow Studio	Grants access to the Workflow Studio design environment to create flows and actions.	Yes	Yes
Service Catalog	Grants access to work with catalog-related file types such as catalog items, record producers, and variables to add catalog items to apps.	Yes	Yes
Tables & forms	Grants access to model and layout-related file types such as table columns, form layout, and list layout.	Yes	Yes
Decision Tables	Grants access to work with decision tables to create decision logic based on multiple if-else rules.	Yes	Yes
Playbooks	Grants access to work with the Playbooks design environment to create processes. Editing activity subflows or actions requires the **Flow Designer** permission.	Yes	Yes
Notifications	Grants access to create automatic email notifications and work with email templates.	Yes	Yes

Security/Entitlement custom collaboration permissions

Manage access control grants access to security management files, such as Access Control Lists and roles.

The default setting for both owners and editors is selected.

Programming tools custom collaboration permissions

Allow scripting grants access to script fields, such as those in Business Rule, UI Action, and Client Script.

The default setting for both owners and editors is de-selected.

Application management custom collaboration permissions

Application management collaboration permissions

Permission	Description	Owner default setting	Editor default setting
Delete application	Grants access to delete the app.	Yes	No
Manage collaborators	Grants access to change the developers and their permissions to collaborate on an app.	Yes	No
Source control	Grants access to work with source control tool integrations.	No	No
Invite collaborators	Grants access to invite developers to collaborate on an app.	Yes	Yes

Deployment custom collaboration permissions

Deployment collaboration permissions

Permission	Description	Owner default setting	Editor default setting
Upgrade app	Grants access to upgrade applications.	No	No
Submit for deployment	Grants access to request deployment for an app.	Yes	No
Publish app to repo	Grants access to publish the app to your repo.	No	No
Publish to app store	Grants access to publish the app to your app store.	No	No

Building pro-code applications

Create and manage custom applications from scratch using classic pro-code tools.

ServiceNow Studio

Use ServiceNow Studio to build apps and app files with integrated tools, access and edit metadata in scoped and global apps, and package app changes for deployment, all in one powerful development tool.

ServiceNow IDE

ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. With the ServiceNow IDE, you can create and develop scoped applications in source code, create JavaScript modules, and use third-party libraries.

ServiceNow SDK

ServiceNow SDK enables developers to create scoped applications in source code locally in Visual Studio Code Desktop and upload changes to a ServiceNow instance. You can also use the ServiceNow SDK to create JavaScript modules and use third-party libraries.

ServiceNow Extensions for Visual Studio Code

The ServiceNow extensions for VS Code editor enables you to edit applications offline and within ServiceNow instances.

Which builder should I use to create an app?

Types of builders

Want to build an app easily, without code?

Creator Studio specializes in helping you craft request-fulfillment applications without writing code. For example, an application to request office supplies by filling out a form, and someone approves or denies your request. For more information, see Exploring Creator Studio.

Need a more general app but still want low-code options?

App Engine Studio lets you build a broader range of apps than Creator Studio without being a programming pro. For more information, see Exploring App Engine Studio.

Are you a developer who wants more control in a centralized user interface?

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity. For more information, see Exploring ServiceNow Studio.

Are you a developer who wants to use industry-standard development tools and processes?

The ServiceNow IDE and ServiceNow SDK support developing applications in source code with ServiceNow Fluent, creating JavaScript modules, and using third-party libraries. ServiceNow Fluent is a domain-specific programming language for creating application metadata in code.

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. The ServiceNow SDK uses Visual Studio Code Desktop locally. For more information, see Building applications in source code.

Scripting

Action Designer

Use Workflow Studio to automate a repeatable task within a flow as a sequence of related steps. Enable flow authors to add actions to multiple flows with minimal configuration.

Client-side scripting

Client scripts enable the system to run JavaScript on the client (web browser) when client-based events occur, such as when a form loads, after form submission, or when a field changes value.

Now Assist for code generation

Now Assist for code generation allows developers to write scripts quickly with AI-generated code suggestions based on text or code prompts.

Now Code Editor

Now Code Editor provides a rich-text editor interface that supports Cascading Style Sheets (CSS), Hypertext Markup Language (HTML), JavaScript, Extensible Markup Language (XML), and JavaScript Object Notation (JSON). Use Now Code Editor to modify UI configuration, data resource configuration, styles, events, client-side and server-side scripts in Next Experience UI Builder components.

Server-side scripting

Server scripts run on the server or database. They can change the appearance or behavior of the ServiceNow AI Platform or run as business rules when records and tables are accessed or modified.

Core functionality

Groups

Simplify user administration by assigning roles to groups. Any business rules, assignment rules, system roles, or attributes that refer to the group apply to all group members automatically.

Notifications

Use ServiceNow Notifications to manage system email, create system notifications, and configure how your system responds to inbound email.

Roles

Roles control access to features and capabilities in applications and modules. The admin role provides access to all features and capabilities.

ServiceNow plugins

Plugins are software components that provide specific features and functionalities within a ServiceNow instance.

UI policies

UI policies dynamically change the behavior of information on a form and control custom process flows for tasks.

Users

User records establish a relationship between an individual and your ServiceNow instance. User records consist of a user name, a password, and information relating to the individual, such as contact information, location, and job title.

Related applications and features

JavaScript APIs

Use JavaScript APIs in scripts that you write to change the functionality of applications, or when you create new applications.

UI Builder

Use UI Builder to build web user interfaces for CSM Configurable Workspace, App Engine Studio generated workspaces and portals, or custom web experiences using Next Experience Components and custom web components.

Building applications in source code

Create and develop custom applications in source code using familiar development tools and processes.

Overview of development in source code

You can create scoped applications in code using the ServiceNow IDE on the ServiceNow AI Platform or locally in Visual Studio Code Desktop with the ServiceNow SDK.

In either development environment, you use ServiceNow Fluent, a domain-specific programming language, to define the metadata that makes up applications. ServiceNow Fluent includes APIs for defining the different types of metadata.

With the ServiceNow IDE or ServiceNow SDK, you can also create JavaScript modules and use third-party libraries in your application to optimize code reuse in scripts within a scoped application.

Developing and maintaining applications in source code enables you to work in familiar development environments, create and modify complex applications, manage code in source control more easily, and catch errors at build time.

Note:
The ServiceNow IDE and ServiceNow SDK don't support the Global scope or global applications in the Xanadu release.

ServiceNow IDE

Develop applications in code in an IDE on the ServiceNow AI Platform.

ServiceNow SDK

Develop applications in code locally and upload changes to an instance.

Comparison of the ServiceNow IDE and the ServiceNow SDK

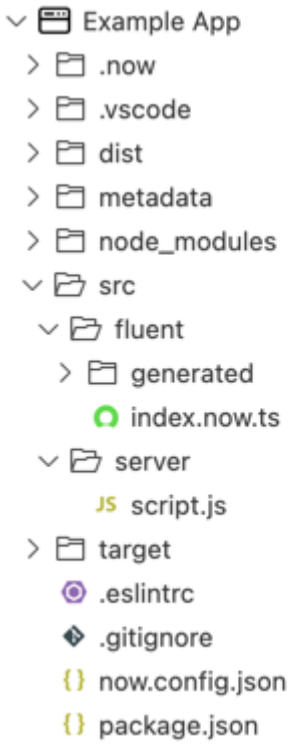
Feature	ServiceNow IDE	ServiceNow SDK
Development environment	Online ServiceNow instance	Local development and the ability to work offline
User interface	IDE based on Visual Studio Code for the Web	Visual Studio Code Desktop
Collaboration	View any user's changes in real time in code or embedded ServiceNow AI Platform user interfaces from the Metadata Explorer. Collaborate with other developers on applications in source control.	Download changes from an instance and install local changes to an instance using the ServiceNow SDK CLI to collaborate with other users. Collaborate with other developers on applications in source control.
Source control	Supports the most common Git functionality and integrating with a Git provider of your choice. One concurrent branch per repository per instance (or developer sandbox).	Full support
Application conversion	Support for converting existing scoped applications not created with the ServiceNow IDE or ServiceNow SDK.	Support for converting existing scoped applications not created with the ServiceNow IDE or ServiceNow SDK.
ServiceNow Fluent	Full support The ServiceNow Fluent Language server is included with the ServiceNow IDE.	Full support The ServiceNow Fluent Language server can be installed from the Visual Studio Code Extension Marketplace.

Feature	ServiceNow IDE	ServiceNow SDK
JavaScript modules and third-party libraries	Full support	Full support
Now Assist for Code	Code auto-complete support for scripts. For information about activation, see Now Assist for code generation.	Not supported

Application structure

Custom scoped applications created with the ServiceNow IDE or ServiceNow SDK include source code files and metadata XML files. The `package.json` and `now.config.json` files define the application structure, which is similar to that of Node.js applications or Node Package Manager (npm) packages.

Default structure of an application created in the ServiceNow IDE



By default, applications include the following directories and files. You can modify certain aspects of the application structure to suit your needs in the `now.config.json` file.

.vscode

Directory containing recommended Visual Studio Code extensions.

dist

Directory containing the build artifacts for packaging. This directory includes the following subdirectories:

- `app`: Directory containing the built metadata XML files.
- `static`: Directory containing the built static asset files.

metadata

Directory containing the application metadata (XML) of the application, such as table schemas and business rules, organized in the same directory structure as existing ServiceNow applications.

> **Note:**
> Application metadata shouldn't be edited from the XML files. Edit application metadata in the source code or on the ServiceNow AI Platform.

node_modules

Directory containing the third-party Node.js modules on which your application depends.

src

Directory containing the source code of your application. This directory includes the following subdirectories:

- `client`: Directory containing the client-side files for developing user interfaces.
- `fluent`: Directory containing ServiceNow Fluent code in `.now.ts` files. The `generated` subdirectory contains the application files converted to ServiceNow Fluent.
- `server`: Directory containing JavaScript module code in `.js` or `.ts` files.

target

Directory containing an installable package (`.zip` file) to upload to an instance.

.eslintrc.json

File containing the ESLint configuration. ESLint helps identify and fix issues in the application code.

.gitignore

File containing a list of directories or files for Git to ignore. These files aren't tracked in source control.

now.config.json

File containing the ServiceNow application configuration. The `now.config.json` file must be in the base directory for an application.

You can configure the directory structure for an application by adding the following parameters. For example:

```
{
   "scope" :  "x_snc_example_app",
   "scopeId" :  "2f8400eb07426110f736e28f69d3017a",
   "name" :  "ExampleApp",
   "dependencies": {
  "global": {
   "tables": ["incident"],
   "roles": ["admin"],
  },
  "x_custom": {
   "tables": ["custom_table"]
  }
 },
   "metadataDir" :  "metadata",
   "fluentDir" :  "src/fluent",
   "generatedDir" :  "generated",
   "serverModulesDir" :  "src/server",
```

```
  "clientDir": "src/client",
  "appOutputDir": "dist/app",
  "staticContentDir": "dist/static",
  "packOutputDir": "target",
  "modulePaths": {
        "src/server/*.ts": "dist/server/*.js",
    },
  "staticContentPaths": {
        "src/client/*.html": "dist/static/*.html",
    },
  "ignoreTransformTableList":
 ["ua_table_licensing_config", "sys_embedded_help_role"],
  "taxonomy":{
   "mapping":{
     "sys_script": "scripts/server/rules",
     "custom_table": "my-custom-folder/my-nested-folder"
   },
   "fallbackFolderName": "unclassified"
  }
}
```

Supported now.config.json parameters

Parameter	Description
dependencies	The items in another application scope on which your application depends. You must specify the application scope and the dependency type and names or sys_ids. `"dependencies": { "<scope>": { "<type>": ["<sys_id or name>"], ... }, ... }` For more information, see Download ServiceNow Fluent application dependencies.
metadataDir	Directory containing the application's metadata as XML files. Default: `metadata`
fluentDir	Directory containing ServiceNow Fluent files (`.now.ts`) that define application metadata in source code. Default: `src/fluent`
generatedDir	Directory containing generated ServiceNow Fluent files, including existing application metadata converted into ServiceNow Fluent code. This directory is relative to the directory defined with the *`fluentDir`* parameter. Default: `generated`
serverModulesDir	Directory containing the JavaScript or TypeScript files to be built into JavaScript modules for use in server-side scripts.

Supported now.config.json parameters (continued)

Parameter	Description
	Default: `src/server`
clientDir	Directory containing the client-side files for developing user interfaces with React. Default: `src/client`
appOutputDir	Directory to output the build artifacts to for packaging. The pack and install commands refer to this directory package the artifacts. Default: `dist/app`
staticContentDir	Directory to output the static asset files used for developing user interfaces. Default: `dist/static`
packOutputDir	Directory to output the installable package (`.zip` file) when building the application. The install command refers to this directory to install the package. Default: `target`
serverModulesIncludePatterns	A list of file patterns to include when building JavaScript modules. Default: `[ "**/*.ts", "**/*.tsx", "**/*.js", "**/*.jsx", "**/*.cts", "**/*.cjs", "**/*.mts", "**/*.mjs", "**/*.json" ]`
serverModulesExcludePatterns	A list of file patterns to exclude when building JavaScript modules. Default: `[ "**/*.test.ts", "**/*.test.js", "**/*.d.ts" ]`

Supported now.config.json parameters (continued)

Parameter	Description
trustedModules	A list of npm packages to identify as trusted (or internal). Trusted modules have access to ServiceNow APIs. For example: `"trustedModules": [ "<package-name>", // Specific package "@servicenow/*" // All packages from an organization ]` In the EcmaScript Module [sys_module] table, the **External source** field is set to false for trusted modules. **⚠ Warning:** Only add dependencies that you trust completely as trusted modules. Valid patterns: • Fully qualified package names, such as `'@servicenow/sdk'`. • Organization prefixes with a wildcard, such as `'@servicenow/*'` or `'@mycompany/*'`.
modulePaths	A map of the module source files to their equivalent output files for if you use a custom transpilation step before building the application. For more information, see Using TypeScript in JavaScript modules with the ServiceNow SDK. **⚠ Warning:** You can't use this parameter and the `tsconfigPath` parameter. Configuring both results in an error.
staticContentPaths	A map of the client-side source files to the output paths for static asset files.
tsconfigPath	A path to a `tsconfig.json` file with custom options for transpiling TypeScript into JavaScript during the build process. Specifying a `tsconfigPath` generates diagnostic results from TypeScript using the `tsconfig.json` file. **⚠ Warning:** You can't use this parameter and the `modulePaths` parameter. Configuring both results in an error. Default: .
ignoreTransformTableList	A list of tables to ignore when transforming application metadata into source code.
tableDefaultLanguage	The BCP 47 code of a default language for field labels [sys_documentation] in a table or column. Th default language is used to resolve field labels with multiple languages.

Supported now.config.json parameters (continued)

Parameter	Description
	Default: en
tableOutputFormat	The type of build artifacts for table metadata XML generated from ServiceNow Fluent code. Valid values: • bootstrap: The build process outputs a bootstrap XML file with the `<database>` root element for the table, field label XML files [sys_documentation], licensing configuration XML files [ua_table_licensing_config], and auto-numbering XML files [sys_number]. • component: The build process outputs XML files for each component of the Table API. Default: `bootstrap`
taxonomy	A configuration for organizing generated ServiceNow Fluent files, which maps table names to directories and defines a fallback directory. The default taxonomy configuration uses ServiceNow standard table classifications to add generated ServiceNow Fluent files in a logical directory structure within the `fluent/generated` directory when metadata is initially transformed into ServiceNow Fluent code. For example: • Business rules [sys_script] are added to the `fluent/generated/server-development/business-rule` directory. • Script includes [sys_script_include] are added to the `fluent/generated/server-development/script-include` directory. You can override the default mappings or configure additional ones. In the following example, the configuration overrides the default directory for business rules [sys_script] and the fallback folder and configures an additional mapping for metadata from a custom table. `"taxonomy": { "mapping": { "sys_script": "scripts/server/rules", "custom_table": "my-custom-folder/my-nested-folder" }, "fallbackFolderName": "unclassified" }` • mapping: An object that maps table names to directories. Directory paths are relative to the directory configured with the *`generatedDir`* parameter and must include only lowercase letters, numbers, hyphens, underscores, and slashes to separate subdirectories.

Supported now.config.json parameters (continued)

Parameter	Description
	• fallbackFolderName: A name for a directory to use for tables that don't have a default or a custom mapping configured. Within this directory, ServiceNow Fluent files are added to subdirectories named after the table, such as `src/fluent/generated/other/x-unmapped-table/`. Default: Default taxonomy mappings are defined for all standard ServiceNow tables and are automatically applied when no custom configuration is defined in the `now.config.json` for an application. The default value of *`fallbackFolderName`* is `other`.

now.prebuild.mjs

Auto-updated file containing complete information about dependencies and their versions. This file is only available with the ServiceNow SDK.

package-lock.json

Auto-updated file containing complete information about dependencies and their versions. This file is only available with the ServiceNow SDK.

package.json

File containing information about your application and custom or third-party module dependencies. The `package.json` file must be in the base directory for an application. On an instance, the `package.json` path is specified in the **Package JSON** field of the custom application record [sys_app] in the format `<scope>/<package-name>/<version>/package.json`.

Related applications and features

JavaScript APIs

Use JavaScript APIs in scripts that you write to change the functionality of applications or when you create applications.

Related topics

ServiceNow Fluent

JavaScript modules and third-party libraries

User interface development with React

ServiceNow Fluent

Define application metadata in source code using the ServiceNow Fluent domain-specific programming language.

Overview of ServiceNow Fluent

ServiceNow Fluent is a domain-specific language (DSL) based on TypeScript for defining the metadata files [sys_metadata] that make up applications and includes APIs for the different types of metadata, such as tables, roles, ACLs, business rules, and Automated Test Framework tests.

Developers define this metadata in a few lines of code instead of through a form or builder tool user interface. Applications created or converted with the ServiceNow IDE or ServiceNow SDK support development in ServiceNow Fluent.

ServiceNow Fluent supports two-way synchronization, which allows changes to metadata to be synced from other ServiceNow AI Platform user interfaces into source code and changes to source code to be synced back to metadata across the instance.

To get started using the ServiceNow IDE or ServiceNow SDK, see the ServiceNow IDE or ServiceNow SDK documentation.

ServiceNow Fluent APIs

ServiceNow Fluent includes APIs for the following types of metadata. You can use the Record API to define application metadata that doesn't have a dedicated API.

For details about the APIs and examples, see ServiceNow Fluent API reference and the ServiceNow SDK examples GitHub repository.

- Access control lists (ACLs)
- Application menus
- Automated Test Framework tests
- Business rules
- Client scripts
- Cross-scope privileges
- Dashboards
- Email notifications
- Flows
- Import sets
- Lists
- Properties
- Records
- Roles
- Script actions
- Script includes
- Scripted REST APIs
- Service level agreements (SLAs)
- Service portal widgets
- Tables
- UI actions
- UI pages
- UI policies
- Workspaces

Note:
A limited number of metadata types, such as Metadata Snapshots [sys_metadata_link] and UX Assets [sys_ux_lib_asset], can't be represented as ServiceNow Fluent code and aren't transformed. These metadata types remain as metadata XML files in the `metadata` directory of your application.

ServiceNow Fluent usage

In files with the `.now.ts` extension, use objects in the ServiceNow Fluent APIs to define metadata in the application. You must also include the required imports for the APIs from `@servicenow/sdk/core`. For objects with server-side scripts, such as the *`BusinessRule`* object, you can import and use code from JavaScript modules.

The following example includes the definitions of a table, client script, and business rule in the application. The client script uses a script from the `client-script.js` file. The business rule uses a function from the `script.js` JavaScript module.

```
import '@servicenow/sdk/global'
import { BusinessRule, ClientScript, DateColumn, StringColumn,
 Table } from '@servicenow/sdk/core'
import { showStateUpdate } from '../server/script.js'

//creates todo table, with three columns (deadline, status and task)
export const x_snc_example_to_do = Table({
    name: 'x_snc_example_to_do',
    schema: {
        deadline: DateColumn({ label: 'Deadline' }),
        state: StringColumn({
            label: 'State',
            choices: {
                ready: { label: 'Ready' },
                completed: { label: 'Completed' },
                inProgress: { label: 'In Progress' },
            },
        }),
        task: StringColumn({ label: 'Task', maxLength: 120 }),
    },
})

//creates a client script that pops up 'Table loaded successfully!!' message everytime todo
 record is loaded
ClientScript({
    $id: Now.ID['cs0'],
    name: 'my_client_script',
    table: 'x_snc_example_to_do',
    active: true,
    appliesExtended: false,
    global: true,
    uiType: 'all',
    description: 'Custom client script generated by Now SDK',
    isolateScript: false,
    type: 'onLoad',
    script: Now.include('../client/client-script.js'),
})

//creates a business rule that pops up state change message whenever a todo record is
 updated
```

```
BusinessRule({
    $id: Now.ID['br0'],
    action: ['update'],
    table: 'x_snc_example_to_do',
    script: showStateUpdate,
    name: 'LogStateChange',
    order: 100,
    when: 'after',
    active: true,
})
```

The client script referenced from the *ClientScript* object:

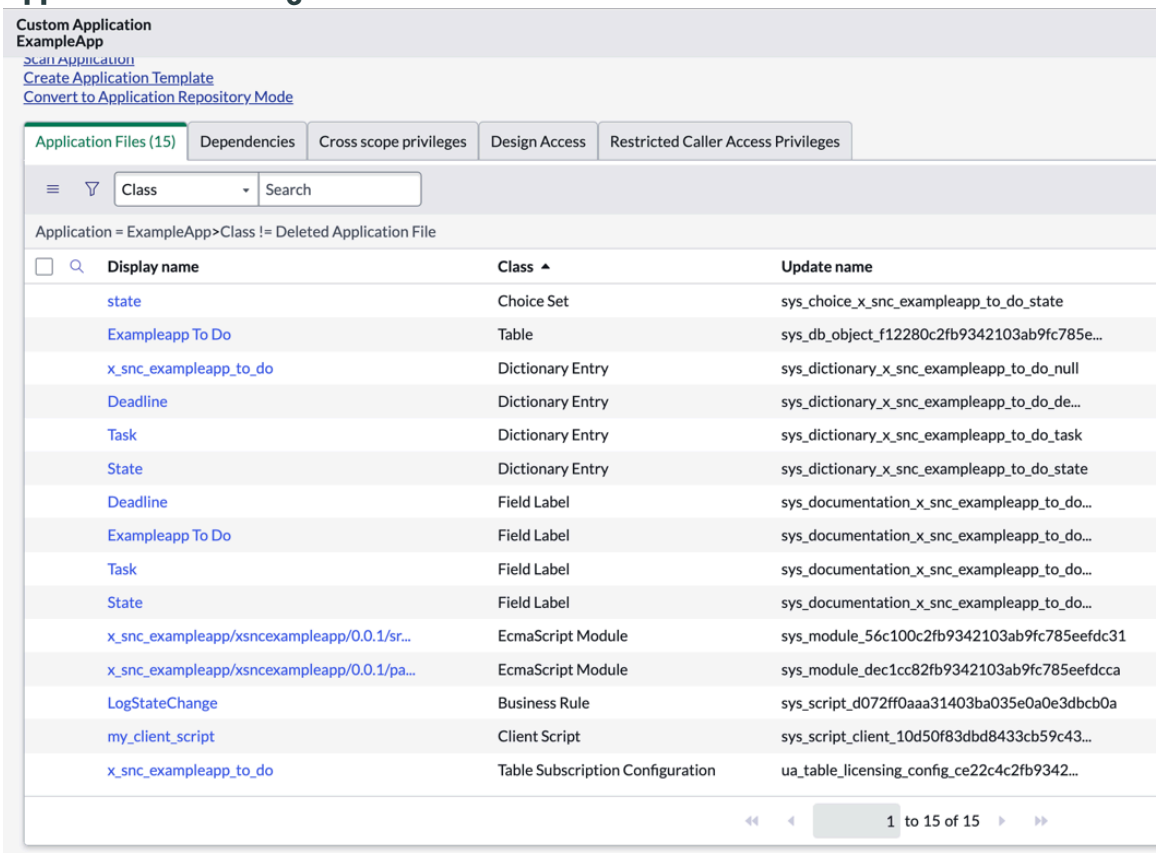
```
function onLoad() {
    g_form.addInfoMessage("Table loaded successfully!!")
}
```

The JavaScript module referenced from the *BusinessRule* object:

```
import { gs } from '@servicenow/glide'

export function showStateUpdate(current, previous) {
    const currentState = current.getValue('state')
    const previousState = previous.getValue('state')

    gs.addInfoMessage(`state updated from "${previousState}"
 to "${currentState}"`)
}
```

After building the application, this source code generates the following application metadata files on the instance.

Application metadata generated from ServiceNow Fluent code



Tip:
You can use the following directives in a code comment to help manage your code:

- `@fluent-ignore`: Suppresses ServiceNow Fluent diagnostic warnings and errors in the following line of code.
- `@fluent-disable-sync`: Turns off syncing changes to a ServiceNow Fluent object. Use before a call expression (for example, `Record({ ... })`) to turn off syncing for that object and its child objects. Only use this directive if you want to ignore changes made outside of the source code to the object and never update it when syncing.
- `@fluent-disable-sync-for-file`: Turns off syncing changes to a ServiceNow Fluent file (`.now.ts`). Use in the first line of the file to turn off syncing for all code in the file. Only use this directive if you want to ignore changes made outside of the source code to the file and never update it when syncing.

Related topics

ServiceNow Fluent API reference

Define application metadata in code with ServiceNow Fluent in the ServiceNow IDE

Define application metadata in code with ServiceNow Fluent and the ServiceNow SDK

JavaScript modules and third-party libraries

Optimize your code base using JavaScript modules to group related code or add third-party libraries and reuse their code within applications.

Overview of using JavaScript modules

A module is a JavaScript file that contains related code that's shared and reused within applications on an instance. You can add JavaScript modules and third-party libraries in scoped

applications that are created or converted with the ServiceNow IDE or ServiceNow SDK. You can also use TypeScript to create modules and compile them to JavaScript before building your application. After installing an application on an instance, JavaScript modules are stored in the EcmaScript Module [sys_module] table.

In a module, you identify code for reuse with `export` statements. Then, use `import` or `require` statements to reuse the code elsewhere in your applications. You must add third-party Node Package Manager (npm) libraries to applications as dependencies to use their module code. For general information about the syntax used to create JavaScript modules, see the JavaScript modules page on the MDN Web Docs website.

Note:
To use global Glide Server APIs in modules, they must be imported from the `@servicenow/glide` package. For more information, see Importing server APIs.

Limitations

- Global applications and application customizations aren't supported with instances on the Xanadu release.
- Modules can be used only within the application scope in which they're added. They can't be used across application scopes.
- A subset of ECMAScript features are supported in modules in accordance with the JavaScript engine feature support.
- Node.js APIs aren’t supported in modules. The ServiceNow SDK build process polyfills any Node.js built-in modules while packaging modules, otherwise modules are resolved from the `node_modules` directory.
- Global variables related to web APIs aren’t supported.
- CommonJS modules from third-party libraries aren't supported unless they define exports. Subpath imports aren't supported with CommonJS modules. ECMAScript modules from third-party libraries are supported.
- `import` and `export` statements are only supported in modules. To import module code in scripts, such as business rules or script includes, use `require` statements.
- JavaScript modules [sys_module] can be modified only in the ServiceNow IDE or in Visual Studio Code with the ServiceNow SDK.

Important:
You can't use third-party libraries that rely on unsupported functionality, such as unsupported APIs or ECMAScript features.

Exporting modules

In a module, identify code for reuse with `export` statements. You can use named exports or default exports. Named exports can be for variables, constants, functions, or classes whereas default exports can be for functions or classes only. The following example is one way of adding a named export for multiple features (a function and a variable) in a module:

```
export { myFunction, myVariable };
```

Importing modules

To import the module code you want to reuse, use `import` statements in other modules or `require` statements in server-side scripts.

The following example is one way that you could import an exported feature in a module:

```
import { feature } from 'path/to/module';
```

The following example is one way that you could import an exported feature in a script:

```
const { feature } = require('path/to/module');
```

Note:
To import code from one TypeScript file to another TypeScript file, you must include the `.ts` file extension. For example, `import { feature } from './module.ts'`.

To use shorthand to import module code, you can use subpaths in the *`imports`* field of the application's `package.json` file. For example:

```
{
 "name": "math",
 "version": "1.0.0",
 "exports": {
  "./functions/*.js": "./src/functions/*.js",
  "./functions/private-functions/*": null
 },
 "imports": {
  "#calc": "calculus",
  "#derivative": "calculus/derivative"
 },
 "dependencies": {
  "calculus": "1.0.0"
 }
}
```

Based on that example, instead of writing out the relative path to `derivative.js` every time you want to import it in the `math` application, you can use the `#derivative` shorthand instead. Subpaths can also be used in the *`imports`* field to use shorthand for dependencies, such as `#calc`.

```
import { derivative } from '#derivative';
import * as calculus from '#calc';
```

Adding third-party libraries

Applications must declare dependencies on third-party libraries to use their module code. In an application's `package.json` file, include the package name and version for any dependencies. For example, to use modules from the "math" library in the "test" application, add the "math" package as a dependency:

```
{
 "name": "test",
 "version": "1.0.0",
 "dependencies": {
  "math": "1.0.0"
 }
}
```

Importing server APIs

To import server APIs and use them in a module, use *`import`* statements. Glide APIs can be imported from the `@servicenow/glide` package or their namespace in the package.

For example:

```
import { API } from '@servicenow/glide';
import { API } from '@servicenow/glide/<namespace>';
```

In the following example, the *gs* (GlideSystem) and *GlideRecord* APIs are imported in a module:

```
import { gs } from '@servicenow/glide';
import { GlideRecord } from '@servicenow/glide';
```

In the following example, the *RESTAPIRequest* and *RESTAPIResponse* APIs are imported from the `sn_ws_int` namespace in a module because they run in that namespace:

```
import { RESTAPIRequest, RESTAPIResponse }
 from '@servicenow/glide/sn_ws_int';
```

To access server APIs in a third-party library module, you must add the module as a trusted module with the *trustedModules* parameter in your application's `now.config.json` file. For more information, see Supported now.config.json parameters.

For more information about available server APIs, see Server API reference.

Importing script includes

To import script includes and use them in a module, use *import* statements. Script includes can be imported from their application scope or the global scope in the `@servicenow/glide` package.

For example:

```
import { global } from '@servicenow/glide/global';
import { ScriptInclude } from '@servicenow/glide/<scope>';
```

For more information about script includes, see Script includes.

Related topics

- Create and use JavaScript modules in applications in the ServiceNow IDE
- Create and use JavaScript modules in applications with the ServiceNow SDK

User interface development with React

Develop a user interface (UI) with the React library to build a full-stack application in source code.

Overview of UI development with React

React is an industry-standard web framework for building UI components that you use to develop and render custom user interfaces. A React component is a self-contained, reusable JavaScript function, such as a button, and supports JavaScript (JS/JSX) and TypeScript (TS/TSX) syntax. For general information about React, see the React documentation on the react.dev website.

With the ServiceNow IDE or ServiceNow SDK, you can use React in your scoped application to create a UI page in ServiceNow Fluent code. The ServiceNow Fluent UI Page API refers to an HTML entry point (`index.html`) that loads the page at the endpoint provided. After building and installing the application on an instance, the static assets are stored in the appropriate tables. For an example of a React application in source code, see the ServiceNow SDK examples GitHub repository. To get started using React, select one of the React templates when creating an application with the ServiceNow IDE or ServiceNow SDK.

Note:
You can use some Next Experience Components in a React application with the React Wrapper Component Library Node Package Manager (npm) package on the public npm registry. To use the React Wrapper Component Library in an application, you can use Build Agent or you must install it from the ServiceNow IDE or ServiceNow SDK. For more information, see Use third-party libraries with the ServiceNow IDE or Use third-party libraries with the ServiceNow SDK.

UI development process

The following list is a high-level overview of the process to develop a React application with the ServiceNow IDE or ServiceNow SDK:

1. Create an application and select one of the React templates.

   The application includes the files and directories required for React development.

2. In the `src/client` directory, add client assets for a user interface, such as an HTML entry point (`index.html`), client scripts, and style sheets. The `index.html` file must contain the `<script>` tag as the JavaScript entry point.
3. Define a UI page [sys_ui_page] in code with the ServiceNow Fluent UI Page API. In the `now.ts` file that contains the UI page definition, you import the HTML and refer to it from the *`html`* property of the *`UiPage`* object.
4. Build the application to execute a pre-build Rollup script that bundles dependencies and packages the client assets in the `src/client` directory before executing the rest of the build process. The standard Rollup build process and plugins are used as the default JavaScript bundler.

   Static assets are output to the `dist/static` directory.

5. Install the application to add the static assets to the application files [sys_metadata] in the appropriate tables: UX Asset [sys_ux_lib_asset], Images [db_image], and UX Theme Asset [sys_ux_theme_asset] tables. These tables support adding files as attachments.

Important:
UI development with React is an experimental feature due to how content is served from an instance and the content types and tables that are currently supported.

Limitations

- The UI page HTML should be modified only in source code. Changes to the HTML of a UI page [sys_ui_page] on an instance aren't synchronized into source code and are likely to result in unintended behavior.
- Audio, video, and WASM content types aren't supported.
- The maximum file size of uploaded attachments is limited by the size configured with *`com.glide.attachment.max_size`* system property. For more information, see Maximum allowed attachment size [Updated in Security Center 1.3].
- Output paths must be deterministic.
- Pre-loading content linked from HTML isn't supported (`rel="preload"`).
- Relative style sheets linked from HTML aren't supported (`rel="stylesheet"`). Import style sheets into code instead (`@import 'path/to/style-sheet'`).
- Relative imports in CSS aren't supported.
- CSS modules aren't supported.

- Only hash routing is supported by UI pages.
- Server-side rendering and React server components aren't supported.

Example: UI page development with React

In a `.now.ts` file in the `src/fluent` directory, a UI page is defined in ServiceNow Fluent code. The HTML of the page is imported from the `index.html` file in the `src/client` directory.

```
//incident-manager.now.ts
import '@servicenow/sdk/global'
import { UiPage } from '@servicenow/sdk/core'
import incidentPage from '../../client/index.html'

UiPage({
    $id: Now.ID['incident-manager-page'],
    endpoint: '<scope>_incident_manager.do',
    description: 'Incident Response Manager UI Page',
    category: 'general',
    html: incidentPage,
    direct: true,
})
```

In the `index.html` file, the `<script>` tag refers to the `main.jsx` file in the `src/client` directory.

```
//index.html
<html>
<head>
  <title>Incident Response Manager</title>

  <!-- Initialize globals and Include ServiceNow's required scripts -->
  <sdk:now-ux-globals></sdk:now-ux-globals>

  <!-- Include your React entry point -->
  <script src="./main.jsx" type="module"></script>
</head>
<body>
  <div id="root"></div>
</body>
</html>
```

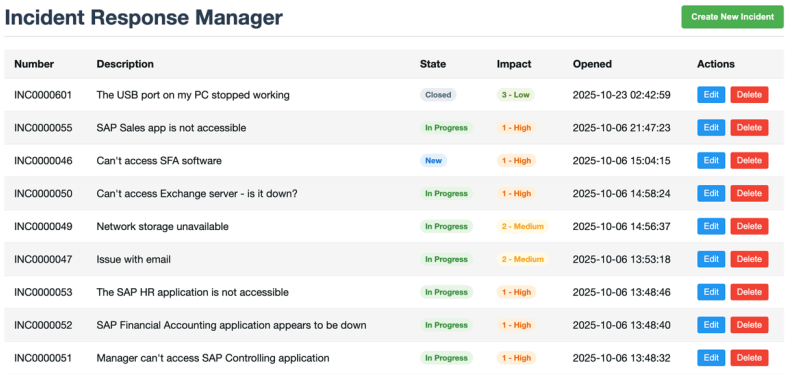
In the `main.jsx` file, the imports required for rendering the page and the main application component are included and the entry point of the React application is defined.

```
//main.jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './app'

const rootElement = document.getElementById('root')
if (rootElement) {
    ReactDOM.createRoot(rootElement).render(
        <React.StrictMode>
            <App />
        </React.StrictMode>
```

```
        )
}
```

After building and installing the application, you can open the page from the endpoint provided in the *UiPage* object (`https://<instance>/<scope>_incident_manager.do`), which displays an interface for creating and managing incidents.

Example of a UI page created with React



Related topics

- UI Page API - ServiceNow Fluent
- ServiceNow Fluent

ServiceNow IDE

Create and develop scoped applications in source code in an integrated development environment (IDE) on the ServiceNow AI Platform to improve collaboration across development teams and accelerate application development.

https://player.vimeo.com/video/1129991964?h=6929aca67a&badge=0&autopause=0&player_id=0&app_id=58479

Note:
The ServiceNow IDE doesn't support the Global scope or global applications in the Xanadu release.

Get started

Explore	Configure	Create
		
Learn about developing applications in the ServiceNow IDE.	Install and configure the ServiceNow IDE for developers to use.	Create workspaces and add applications.

Integrate	Develop	Reference
		
Integrate with source control providers.	Develop applications in source code.	Get details about ServiceNow Fluent APIs and more.

Which builder should I use to create an app?

Are you a developer who wants to use industry-standard development tools and processes?

The ServiceNow IDE and ServiceNow SDK support developing applications in source code with ServiceNow Fluent, creating JavaScript modules, and using third-party libraries. ServiceNow Fluent is a domain-specific programming language for creating application metadata in code.

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. The ServiceNow SDK uses Visual Studio Code Desktop locally. For more information, see Building applications in source code.

Are you a developer who wants more control in a centralized user interface?

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity. For more information, see Exploring ServiceNow Studio.

Need a more general app but still want low-code options?

App Engine Studio lets you build a broader range of apps than Creator Studio without being a programming pro. For more information, see Exploring App Engine Studio.

Want to build an app easily, without code?

Creator Studio specializes in helping you craft request-fulfillment applications without writing code. For example, an application to request office supplies by filling out a form, and someone approves or denies your request. For more information, see Exploring Creator Studio.

Troubleshoot and get help

- To learn more about what's new in the Xanadu release, see the ServiceNow IDE release notes.
- Introduction to the ServiceNow IDE on ServiceNow University
- ServiceNow IDE Demo video
- ServiceNow IDE and Fluent Creator Toolbox video
- ServiceNow SDK examples GitHub repository
- ServiceNow IDE, SDK, and Fluent forum in the ServiceNow Community

- ServiceNow IDE, SDK, and Fluent articles in the ServiceNow Community
- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

Related topics

ServiceNow Fluent

JavaScript modules and third-party libraries

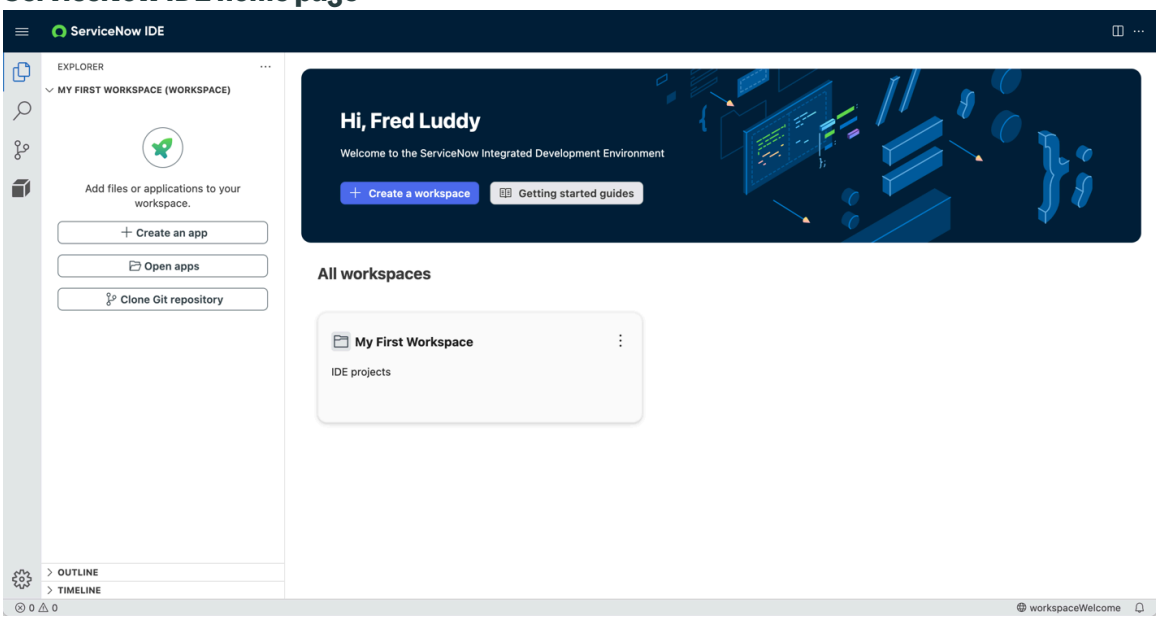
Exploring the ServiceNow IDE

Learn about developing scoped applications in source code in the ServiceNow IDE.

ServiceNow IDE overview

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. With the ServiceNow IDE, you can get started building scoped applications quickly using familiar tools and industry-standard development practices. The ServiceNow IDE has many of the same features as Visual Studio Code, including type safety, IntelliSense, dependency enforcement, code search, and source control integration.

ServiceNow IDE home page



In the ServiceNow IDE, you can create scoped applications in source code using ServiceNow Fluent to define application metadata [sys_metadata]. To see a side-by-side visual representation of your changes in real time, you can open the application metadata in other ServiceNow AI Platform user interfaces embedded within the ServiceNow IDE.

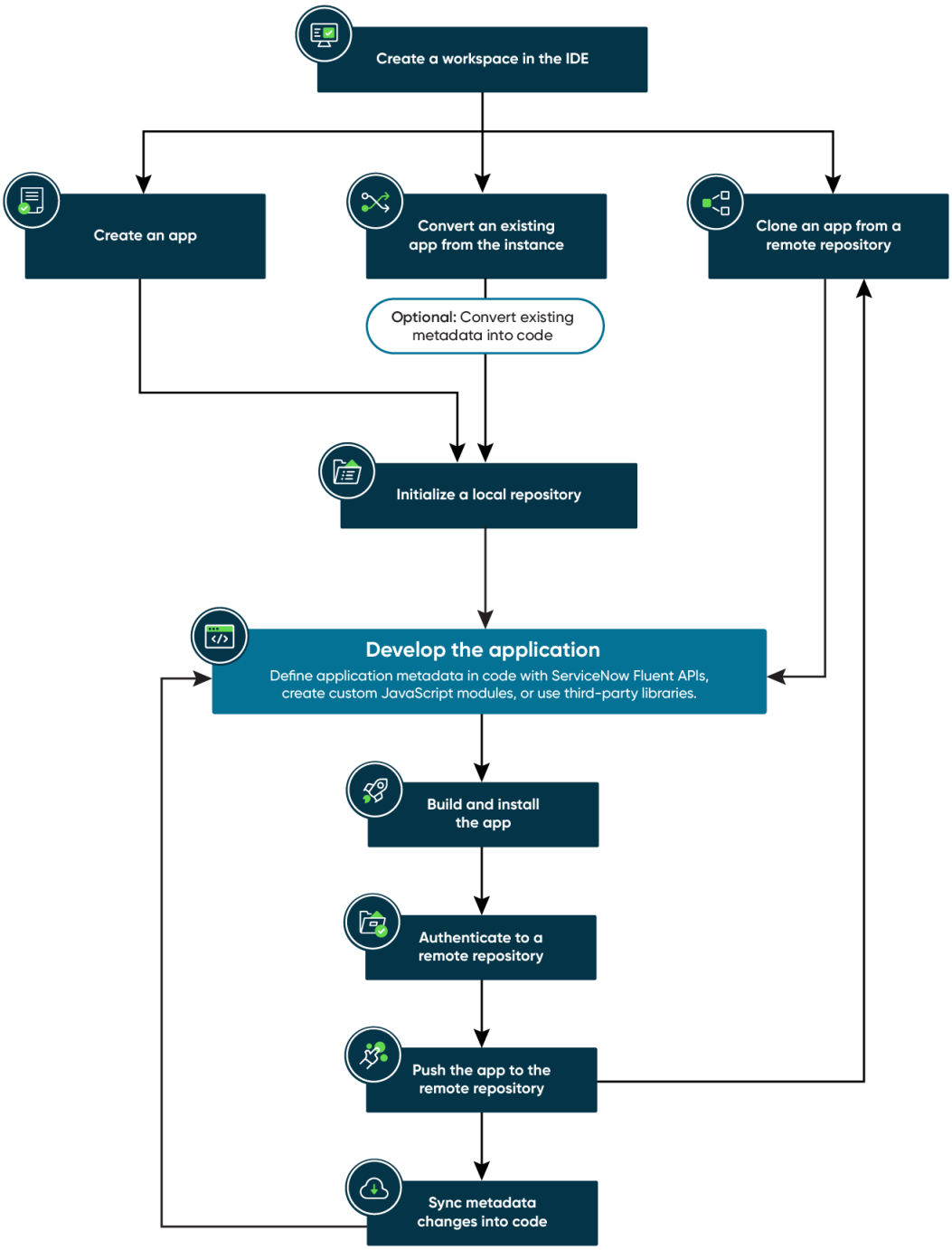
Optionally, you can create JavaScript modules and use third-party libraries to organize and reuse code within scoped applications.

In the background, this functionality is enabled by the ServiceNow SDK application packaging service, which builds applications in the ServiceNow IDE.

ServiceNow IDE workflow

The following infographic shows the workflow for developers to get started developing applications with the ServiceNow IDE.

Developing applications with the ServiceNow IDE



1. From the ServiceNow Store, an administrator installs the ServiceNow IDE application on a non-production instance.
2. Depending on the authentication method, an administrator or developer configures the credentials needed to connect to their Git provider.

- To use OAuth 2.0 authentication, an administrator configures an OAuth 2.0 application registry in their Git provider and on the instance.
- To use basic authentication, a developer generates a personal access token from their Git provider.

3. From their Git provider, a developer creates a dedicated Git repository for an application.
4. From the ServiceNow IDE, the developer configures their Git credentials.
5. The developer creates a ServiceNow IDE workspace to organize applications.
6. The developer creates a scoped application or converts an existing scoped application from the ServiceNow IDE.
7. The developer initializes a local Git repository for the application and pushes it to the remote repository they created to manage the application in source control.
8. The developer can define application metadata in source code using ServiceNow Fluent, create custom JavaScript modules, or use third-party libraries.
9. When metadata changes are detected, the developer synchronizes the application to download and transform changes to metadata from other interfaces into source code.
10. After saving their changes, they build the application, which compiles the source code and transforms it into application metadata.
11. When the build is complete, the developer can view their changes reflected in other embedded ServiceNow AI Platform user interfaces without leaving the ServiceNow IDE.
12. The developer stages, commits, and pushes their changes to the Git repository.
13. From the ServiceNow IDE, other developers can clone the repository, create branches, and begin collaborating on the application.

Other users can modify the application metadata at the same time as developers modify the source code. Developers can reuse module code in other modules or scripts within the application.

ServiceNow IDE benefits

Benefit	Feature	Users
Develop applications in an IDE based on Visual Studio Code on the ServiceNow AI Platform	ServiceNow IDE user interface	Developers
Write source code to define the metadata that makes up ServiceNow applications	ServiceNow Fluent Define application metadata in code with ServiceNow Fluent in the ServiceNow IDE	Developers
Organize and reuse code within scoped applications with custom JavaScript modules and third-party JavaScript utilities	JavaScript modules and third-party libraries Create and use JavaScript modules in applications in the ServiceNow IDE	Developers

Benefit	Feature	Users
	Use third-party libraries in applications in the ServiceNow IDE	
Collaborate on applications with users of different skill sets	Build and install an application in the ServiceNow IDE	Developers
Integrate source control with your Git provider of choice	Integrating source control with the ServiceNow IDE	Developers, Administrators

ServiceNow IDE user interface

The ServiceNow IDE user interface contains many of the same features as Visual Studio Code for the Web and some unique functionality for creating ServiceNow applications.

Overview of the ServiceNow IDE user interface

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform, which includes the following features:

- Basic editing and keyboard shortcuts
- IntelliSense
- Code navigation
- Code and file search
- Command palette
- Source control integration

Visual Studio Code for the Web is similar to Visual Studio Code Desktop but doesn't support all of the features that are available in the desktop interface. For information about browser-based implementations of Visual Studio Code, see Visual Studio Code for the Web. The following Visual Studio Code features aren't supported in the ServiceNow IDE:

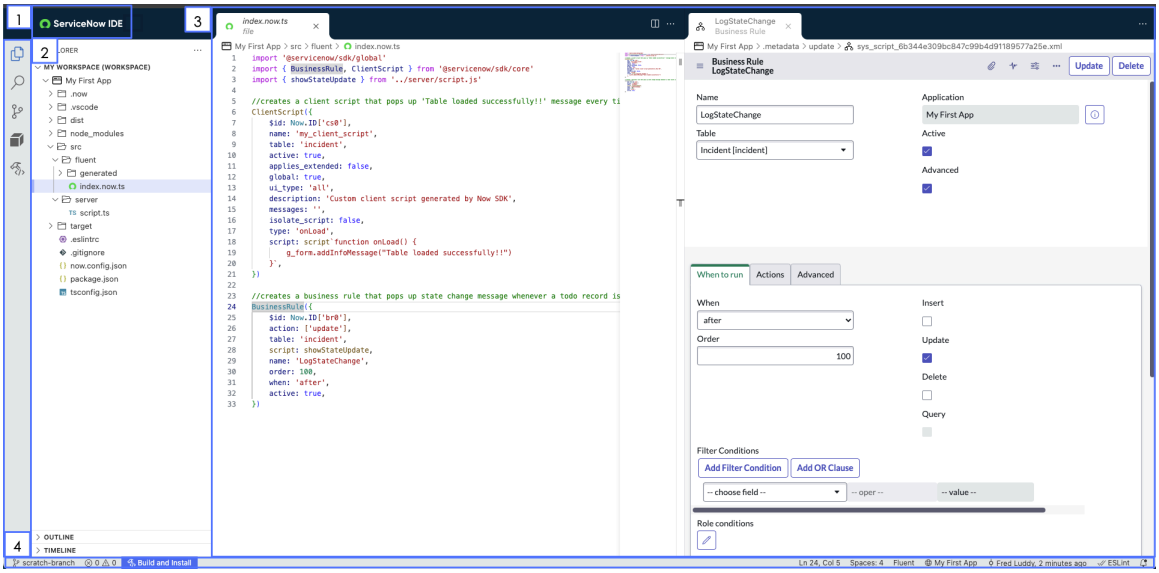
- Integrated terminal
- Debugger
- Marketplace extensions
- Integrated unit testing
- Remote development

For general information about common features in Visual Studio Code and the ServiceNow IDE, see the Visual Studio Code documentation website.

Features in the ServiceNow IDE

Learn about features of the ServiceNow IDE user interface.

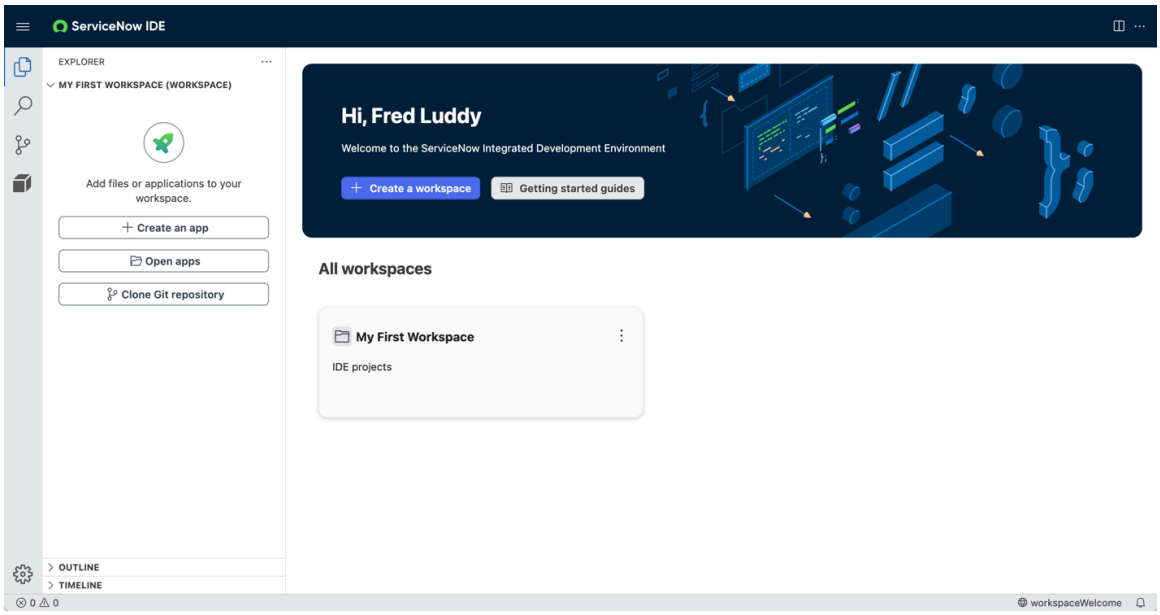
ServiceNow IDE user interface



1. ServiceNow IDE home page: Get started creating workspaces and applications and view existing workspaces.
2. Activity Bar
   - File Explorer view: Navigate through applications in your workspace and their code.
   - Search view: Find content within files.
   - Source Control view: Integrate with a Git repository for version control of your code.
   - Metadata Explorer view: Create, update, and view application metadata.
   - Now SDK view: Synchronize changes to application metadata or build and install applications.
   - Manage menu: Access and configure various aspects of the user interface.
3. Editor: Write source code and view application metadata through embedded ServiceNow AI Platform user interfaces.
4. Status bar: View status information about your applications and files and build applications.

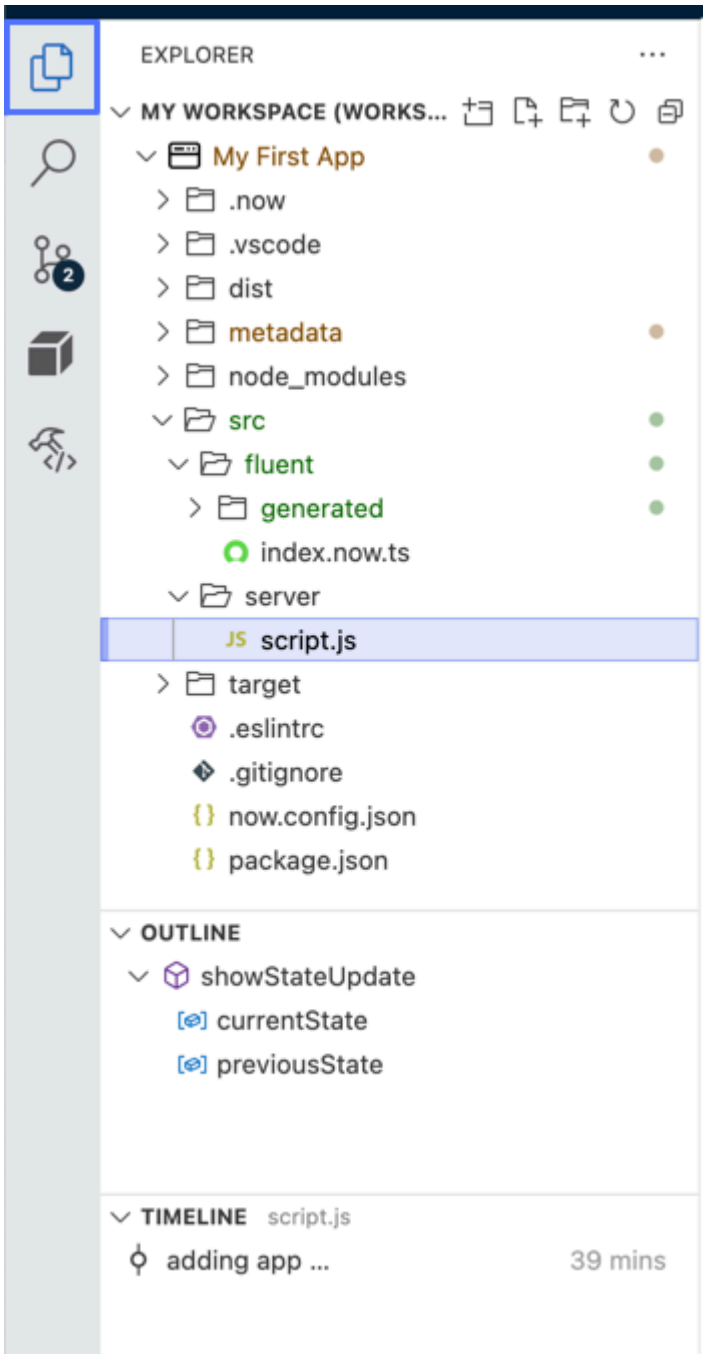
In addition to these features, you can open the command palette to run a variety of commands to develop applications, manage files, use source control, and more. For more information, see Command palette.

ServiceNow IDE home page



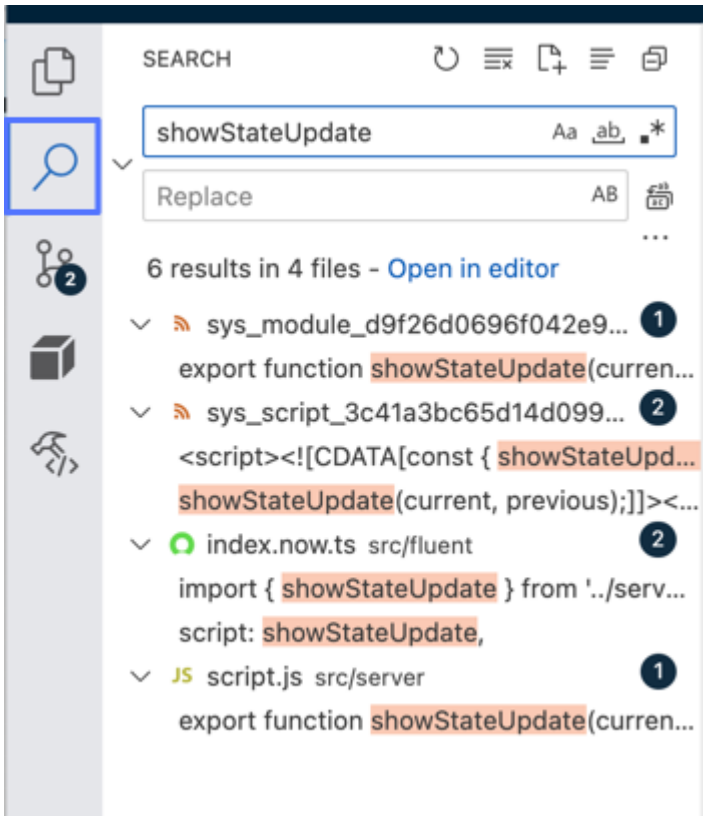
Create workspaces, access existing workspaces, and create or open applications. Clone a Git repository to add an existing application from a remote repository. To get started creating workspaces and applications, see Developing applications with the ServiceNow IDE. To clone a Git repository, see Clone a Git repository with the ServiceNow IDE.

File Explorer view



Navigate through applications in a workspace and add files or folders. In the Outline section, view the outline of the active file in the editor. In the Timeline section, view a history of changes to the active file. For information about the default application file structure, see the Application structure section of the Building applications in source code topic.

Search view

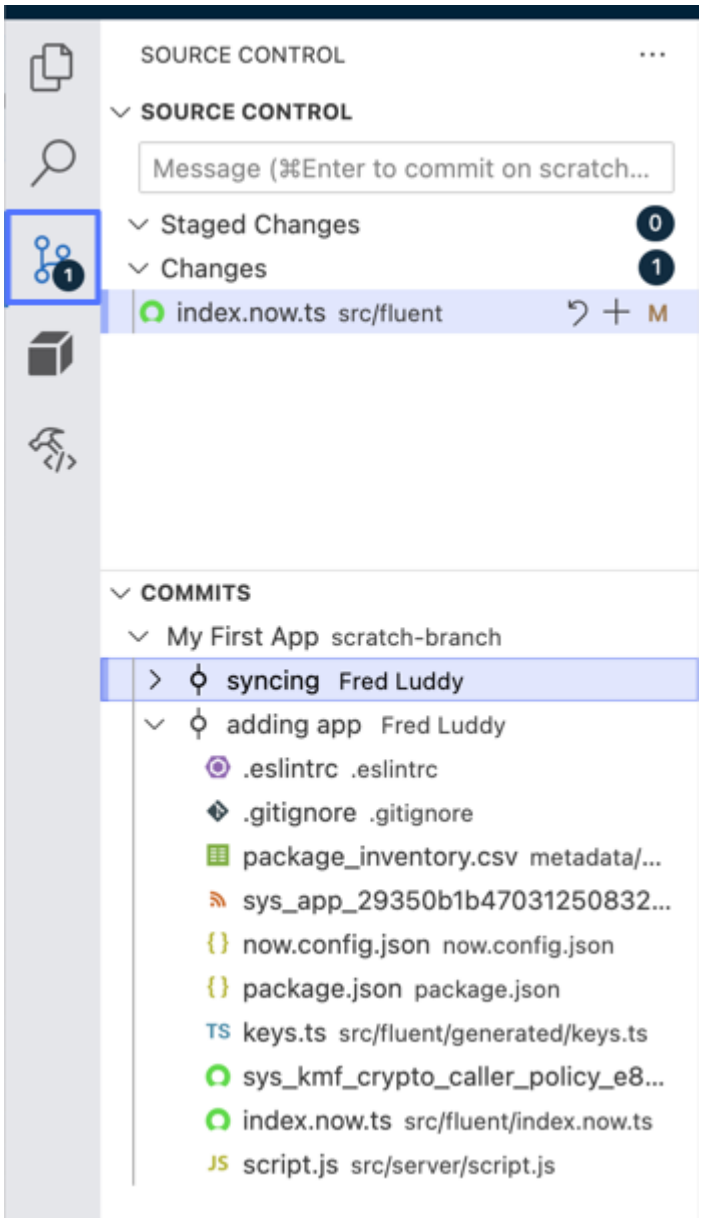


Search across files in the active workspace. The Search view supports using regular expressions, find and replace, and advanced search options.

To search for files by name, open the search box with the following keyboard shortcuts:

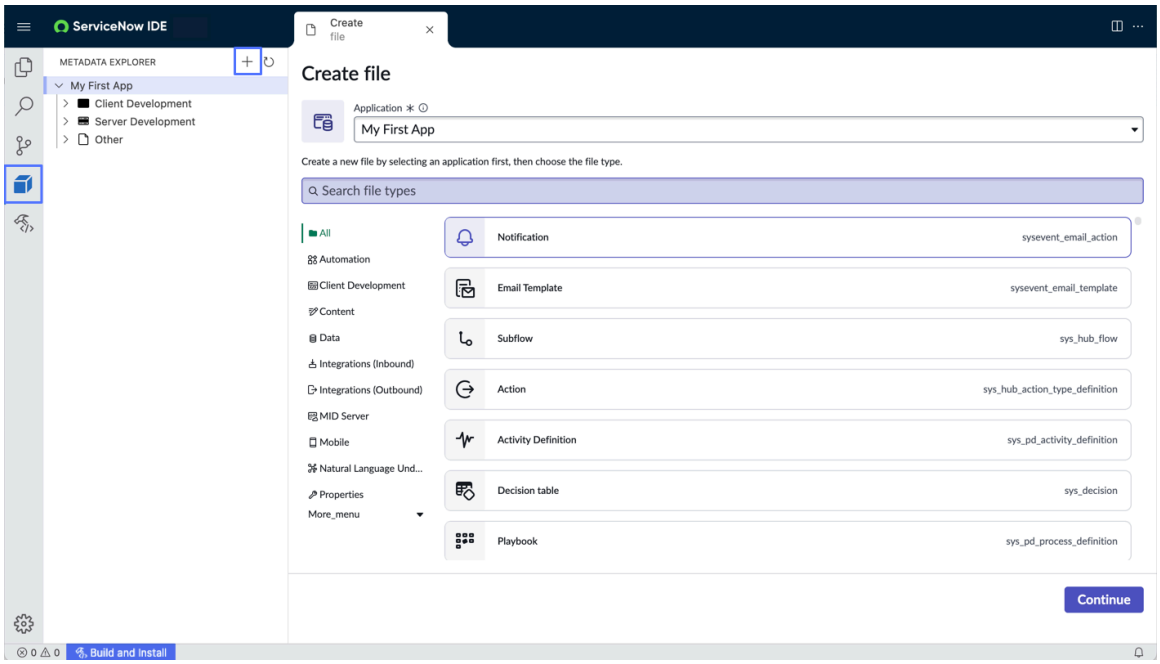
- Windows: Ctrl-P
- Mac: Cmd-P

Source Control view



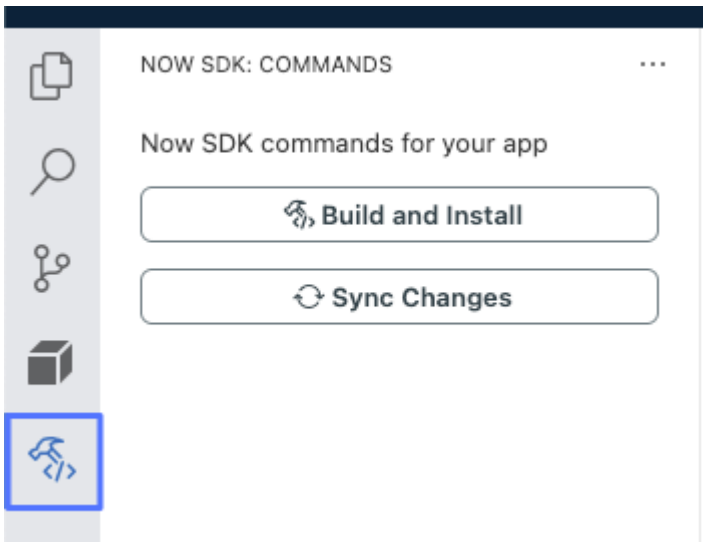
See changes to files grouped by application and the type of change. Stage, commit, and push changes to a remote repository and pull changes into your local repository. Create and checkout branches for your applications. In the Commits section, see the commit history for branches in the repository. To get started integrating with source control providers, see Integrating source control with the ServiceNow IDE.

Metadata Explorer view



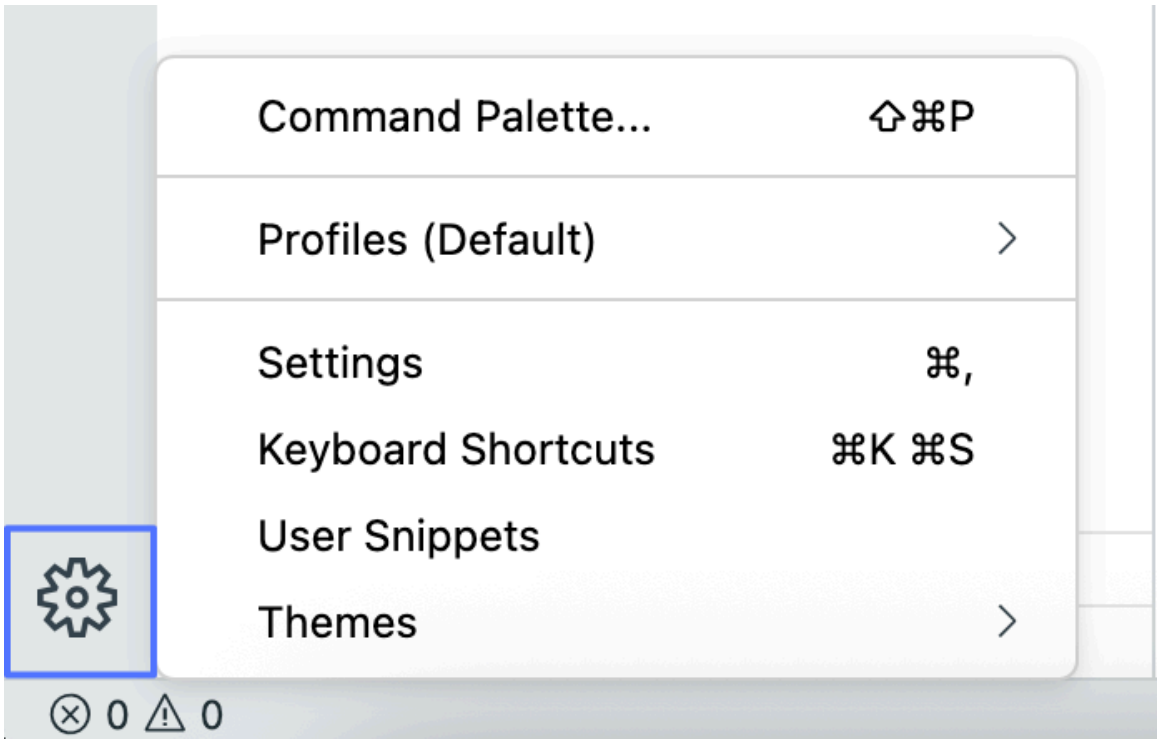
Create, update, and view application metadata in the most applicable embedded ServiceNow AI Platform user interface in the editor. If you need to create some application metadata outside of source code, you can create application files from the Metadata Explorer. For more information, see Create an application file from the Metadata Explorer.

Now SDK view



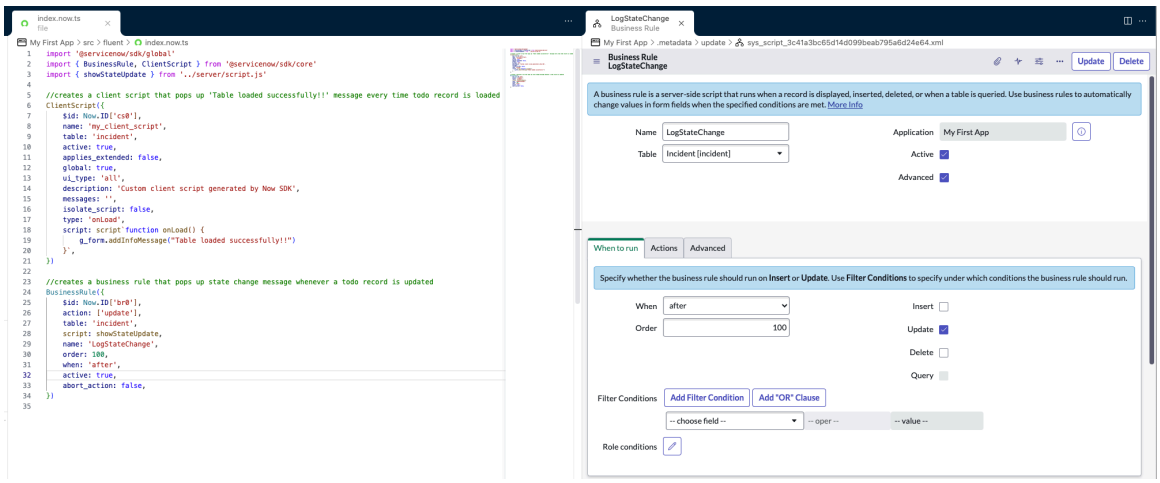
Run commands from the ServiceNow SDK application packaging service to synchronize changes to application metadata or build and install an application. For more information, see Synchronizing applications in the ServiceNow IDE and Build and install an application in the ServiceNow IDE.

Manage menu



Access and configure various aspects of the ServiceNow IDE user interface from the Manage menu. You can manage your settings, select a theme preference, view or edit keyboard shortcuts, and more.

Editor

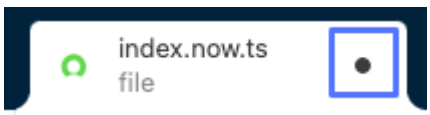


Write and edit code using IntelliSense, custom language processing and validation for ServiceNow Fluent, side-by-side editing, and more.

After building an application, review changes to ServiceNow Fluent source code as application metadata in embedded ServiceNow AI Platform user interfaces without leaving the ServiceNow IDE. Application metadata opens in the most applicable embedded ServiceNow AI Platform user interface. From links in the editor, you can open the Metadata Explorer view for metadata defined with a dedicated ServiceNow Fluent API.

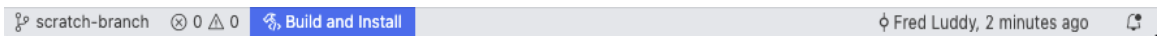
```
import '@servicenow/sdk/global'
import { DateColumn, StringColumn, Table } from '@servicenow/sdk/core'

//creates todo table, with three columns (deadline, status and task)
Open Table
export const x_snc_hello_world_to_do = Table({
    name: 'x_snc_hello_world_to_do',
    schema: {
        deadline: DateColumn({ label: 'Deadline' }),
        state: StringColumn({
            label: 'State',
            choices: {
                ready: { label: 'Ready' },
                completed: { label: 'Completed' },
                inProgress: { label: 'In Progress' },
            },
        }),
        task: StringColumn({ label: 'Task', maxLength: 120 }),
    },
})
```

Switch between tabs and split the editor into groups. If a file has unsaved changes, the file tab includes a dot icon.



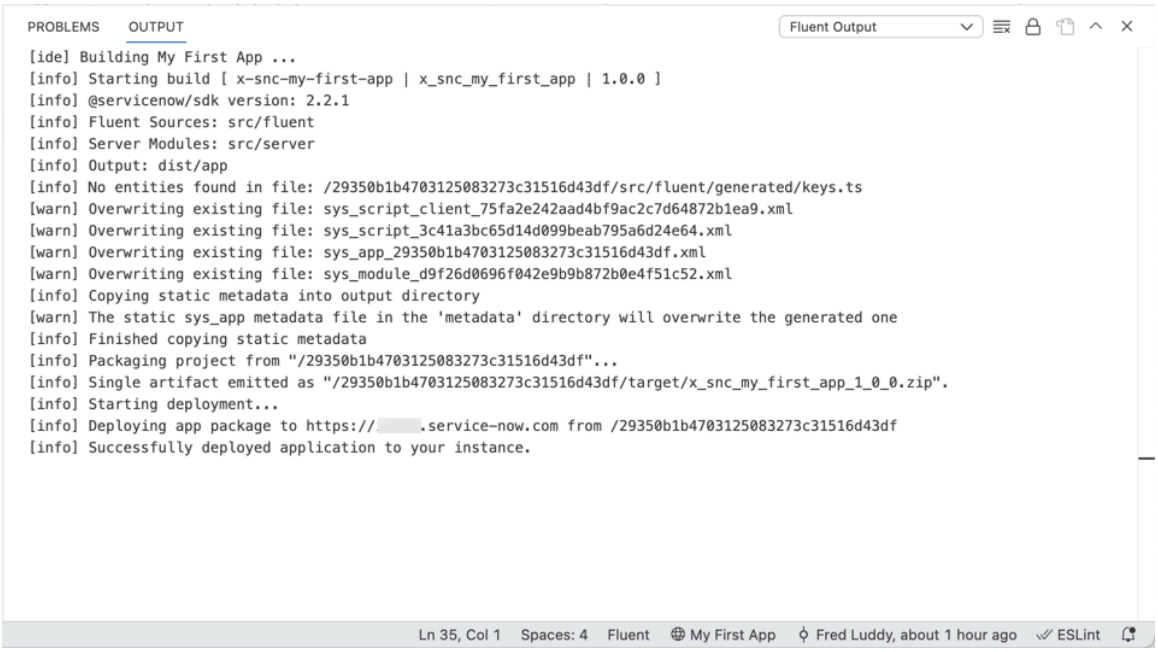
To get started developing applications in source code, see Developing applications with the ServiceNow IDE.

Status bar

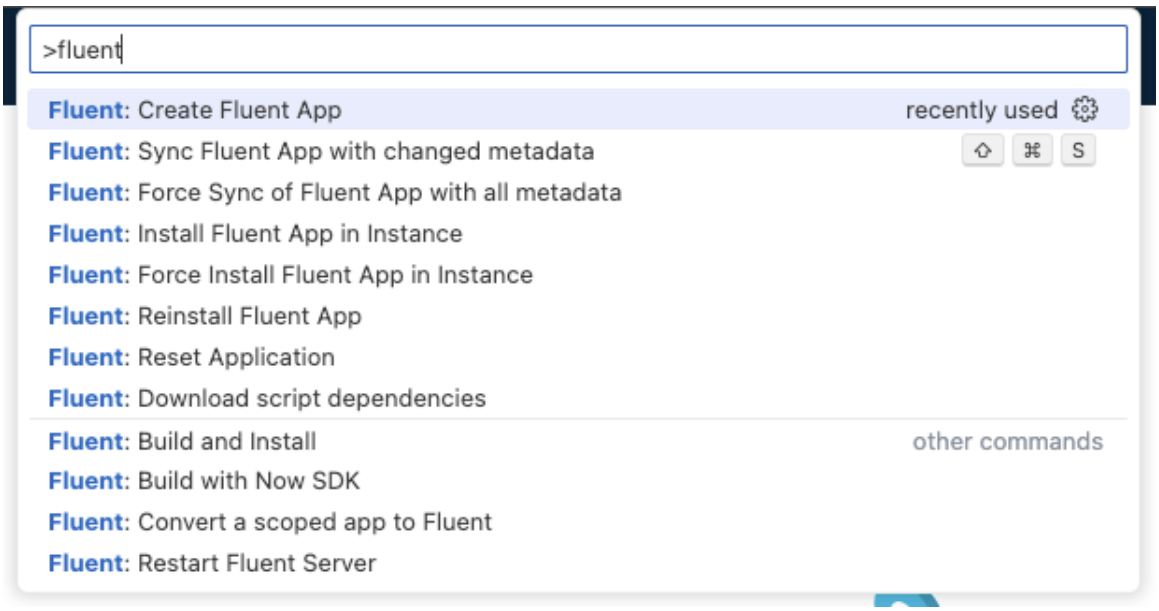


Get information about the state of your application and build applications. View the progress when creating, building, and syncing applications. In addition, the status bar displays the active branch, problems in your code, the active application scope, the active update set, and more.

From the status bar, the panel opens and displays more information about problems in your code and logging related to creating, building, and syncing applications.



Command palette



Run commands to configure the user interface, develop applications, manage files, use source control, access a list of keyboard shortcuts, and more. The command palette includes commands unique to the ServiceNow IDE and Visual Studio Code commands. For a list of commands specific to the ServiceNow IDE, see ServiceNow IDE commands.

To open the command palette, use one of the following keyboard shortcuts:

- Windows: Ctrl-Shift-P
- Mac: Cmd-Shift-P

To close the command palette, press the Escape key.

Getting started: Create your first application in the ServiceNow IDE

Get started in the ServiceNow IDE by creating an application that you can develop in source code, pushing it to a remote repository, and cloning it on another instance.

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. With the ServiceNow IDE, you can get started building scoped applications quickly using familiar tools and industry-standard development practices. For an overview of the ServiceNow IDE user interface, see ServiceNow IDE user interface.

Tutorial overview

In this tutorial, you complete the following tasks to familiarize yourself with developing a new application in source code from the ServiceNow IDE:

- Create an application that supports development in source code.
- Configure basic authentication and connect your application in the ServiceNow IDE to a remote Git repository and then push your application to the repository.
- Create a simple table for a to-do list using ServiceNow Fluent APIs.
- Install a third-party library and use its code in a JavaScript module.
- Clone the application from the remote Git repository onto another instance.

Afterward, you should be ready to take the basic application you create and continue learning to develop it into a more complex application that meets your requirements.

Before you begin

You should have access to the following items to complete this tutorial:

- A non-production instance on the Xanadu release or later with ServiceNow IDE version 2.0.4 or later installed. To complete the last part of the tutorial by cloning the application to another instance, you need access to a second non-production instance. Personal developer instances (PDIs) are supported and can be obtained from developer.servicenow.com. For more information, see Install or update the ServiceNow IDE.
- A user with the admin role on both instances.
- A Git provider such as GitHub, GitLab, Bitbucket, or Azure Repos. This tutorial demonstrates using GitHub.

Tutorial part 1: Create an application in the ServiceNow IDE

Create an application that you can develop in source code in the ServiceNow IDE.

Before you begin

Role required: admin

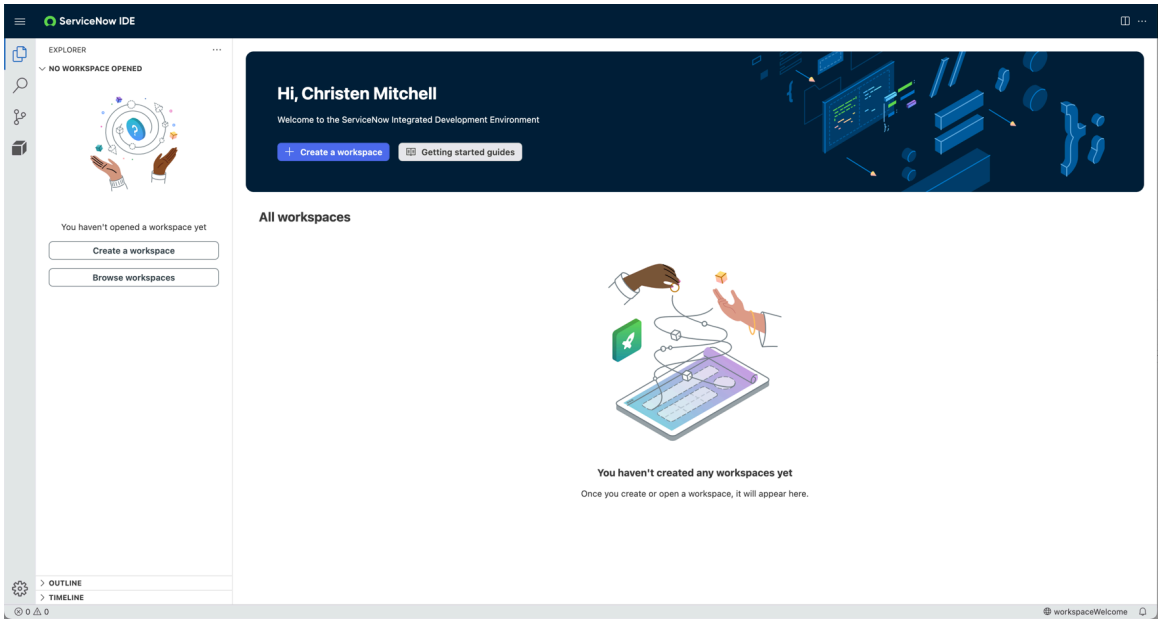
About this task

In the ServiceNow IDE, you can develop scoped applications in source code using ServiceNow Fluent to define application metadata [sys_metadata]. You can also create JavaScript modules and use third-party libraries to organize and reuse code within scoped applications. For an application to support development in source code, you must create a scoped application or convert an existing scoped application using the ServiceNow IDE or ServiceNow SDK. In this tutorial, you create a scoped application from the ServiceNow IDE.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.

   Before you can create an application, you must create a workspace in which you can add applications and navigate through them from one place.



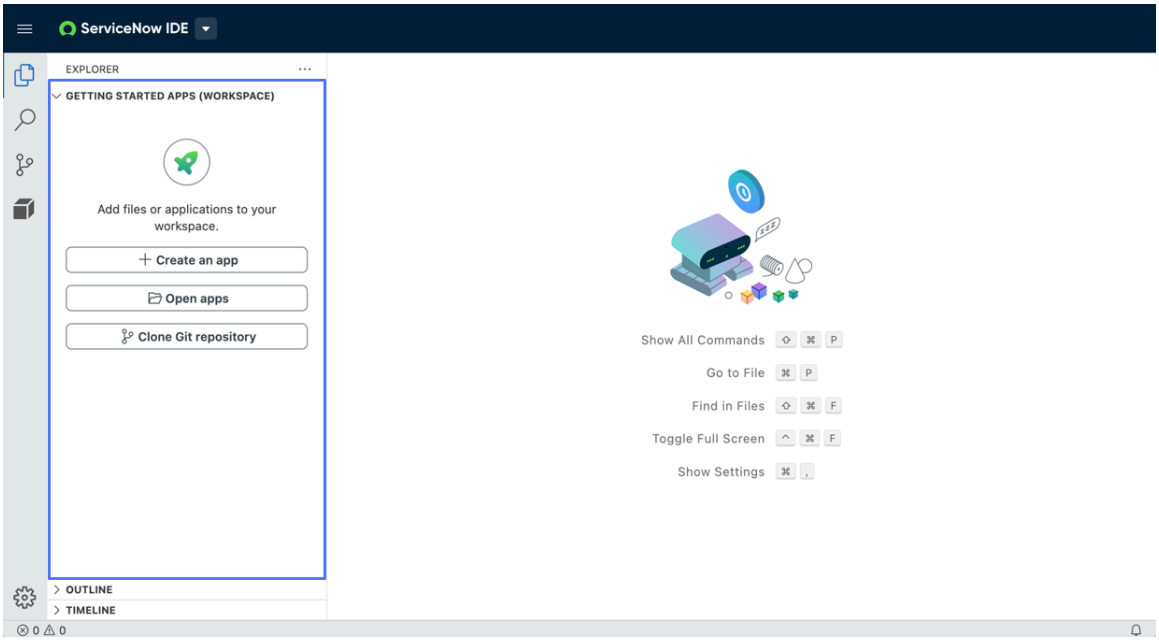
Workspaces are specific to a user, and you can create multiple workspaces to group different sets of applications. Applications can be added or removed from a workspace at any time. Workspaces in the ServiceNow IDE are based on workspaces in Visual Studio Code.

2. Create a workspace.

   a. Select **Create a workspace**.

   b. Enter a name for the workspace and press Enter.

   c. Enter a description for the workspace and press Enter.
   The workspace becomes the active workspace but doesn't contain any applications yet.



Tip:
To switch the active workspace, you can browse and select other workspaces from the ServiceNow IDE home page or from the command palette with the `Workspaces: Browse Workspaces` command.

3. Create an application in the workspace.

   a. Select **Create an app**.

   b. Enter a name, such as `Hello World`, and press Enter.

   c. Enter a description for the application and press Enter.

   d. Enter a scope, such as `x_snc_hello_world`, and press Enter.

   Important:
   The scope name must be unique on the instance, begin with x_<prefix>, and be 18 characters or fewer. For more information, see Namespace identifier.

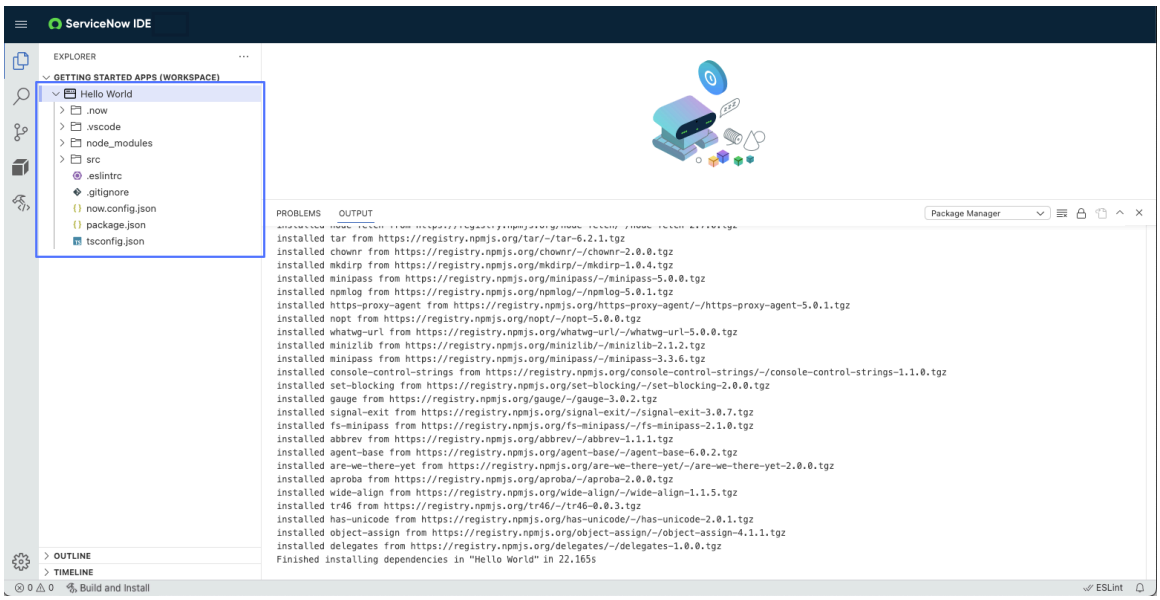
   e. Enter a package name, such as `x-snc-hello-world`, and press Enter.
   The package name must adhere to Node Package Manager (npm) package naming standards.

   f. Select the **A basic application using NowSDK and TypeScript** template to use TypeScript to create JavaScript modules.

   The application templates define the default application structure.

   Note:
   TypeScript uses static typing and type annotations to support developers catching errors earlier while writing code. If you'd prefer to get started with JavaScript instead, select **JavaScript**.

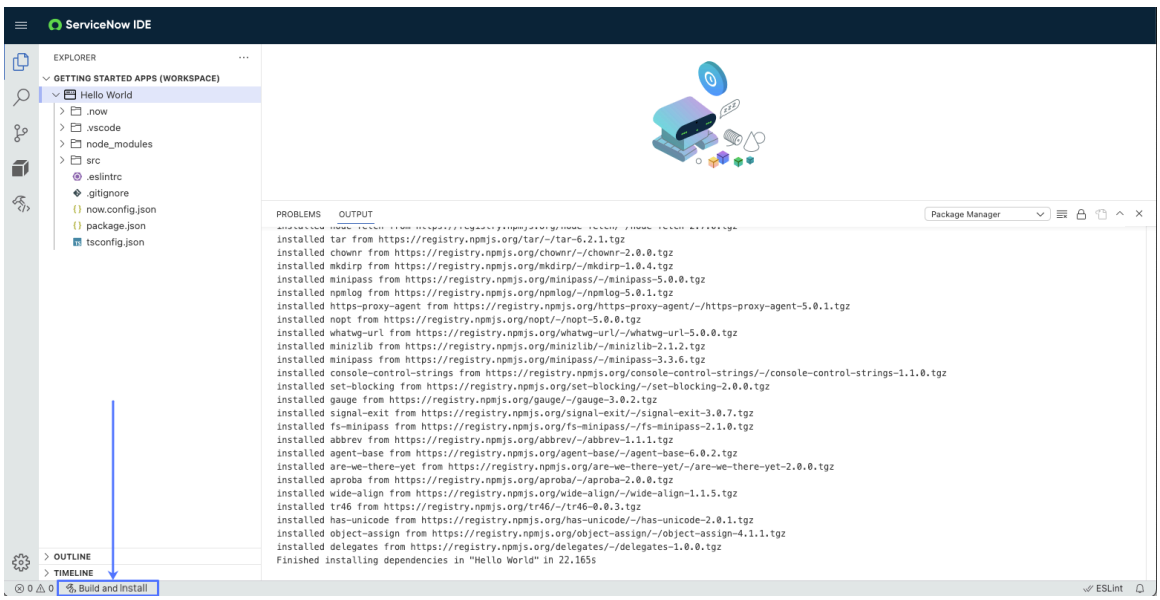
Your application is added to your workspace. From the File Explorer view ( ), you can navigate through the application files.



The `package.json` and `now.config.json` files define the application structure, which is similar to that of Node.js applications or Node Package Manager (npm) packages. A sample ServiceNow Fluent file (`index.now.ts`) and JavaScript module (`script.js` or `script.ts`) were created in the `src` directory, and dependencies were installed into the `node_modules` directory. The application is also added to the Custom Application [sys_app] table.

Next, build and install the application to make it available for development across the instance.

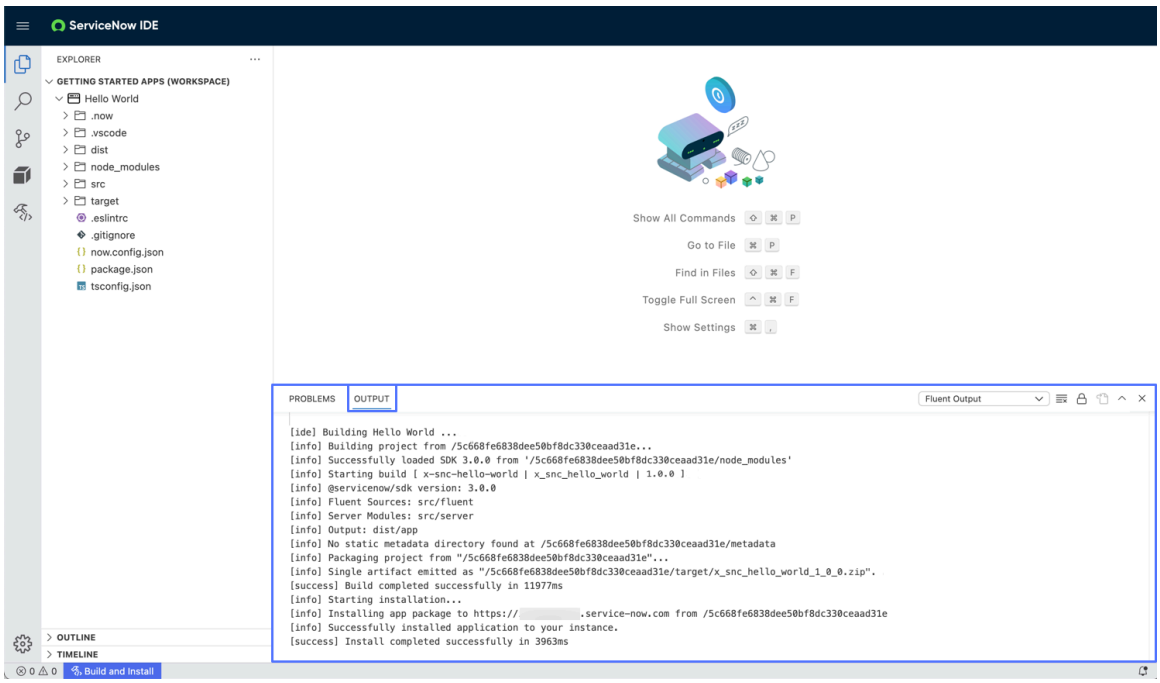
g. From the Status Bar at the bottom of the ServiceNow IDE, select **Build and Install**.



Building and installing the application compiles its ServiceNow Fluent code and JavaScript or TypeScript module code into Application Files [sys_metadata] and EcmaScript Modules

[sys_module] on the instance, respectively. Build artifacts in the `dist/app` directory are packaged into an installable `.zip` file in the `target` directory.

Logs in the Output panel indicate the status of the build and if the application is installed successfully. If either process fails, review the logs to identify any issues.



For information about the application structure, see the Application structure section of the Building applications in source code topic.

What to do next

Continue to Tutorial part 2: Initialize a repository for your application.

Tutorial part 2: Initialize a repository for your application

Initialize a local Git repository for an application in the ServiceNow IDE and push it to a remote Git repository in GitHub to manage the application in source control.

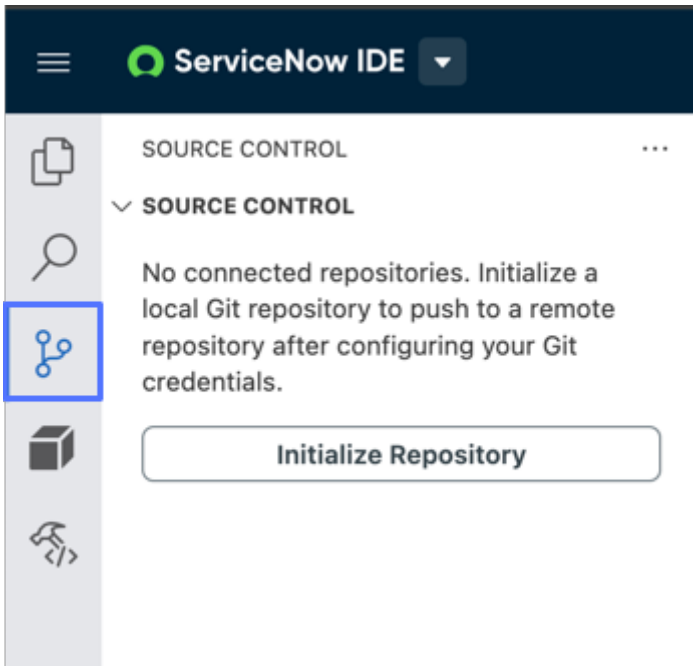
Before you begin

Complete Tutorial part 1: Create an application in the ServiceNow IDE.

Role required: admin

Procedure

1. From the Activity Bar, select the Source Control view ( ).



2. Initialize a Git repository that's local to the ServiceNow IDE.
   a. Select **Initialize Repository**.
   b. Press Enter to use **main** as the default branch name.

   A local repository is initialized for your application, and the application files that are tracked in source control are in the Untracked Changes list. Application files and directories specified in the `.gitignore` file aren't tracked in source control.

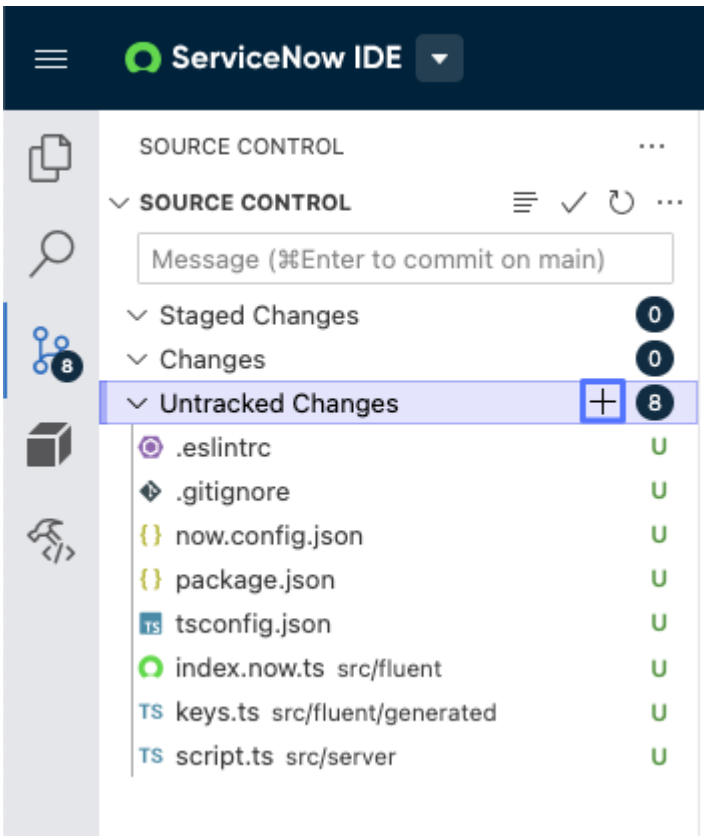
   Tip:
   If you select a file in the list, you can view a comparison of the current and previous version of the file in the Diff editor.

   Before you can push your application to a remote Git repository, you must stage and commit your changes in the local Git repository.

3. Stage and commit your changes.

a. From the Untracked Changes list, select the Stage All Untracked Changes icon (+).

Example

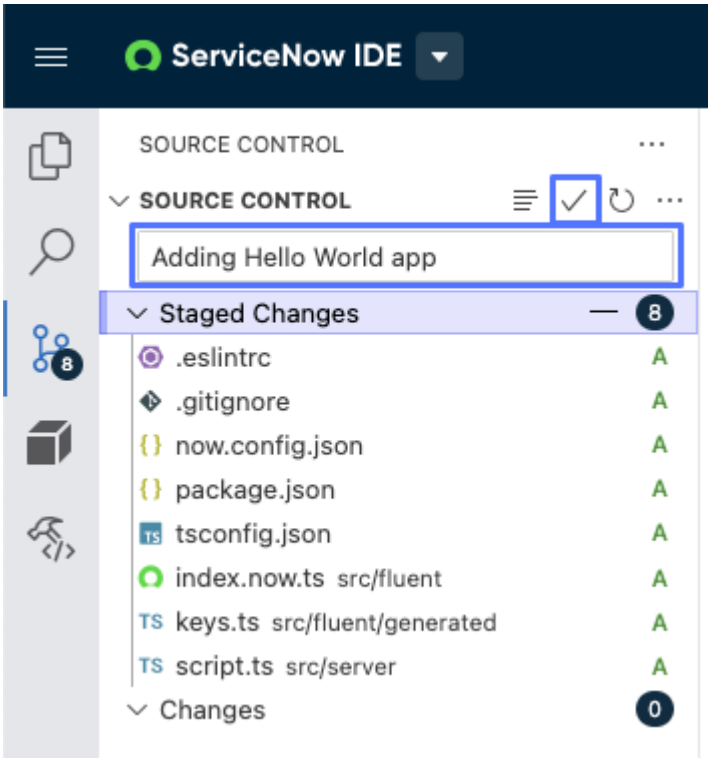


Your changes move to the Staged Changes list.

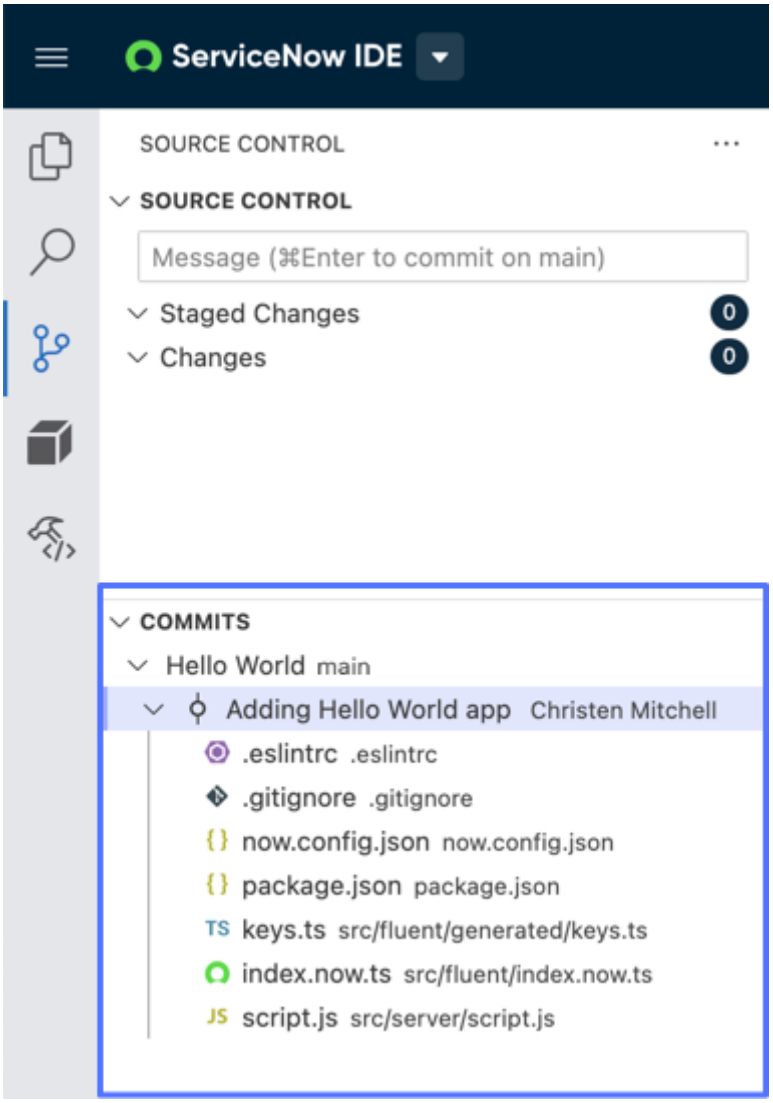
b. In the message box, enter a commit message.

c. Select the Commit icon ( ).

Example



Now that you have committed changes, you can expand the Commits section to see your commit history and details about each commit.



To push your changes to a remote repository that other developers can access, you need to create a repository with a Git provider like GitHub and generate a personal access token.

Note:
The following steps demonstrate using GitHub and basic authentication as an example, but you could use another Git provider of your choosing or OAuth 2.0 authentication. For more information, see Integrating source control with the ServiceNow IDE.

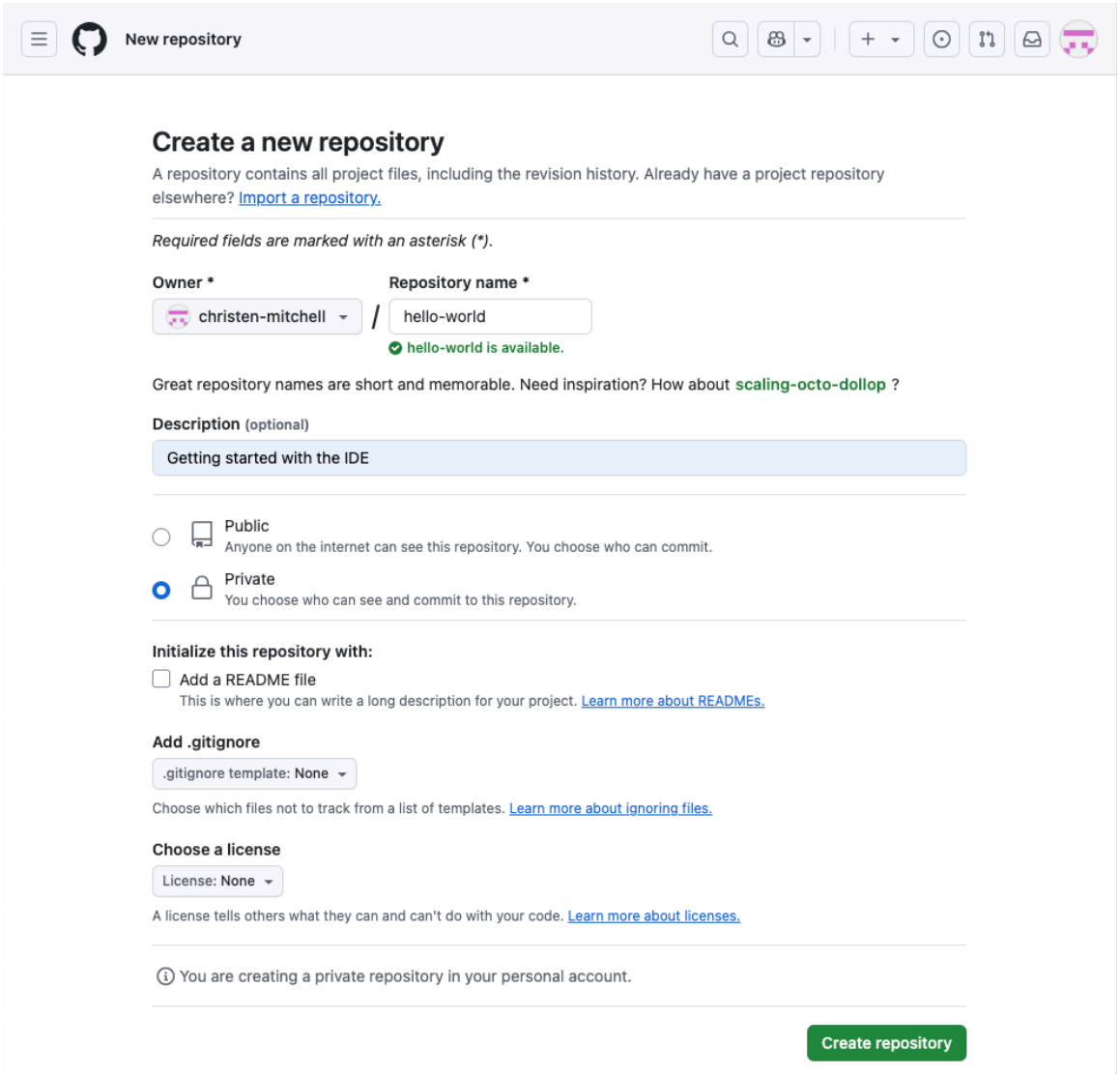
4. Create a remote repository.

   a. Log in to your GitHub account.

   b. Select the Create new menu (+) and select **New repository**.

   c. On the New repository form, fill in the fields.

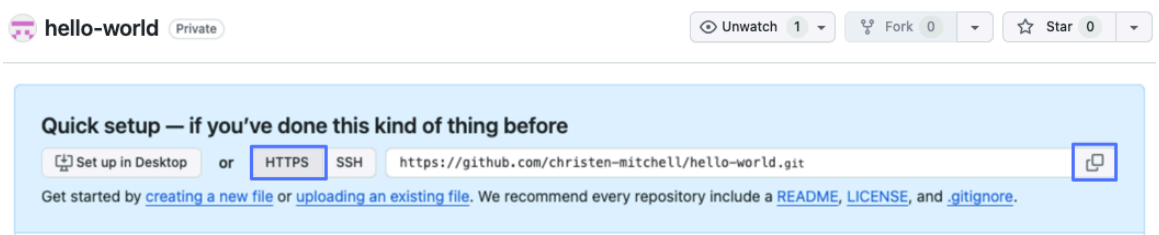
Example



d. Select **Create repository**.

e. From your repository, copy the HTTPS URL and paste it somewhere that you can easily access it again for use in the following steps.

Example



5. Create a personal access token.

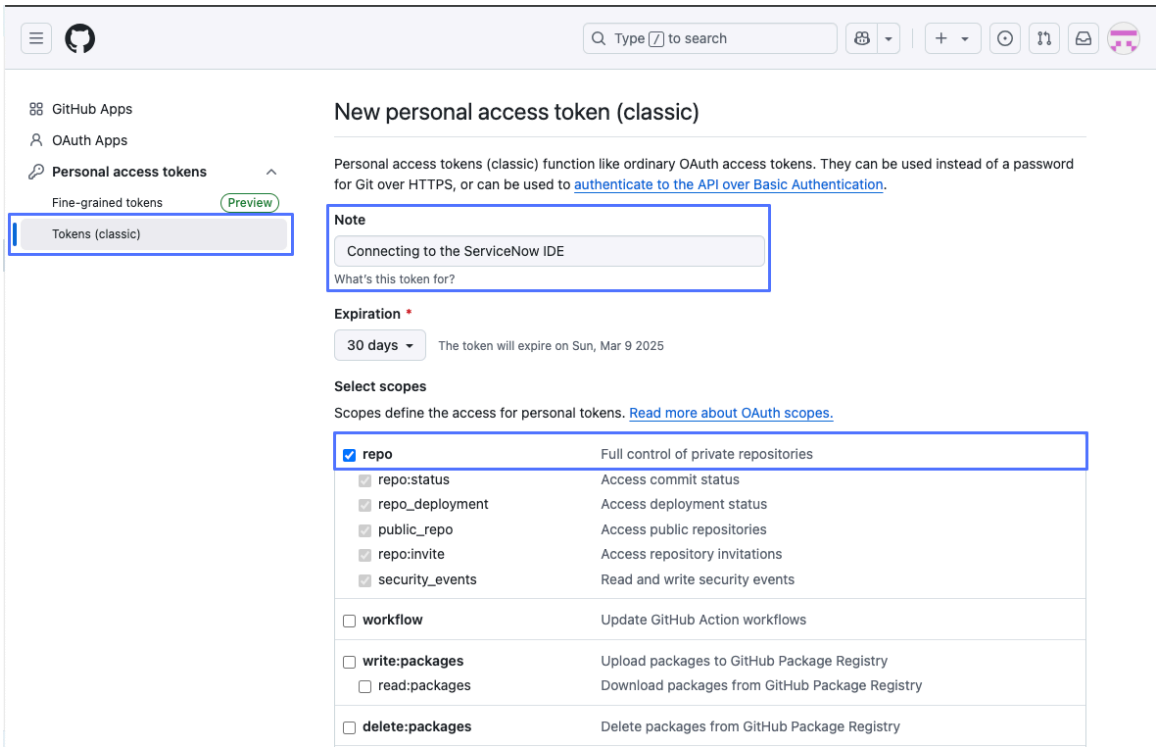
a. From your user profile in GitHub, select **Settings**.

b. From the settings categories, select **Developer Settings**.

c. From the Personal access tokens menu, select **Tokens (classic)**.

d. From the Generate new token menu, select **Generate new token (classic)**.

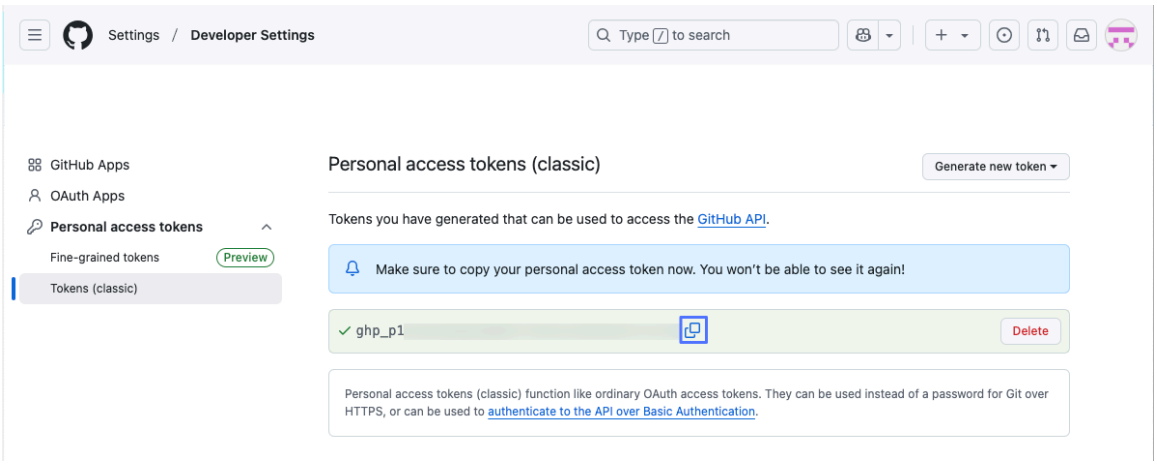
e. On the New personal access token (classic) form, select the **repo** scope and fill in the remaining required fields.
The **repo** scope is required to connect to the ServiceNow IDE.



f. Select **Generate token**.

g. Copy the token and paste it somewhere that you can easily access it again for use in the following steps so you don't need to regenerate it.

Example



6. Configure basic authentication in the ServiceNow IDE to connect to the remote repository.

a. Navigate to the ServiceNow IDE.

b. Use one of the following keyboard shortcuts to open the command palette:

- Windows: Ctrl-Shift-P
- Mac: Cmd-Shift-P

c. Enter `Git: Set IDE Git credentials` and press Enter.

d. From the New Git credential form, select **Basic auth**.

e. On the form, fill in the fields.

New Git credential form

Field	Description
Git repository URL	The HTTPS URL of the Git repository you created in GitHub. For example: `https://github.com/<owner>/hello-world.git`.
Git username	Your GitHub user name.
Personal access token	The personal access token that you generated from GitHub.

f. Select **Submit**.

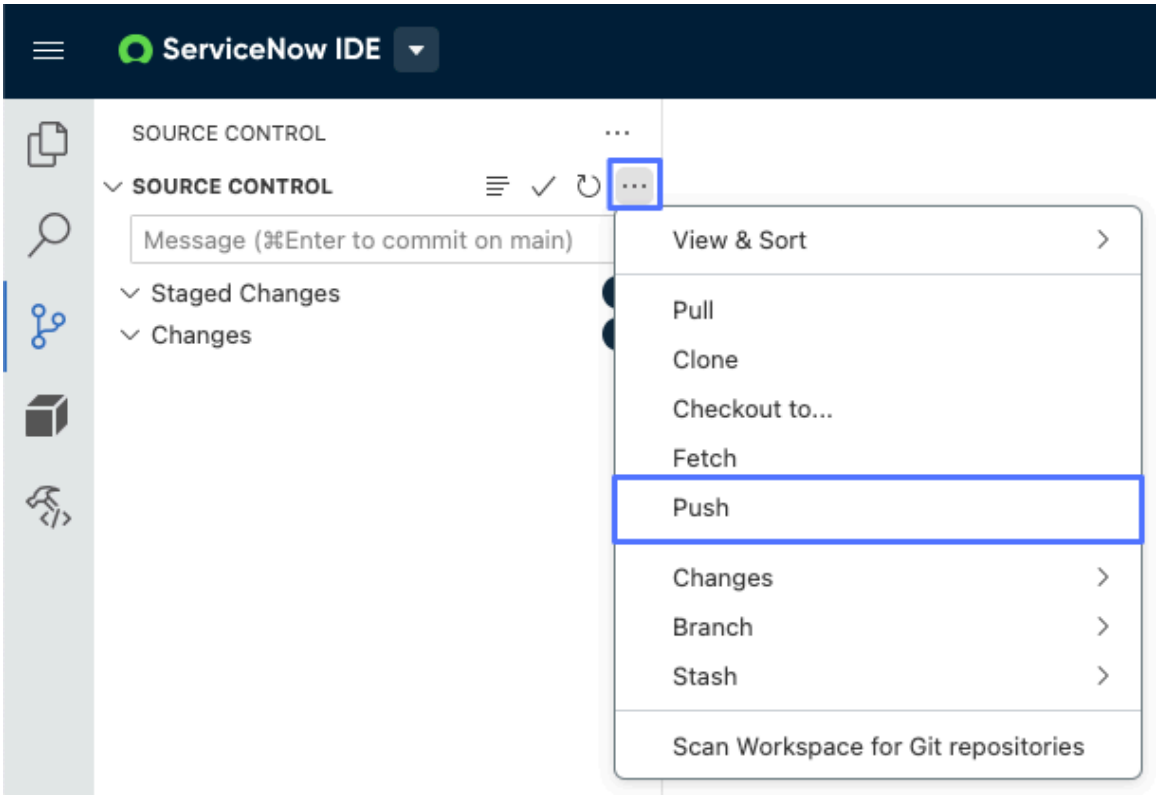
Now that your authentication credentials are configured, you can push your application to the remote repository.

Tip:
If you need to manage existing Git credentials, use the `Git: Manage Git credentials` command from the command palette.

7. Push your changes to the remote repository.

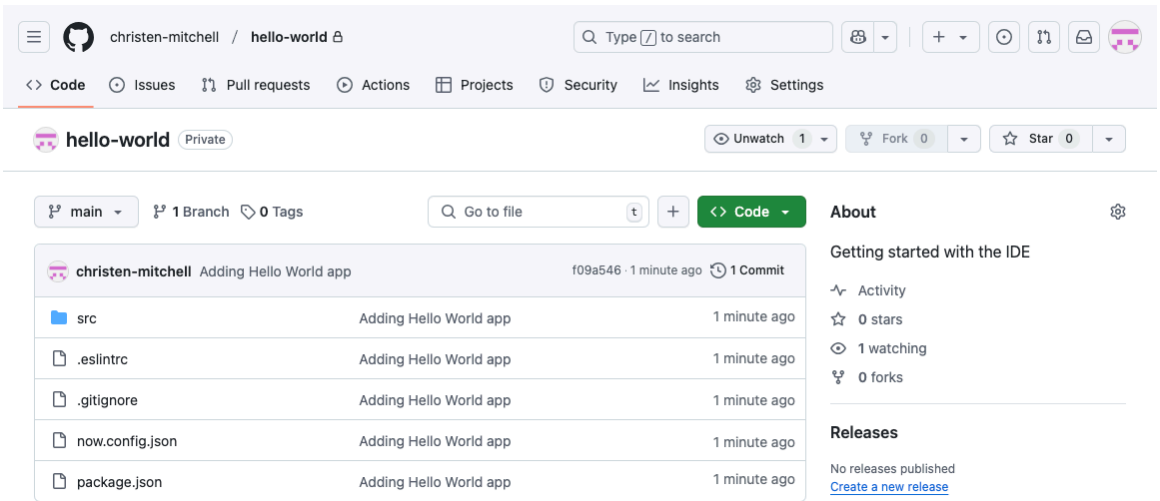
a. Select the More actions menu icon (···) and select **Push**.

Example



b. Enter the HTTPS URL of the remote Git repository and press Enter.

The application is added to the remote repository in GitHub, and changes to the application across users are tracked remotely.



Result

You can check out or create branches in the repository and push changes to the remote repository. For more information, see Using source control in the ServiceNow IDE.

Note:
An application on an instance can be connected to only one repository at a time. To update the remote repository an application is connected to, you can use the `Git: Update remote origin` command from the command palette and enter a different remote repository URL.

What to do next
Continue to Tutorial part 3: Define a table in ServiceNow Fluent code.

Tutorial part 3: Define a table in ServiceNow Fluent code

Create a table and reference it in sample script definitions using the ServiceNow Fluent APIs.

Before you begin
Complete Tutorial part 2: Initialize a repository for your application.

Role required: admin

About this task
ServiceNow Fluent is a domain-specific language (DSL) based on TypeScript for defining the metadata files [sys_metadata] that make up applications and includes APIs for the different types of metadata, such as tables, roles, ACLs, business rules, and Automated Test Framework tests. You use objects in the ServiceNow Fluent APIs to define metadata in files with the `.now.ts` extension. The ServiceNow IDE has language processing and validation for ServiceNow Fluent APIs and applications by default. For more information about ServiceNow Fluent, see ServiceNow Fluent.

In this example, you create a simple table for a to-do list using objects in the ServiceNow Fluent Table API. Then, you update the sample code for business rule and client script definitions to reference the new table. Lastly, you review your changes from the Metadata Explorer. For more information about the Table API, see Table API - ServiceNow Fluent.

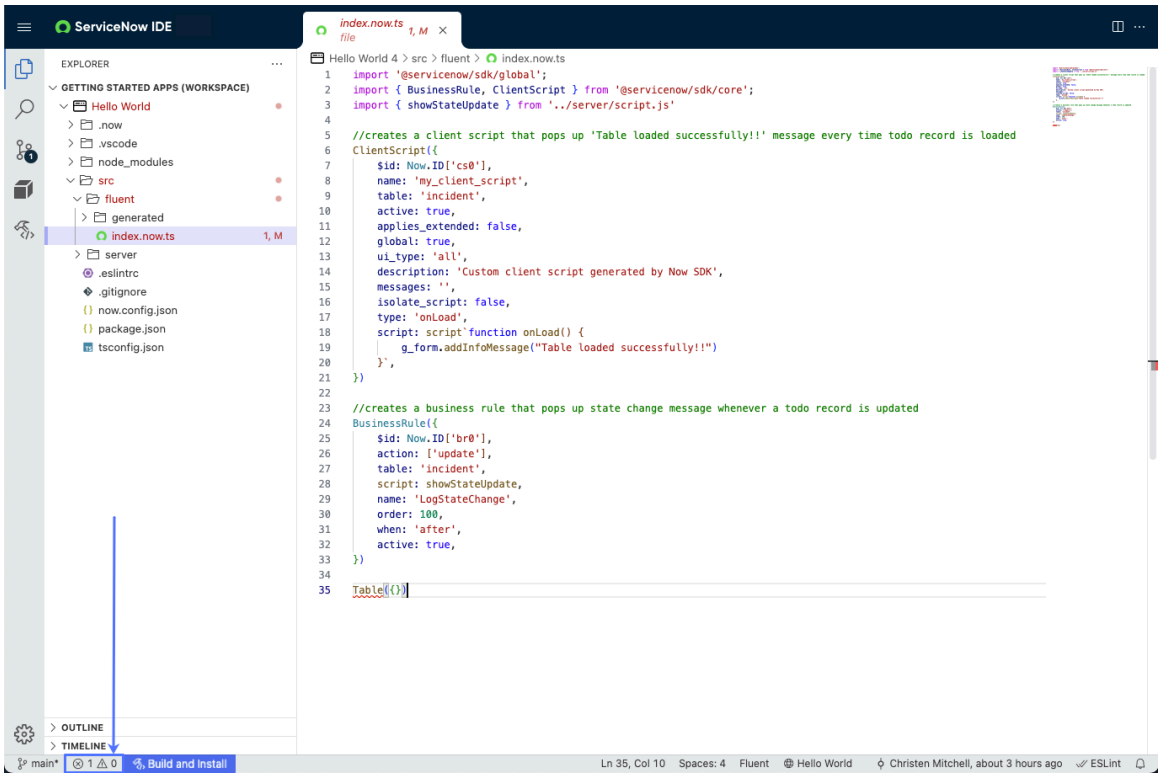
Procedure

1. From the Activity Bar, select the File Explorer view ( ).
2. Navigate to the `src/fluent` directory in your application.
3. Open the `index.now.ts` sample file.

   Tip:
   You can write ServiceNow Fluent code in a single file or as many `.now.ts` files as you want and organize files in directories within the `fluent` directory.

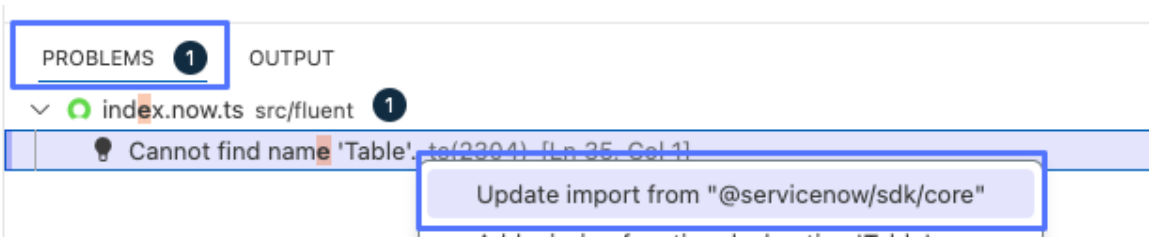
4. On a new line at the end of the file, enter `Table({})` to add the Table object.
5. From the Status Bar, select the diagnostics icon (⊗ 1 ⚠ 0) to open the Problems panel and check for issues in the code.

Example



6. Right-click the error that appears and select the **Update import from "@servicenow/sdk/core"** quick fix.

Example



In line 2, the *Table* object is added to the list of imports from `@servicenow/sdk/core`:

```
import { BusinessRule, ClientScript, Table }
 from '@servicenow/sdk/core'
```

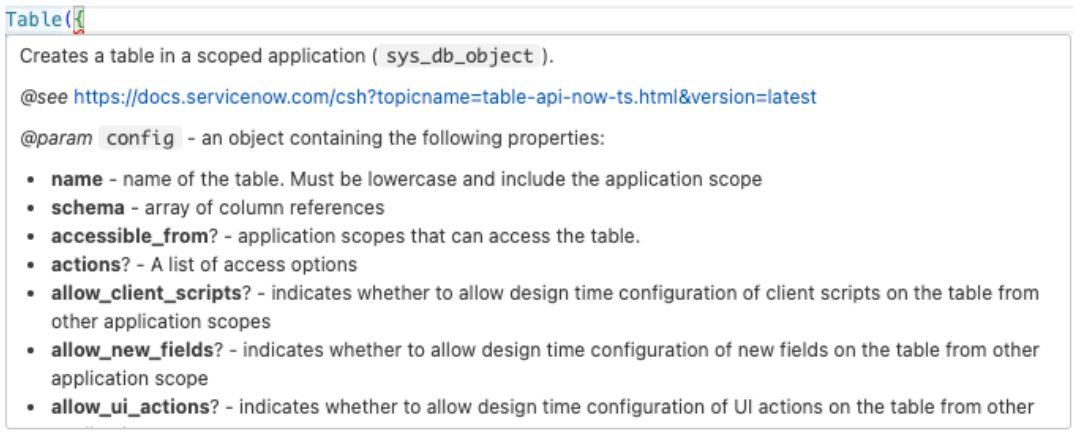
Tip:
After fixing this issue, you can close the Problems panel while you proceed through the following steps. In a later step, you return to it if any issues remain.

7. In the *Table* object, add the following properties.

   - name: The table name must begin with the application scope and use all lowercase letters in the following format: `<scope>_<name>`. You can find the scope in the `now.config.json` file for the application.
   - label: The label should be unique and appears for the table in list and form views.
   - extends: The name of another table on which the table is based.

```
Table({
    name: 'x_snc_hello_world_to_do', //ensure that the name begins with the correct scope (<scope>_<name>)
    label: 'To-do Items',
    extends: 'task',
})
```

Tip:
Hover over an object to see its in-product documentation.



8. For type-ahead support when defining columns in the table, before the *Table* object, add an exported variable with the same name as the *name* property.

```
export const x_snc_hello_world_to_do = Table({
    name: 'x_snc_hello_world_to_do',
    label: 'To-do Items',
    extends: 'task',
})
```

9. In the *Table* object, add the *schema* property to define columns in the table.

```
export const x_snc_hello_world_to_do = Table({
    name: 'x_snc_hello_world_to_do',
    label: 'To-do Items',
    extends: 'task',
    schema: {
        //define columns here
    }
})
```

The schema property is an array of *Column* objects. There are many types of columns based on the field type. Column objects use the format *<Type>Column* where *<Type>* is the field type.

Use the following details to define three columns in the table: Deadline, Matrix, and Task. Refer to the Column object documentation to help you configure each column.

Column name	Details
deadline	○ Label: Deadline ○ Type: Date/Time

Column name	Details
matrix	○ Label: Matrix ○ Type: String ○ Choices: ▪ Label: Urgent and Important ▪ Label: Important but Not Urgent ▪ Label: Urgent but Not Important ▪ Label: Neither Urgent nor Important
task	○ Label: Task ○ Type: String ○ Max length: 120

Example
With these details, the *schema* property should look similar to this example. The keys that you use for the choices can be any string.

```
export const x_snc_hello_world_to_do = Table({
    name: 'x_snc_hello_world_to_do',
    label: 'To-do Items',
    extends: 'task',
    schema: {
        deadline: DateColumn({ label: 'Deadline' }),
        matrix: StringColumn({
            label: 'Matrix',
            choices: {
                do: { label: 'Urgent and Important' },
                decide: { label: 'Important but Not Urgent' },
                delegate: { label: 'Urgent but Not Important' },
                delete: { label: 'Neither Urgent nor Important' },
            },
        }),
        task: StringColumn({ label: 'Task', maxLength: 120 }),
    },
})
```

10. In line 2, add imports for the *DateColumn* and *StringColumn* objects.

```
import { BusinessRule, ClientScript, Table, DateColumn,
 StringColumn } from '@servicenow/sdk/core'
```

11. Update the existing business rule and client script definitions to reference the table you created.
    a. In the *ClientScript* object, change the value of the *table* property to the table name (`x_snc_hello_world_to_do`).
    b. Repeat the previous step for the *BusinessRule* object.
12. If the diagnostics icon (⊗ 1 ⚠ 0) shows any errors or warnings, select it to open the Problems panel and review the diagnostic messages and quick fixes to resolve them.
13. Save your changes using one of the following keyboard shortcuts.

- Windows: Ctrl-S
- Mac: Cmd-S

Note:
If you have unsaved changes in a file, a dot icon appears on the file tab.

14. From the Status Bar, select **Build and Install**.

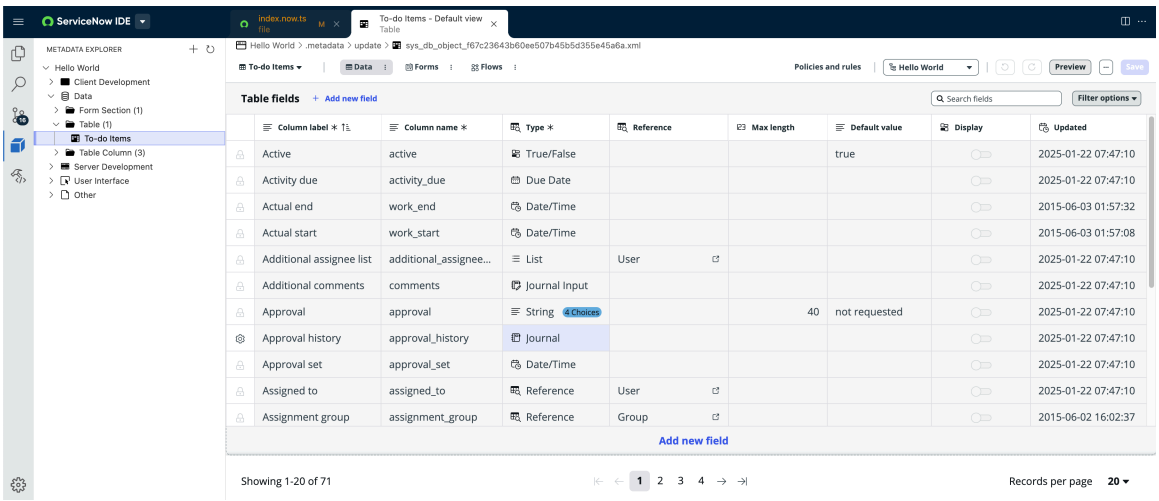
    If the installation completes successfully, the updated ServiceNow Fluent source code is compiled into Application Files [sys_metadata] on the instance.

15. Review your changes as metadata.

    a. From the Activity Bar, select the Metadata Explorer view ( ).

    b. Select your application to expand it.

    c. Navigate to **Data** > **Table** and select **To-do Items**.
    The table opens in Table Builder.



16. **Optional:** Edit the metadata and synchronize your changes into the source code.
    From the Metadata Explorer, you can simulate another user editing the metadata outside of the source code to see changes transformed back into the code you added.

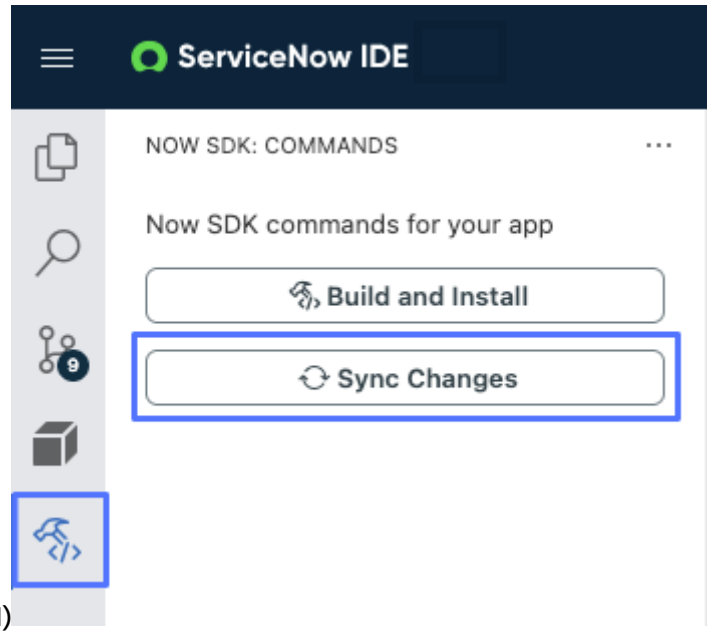
    a. In Table Builder, search for the Task field and change the Column label from "Task" to "To Do".

    b. Select **Save**.

    c. From the Activity Bar, select the Now SDK view ( ).

    d. Select **Sync Changes**.

Example



(Optional)

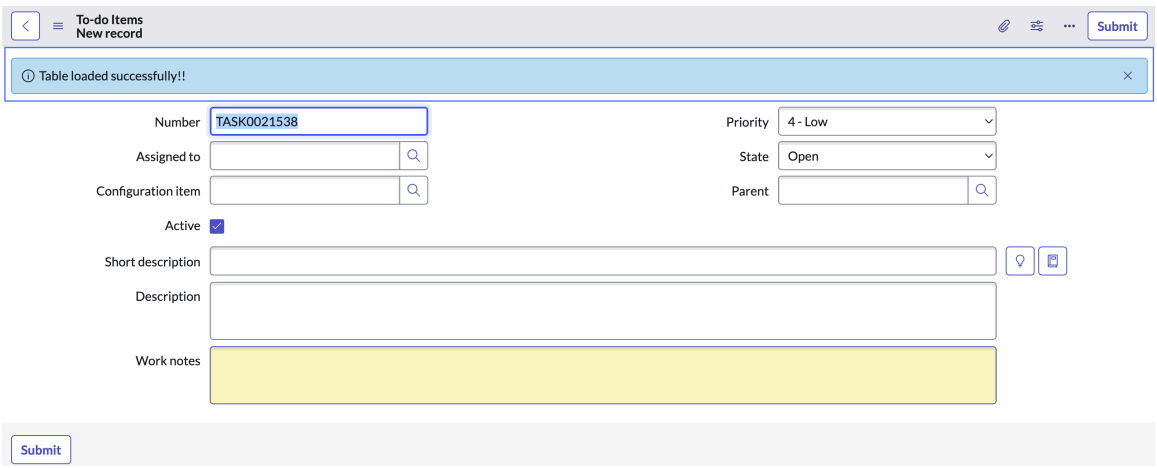
In the `index.now.ts` file, you should see the *`label`* property of the task column changed to `'To Do'`.

```
task: StringColumn({ label: 'To Do', maxLength: 120 }),
```

Result

You have created your first application metadata using ServiceNow Fluent APIs. The To-do Items [x_snc_hello_world_to_do] table can be modified in source code by other ServiceNow IDE users or from other ServiceNow AI Platform user interfaces.

From the ServiceNow AI Platform, you can navigate to the list view of the table by entering `x_snc_hello_world_to_do.list` in the navigation filter. Because you updated the client script definition to run on the To-do Items [x_snc_hello_world_to_do] table, if you select **New** to add a record to the table, the message from the client script appears when the record loads.



What to do next

Continue to Tutorial part 4: Install and use a third-party library.

Tutorial part 4: Install and use a third-party library

Install a third-party library from Node Package Manager (npm) and use it in a JavaScript module.

Before you begin

Complete Tutorial part 3: Define a table in ServiceNow Fluent code.

Role required: admin

About this task

Installing third-party libraries allows you to use existing open-source functionality in JavaScript modules to accelerate application development. Then you can refer to JavaScript modules that call third-party code from server-side script definitions in your source code, such as the business rule in the `index.now.ts` file.

In this example, you install a Lodash library to get common JavaScript utilities and methods. You use one of these methods, *`snakeCase`*, in the *`showStateUpdate`* function in a sample JavaScript module to display a message string in snake case, which separates words with underscores instead of spaces. In the `index.now.ts` file, the sample business rule is configured to use the *`showStateUpdate`* function for its script and to run after a record is updated in the To-do Items [x_snc_hello_world_to_do] table.

Procedure

1. Install the *`snakeCase`* method from the Lodash library in your application.

   a. From the Activity Bar, select the File Explorer view ( ).

   b. Open the `package.json` file for the application.

   c. After the `devDependencies` field, add the `dependencies` field with the package name and version of the library.

   Example

   ```
   },
        "dependencies": {
             "lodash.snakecase": "4.1.1"
        }
   }
   ```

   d. For applications that use TypeScript in JavaScript modules, add the lodash.snakecase types to the `devDependencies` field to get the type annotations for the library.

   ```
   "devDependencies": {
             "@types/lodash.snakecase": "4.1.1",
             "typescript": "5.5.4",
             "@servicenow/sdk": "2.2.4",
             "@servicenow/glide": "26.0.1",
             "eslint": "8.50.0",
             "@servicenow/eslint-plugin-sdk-app-plugin": "2.2.4"
        }
   ```

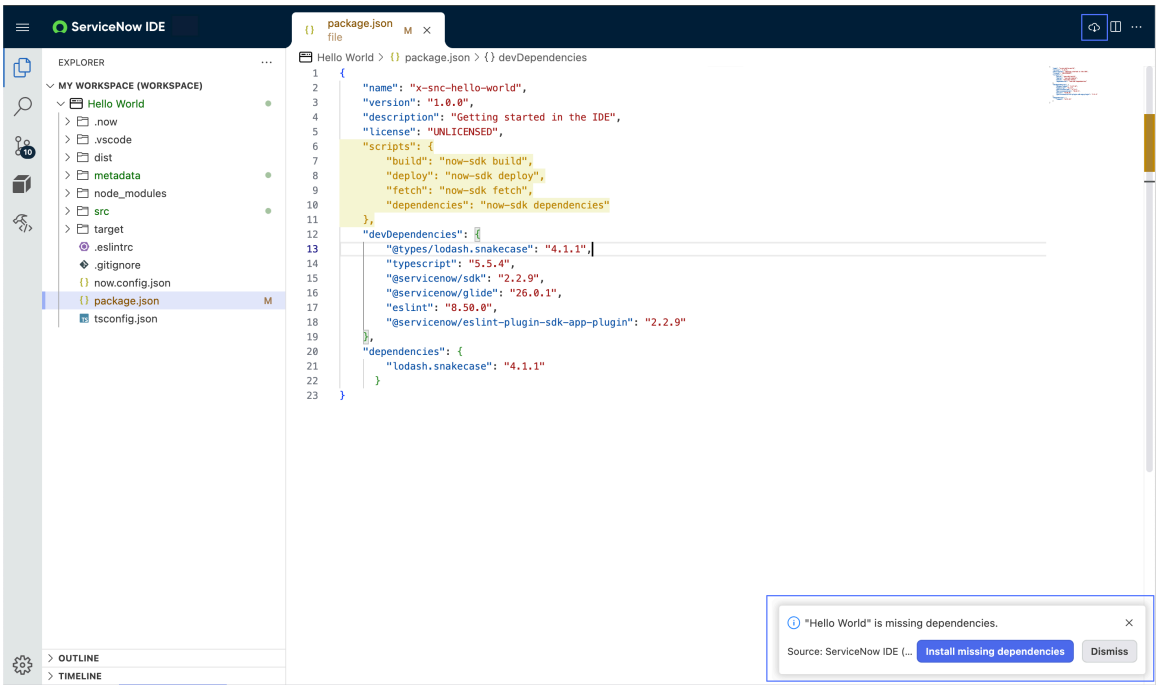
   e. Save your changes.

   f. When prompted, select **Install missing dependencies**.

   Tip:

   You can also select the Install Dependencies icon ( ) or use the `Package Manager: Install Dependencies` command from the command palette.

Example



Libraries are installed as modules in the `node_modules` directory.

2. Use code from the Lodash library in a JavaScript module in your application.

   a. Navigate to the `src/server` directory in your application.

   b. Open the `script.ts` sample module.

   c. In the *`addInfoMessage`* method, wrap the message string in the *`snakeCase`* method from Lodash to convert it to snake case.

```
gs.addInfoMessage(snakeCase(`state updated from "${previousState}"
 to "${currentState}"`))
```

   d. In line 2, add an import for the *`snakeCase`* method in the *`lodash`* module.

Example

```
import snakeCase from 'lodash.snakecase'
```

> **Note:**
> The global Glide APIs are also imported so that you can use methods like *`addInfoMessage`* in your module code.
>
> ```
> import { gs } from '@servicenow/glide'
> ```

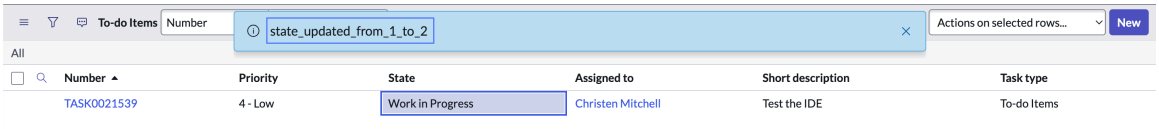
   e. Save your changes.

3. From the Status Bar, select **Build and Install**.

   If the installation completes successfully, the Lodash libraries are added to the EcmaScript Module [sys_module] table, and the `script.ts` module is updated in the EcmaScript Module [sys_module] table.

Result

After you update any field on a record in the To-do Items [x_snc_hello_world_to_do] table, the sample business rule runs and displays the message in snake case, with the words separated by underscores instead of spaces.



What to do next

Continue to Tutorial part 5: Clone the application on a different instance.

Tutorial part 5: Clone the application on a different instance

Clone the application from the remote repository to develop it on another instance.

Before you begin

Complete Tutorial part 4: Install and use a third-party library.

Role required: admin

About this task

If you have access to another instance, you can install the application from the remote repository onto the instance to support development across instances.

Note:
The instance and your user on the instance must meet the same requirements mentioned at the beginning of this tutorial.

Cloning is intended for developing an application on multiple non-production instances and managing it in a single repository. To publish an application and deploy it to a production instance, use the Application Repository. For more information, see ServiceNow application repository.

Procedure

1. Push your changes to the remote repository.

   a. From the Activity Bar, select the Source Control view ( ).

   b. From the Changes list, select the Stage All Tracked Changes icon ( ).

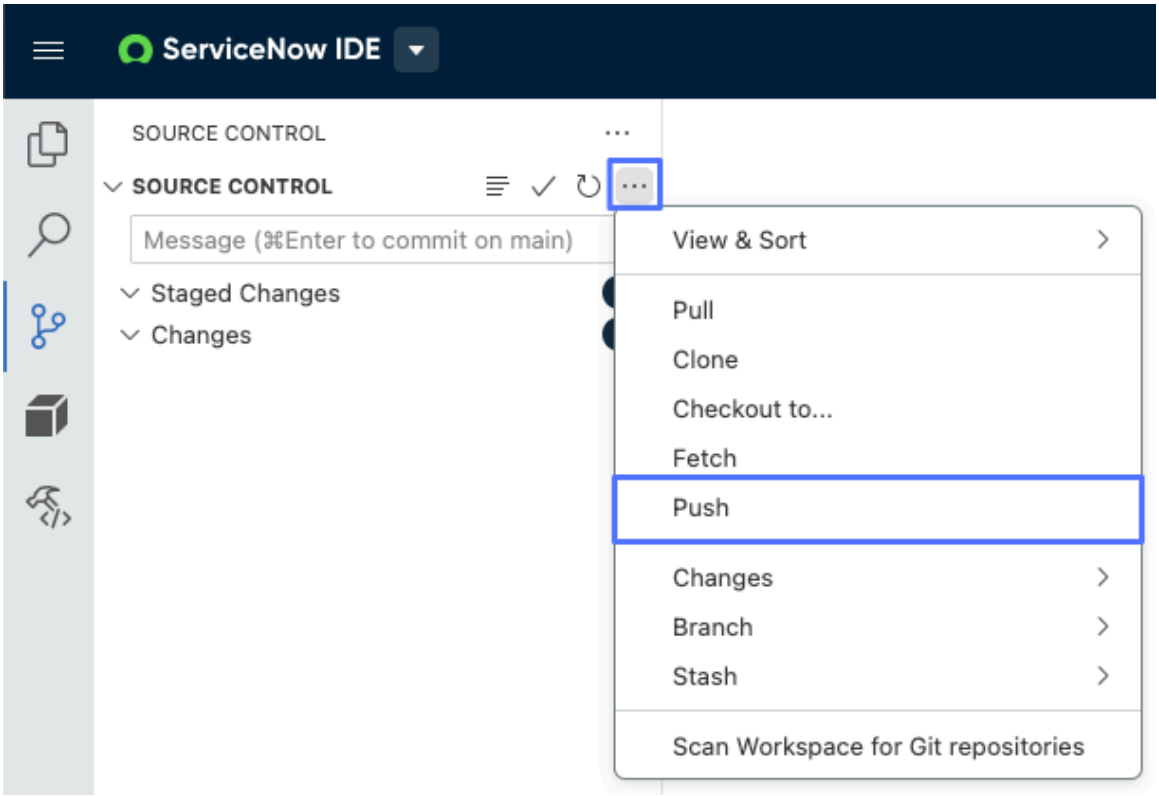
   c. From the Untracked Changes list, select the Stage All Untracked Changes icon ( ).
   All of your changes should be in the Staged Changes list.

   d. In the message box, enter a commit message.

   e. Select the Commit icon ( ).

f. Select the More actions menu icon (···) and select **Push**.

Example



All of your changes are available in the remote repository. You or other developers can clone the application onto another instance.

2. Log in to another ServiceNow instance.
3. Navigate to **All** > **App Development** > **ServiceNow IDE**.
4. Create a workspace or open an existing one.
5. Configure basic authentication in the ServiceNow IDE to connect to the remote repository.

   a. Use one of the following keyboard shortcuts to open the command palette:
   - Windows: Ctrl-Shift-P
   - Mac: Cmd-Shift-P

   b. Enter `Git: Set IDE Git credentials` and press Enter.

   c. From the New Git credential form, select **Basic auth**.

   d. On the form, fill in the fields.

New Git credential form

Field	Description
Git repository URL	The HTTPS URL to the Git repository you created in GitHub. For example: `https://github.com/<owner>/hello-world.git`.
Git username	Your GitHub user name.
Personal access token	The personal access token that you generated from GitHub.

e. Select **Submit**.

6. Clone the application.

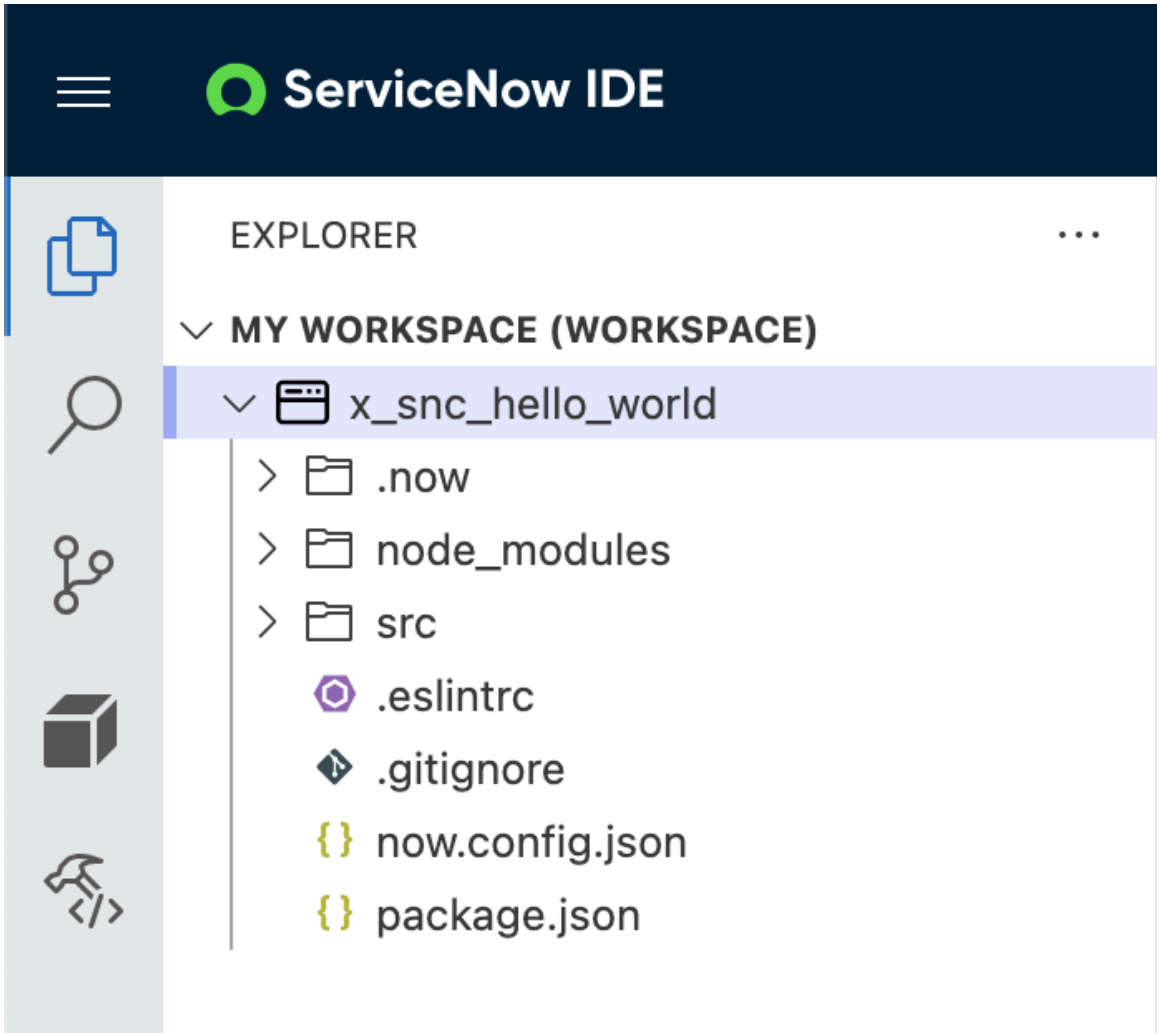
a. Use one of the following keyboard shortcuts to open the command palette:

- Windows: Ctrl-Shift-P
- Mac: Cmd-Shift-P

b. Enter `Git: Clone` and press Enter.

c. Enter the HTTPS URL to the remote Git repository and press Enter.

Result
The application is added to the instance with the files from the remote repository.



What to do next
Continue to Tutorial part 6: Learn more about the ServiceNow IDE.

Tutorial part 6: Learn more about the ServiceNow IDE

Learn more about the ServiceNow IDE to develop more complex applications.

Congratulations! You have created your first application in the ServiceNow IDE, learned how to write ServiceNow Fluent code to define application metadata, installed a third-party library and used its code in a JavaScript module, and managed all of these changes in source control with GitHub.

For more information about development in the ServiceNow IDE, refer to the following resources:

- ServiceNow IDE documentation
- ServiceNow Fluent API reference
- Introduction to the ServiceNow IDE on ServiceNow University
- ServiceNow IDE Demo video
- ServiceNow IDE and Fluent Creator Toolbox video
- ServiceNow SDK examples GitHub repository
- ServiceNow IDE, SDK, and Fluent forum in the ServiceNow Community
- ServiceNow IDE, SDK, and Fluent articles in the ServiceNow Community

- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

Configuring the ServiceNow IDE

Install and set up the ServiceNow IDE for developers to use on a ServiceNow instance.

Install or update the ServiceNow IDE

Install or update the ServiceNow IDE application (sn_glider) from the ServiceNow® Store.

Before you begin

Review the ServiceNow IDE application listing in the ServiceNow Store for information on dependencies, licensing or subscription requirements, and release compatibility.

Role required: admin

About this task

ServiceNow IDE is active by default on instances on the Xanadu release. Update to the latest version of the ServiceNow IDE to use the latest features.

Procedure

1. Navigate to **Admin** > **Application Manager**.
2. Search for the ServiceNow IDE application (sn_glider).

   You can search for the application by its name or ID. If you can't find the application, you might have to request it from the ServiceNow Store.
3. Select the ServiceNow IDE application (sn_glider).
4. Select **Install** or **Proceed to Update**.

   In the Review Installation Details dialog box, you can select the version to install, review any dependencies that are installed along with your application, and select whether to install the application now or later.
5. In the dialog box, select **Install**.

What to do next

Create a workspace for your applications. For more information, see Create a workspace in the ServiceNow IDE.

ServiceNow IDE uses the public npm registry (`https://registry.npmjs.org`) as its default package source. If your network blocks access to this registry, you must have access to an alternate registry to download packages and build applications in the ServiceNow IDE. If access to the public npm registry is blocked on your system, you must configure a private npm registry in your Package Manager user settings in the ServiceNow IDE. For more information, see Install an npm package from a private registry with the ServiceNow IDE.

Update the ServiceNow SDK version for an application in the ServiceNow IDE

Configure which version of the ServiceNow SDK an application uses from the ServiceNow IDE.

Before you begin

Role required: admin

About this task

From the ServiceNow IDE, you can configure which version of the ServiceNow SDK, including the ServiceNow Fluent APIs, an application uses beginning with ServiceNow SDK version 3.0.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace with an application.
3. From the Activity Bar, select the File Explorer view ( ).
4. Open the `package.json` file for the application.
5. In the `devDependencies` field, update the versions of the `@servicenow/sdk` and `@servicenow/eslint-plugin-sdk-app-plugin` packages to 3.0.0 or later.

   Example

   For example:

   ```
   "devDependencies": {
               "@servicenow/sdk": "3.0.0",
               "@servicenow/glide": "26.0.1",
               "eslint": "8.50.0",
               "@servicenow/eslint-plugin-sdk-app-plugin": "3.0.0"
   }
   ```

6. Save your changes.
7. Use one of the following keyboard shortcuts to open the command palette:
   - Windows: Ctrl-Shift-P
   - Mac: Cmd-Shift-P
8. Enter `Package Manager: Install Dependencies` and press Enter.
   The dependencies are updated in the `node_modules` directory.

Result

You can use the latest version of the ServiceNow SDK application packaging service and ServiceNow Fluent APIs in your application.

Adding applications in ServiceNow IDE

Add applications in the ServiceNow IDE to get started with development in source code.

1. Create a workspace in the ServiceNow IDE

   Create a workspace in which you can add applications in the ServiceNow IDE.

2. Get started developing a scoped application in source code by creating an application, converting an existing application, or cloning an existing application from a Git repository:

   - Create an application with the ServiceNow IDE

     Create a scoped application that supports development in source code.

   - Convert an application with the ServiceNow IDE

     Convert an existing scoped application to support development in source code.

- Clone a Git repository with the ServiceNow IDE

  Clone an existing scoped application that supports development in source code from a remote Git repository.

Create a workspace in the ServiceNow IDE

Create a workspace to view and organize the applications you're working on in the ServiceNow IDE.

Before you begin

Install the ServiceNow IDE. For more information, see Install or update the ServiceNow IDE.

Role required: admin

About this task

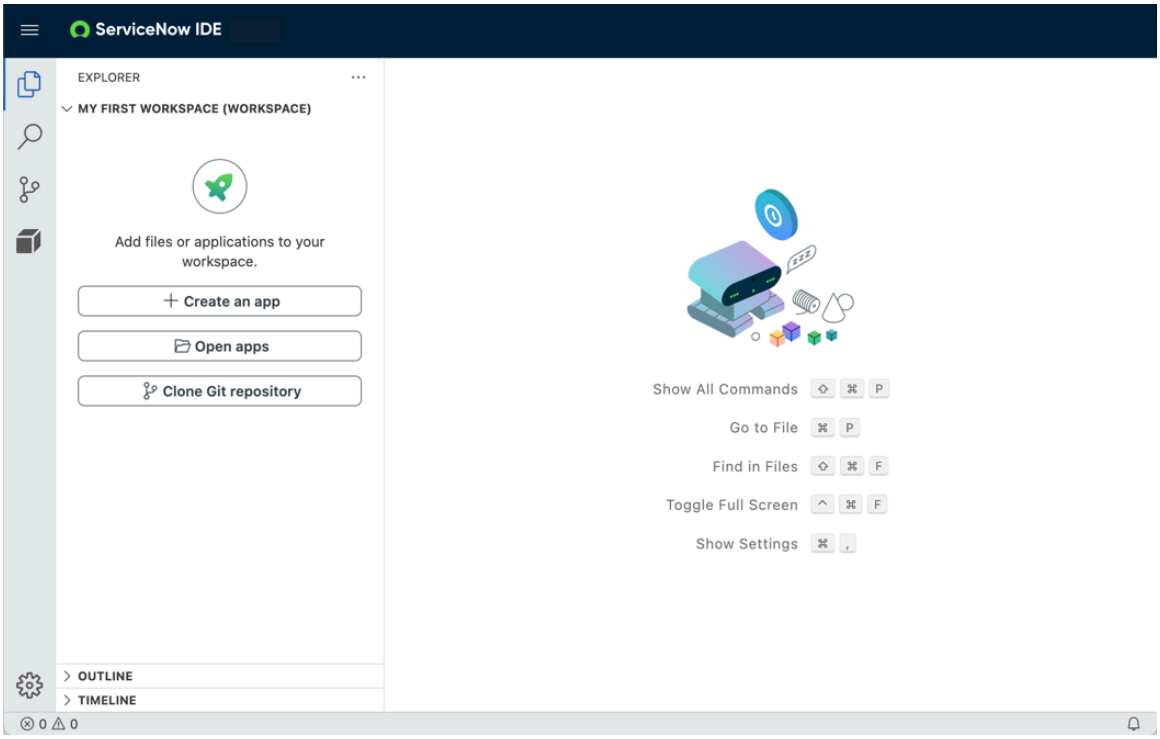
In a workspace in the ServiceNow IDE, you can add any applications that you're working on to navigate through multiple applications from one place. You can create multiple workspaces to group different sets of applications. Workspaces are specific to a user, and applications can be added or removed from a workspace at any time. When you open the ServiceNow IDE, the workspace in which you were most recently working opens. Workspaces in the ServiceNow IDE are based on workspaces in Visual Studio Code. For general information about workspaces, see the Visual Studio Code documentation website.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. From the ServiceNow IDE home page, select **Create a workspace**.

   Tip:
   You can also use the `Workspaces: Create a Workspace` command from the command palette.

3. Enter a name for the workspace and press Enter.
4. Enter a description for the workspace and press Enter.
   The workspace becomes the active workspace.



5. **Optional:** Add existing applications to the workspace.

   a. Select **Open apps**.

   b. Select an application.
   Only applications created or converted with the ServiceNow IDE or ServiceNow SDK can be added. You can add additional applications to a workspace from the main menu or the command palette with the `Workspaces: Add Application to Workspace` command.

What to do next

Create, convert, or clone an application and add it to your workspace. For more information, see Adding applications in ServiceNow IDE.

To switch the active workspace, you can browse and select other workspaces from the ServiceNow IDE home page or from the command palette with the `Workspaces: Browse Workspaces` command.

Create an application with the ServiceNow IDE

Create a scoped application to develop in source code with the ServiceNow IDE.

Before you begin

Create a workspace for your applications. For more information, see Create a workspace in the ServiceNow IDE.

Role required: admin

Procedure

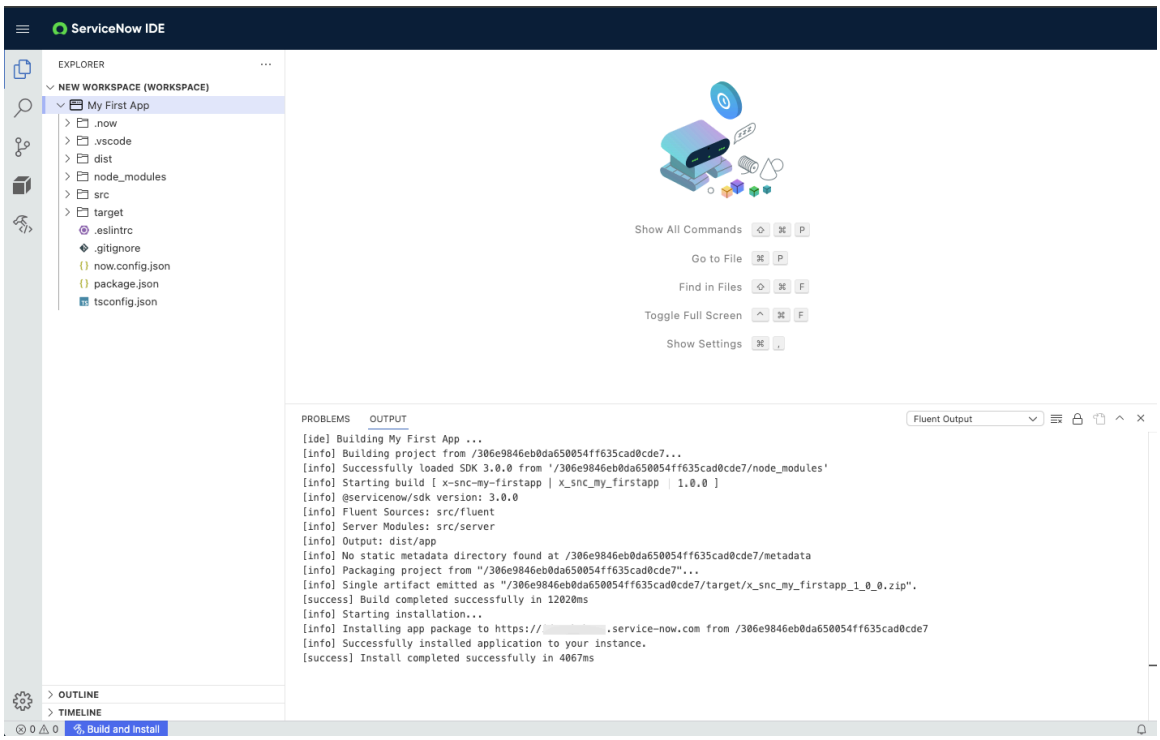
1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace.
3. Use one of the following keyboard shortcuts to open the command palette:

- Windows: Ctrl-Shift-P
- Mac: Cmd-Shift-P

4. Enter `Fluent: Create Fluent App` and press Enter.
5. Enter a name for the application and press Enter.
6. Enter a description for the application and press Enter.
7. Enter a scope for the application and press Enter.
   The scope name must be unique on the instance, begin with x_<prefix>, and be 18 characters or fewer. For more information, see Namespace identifier.
8. Enter a package name for the application and press Enter.
   The package name must adhere to Node Package Manager (npm) package naming standards.
9. Select a template that defines the default application structure.
10. **Optional:** Build and install your application to compile source code into application metadata and make your changes available across the instance. For more information, see Build and install an application in the ServiceNow IDE.

Result

A scoped application with the default application structure is added to the instance and open in your workspace. For information about the application structure, see the Application structure section of the Building applications in source code topic.



Trouble?

In the status bar, a message confirms whether the application was created. If the application creation fails, review the output logs in the panel.

> **Note:**
> For the ServiceNow IDE to install the required dependencies in an application, the public npm registry must respond with the HTTP `Access-Control-Allow-Origin` header.

What to do next

From your Git provider, create a dedicated Git repository for the application. Initialize a local Git repository for your application and push it to the remote repository. For more information, see Initialize a Git repository with the ServiceNow IDE.

In the ServiceNow IDE, start developing your application in source code with ServiceNow Fluent, writing custom JavaScript modules, or adding third-party libraries.

Convert an application with the ServiceNow IDE

Convert an existing scoped application to support development in source code with the ServiceNow IDE.

Before you begin

Create a workspace for your applications. For more information, see Create a workspace in the ServiceNow IDE.

Role required: admin

> **Note:**
> With ServiceNow IDE version 3.1.4 and earlier, the sn_glider.ide_fluent_admin role is also required to convert applications.

About this task

Existing scoped applications that weren't created with the ServiceNow IDE or ServiceNow SDK must be converted to support development in source code. Converting an application adds the necessary files and directories for developing it in source code. You can choose whether to convert existing application metadata into ServiceNow Fluent code.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace.
3. Use one of the following keyboard shortcuts to open the command palette:
   - Windows: Ctrl-Shift-P
   - Mac: Cmd-Shift-P
4. Enter `Fluent: Convert a scoped app to Fluent` and press Enter.
5. Select a scoped application from the list.

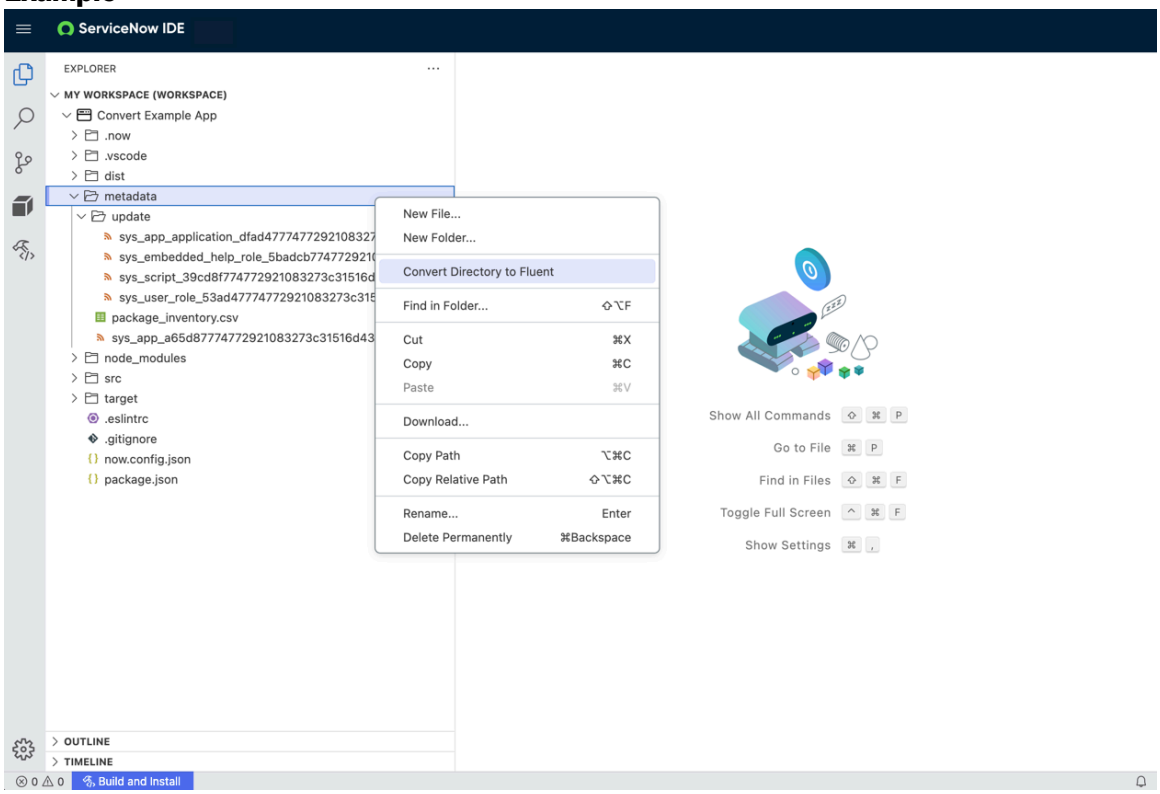
   Applications that haven't been converted are listed as `Is Fluent app: False`.

   The scoped application is added to your workspace with the default application structure but the application metadata isn't converted into ServiceNow Fluent code. For information about the application structure, see the Application structure section of the Building applications in source code topic.
6. **Optional:** To convert existing metadata into ServiceNow Fluent code, complete the following steps.

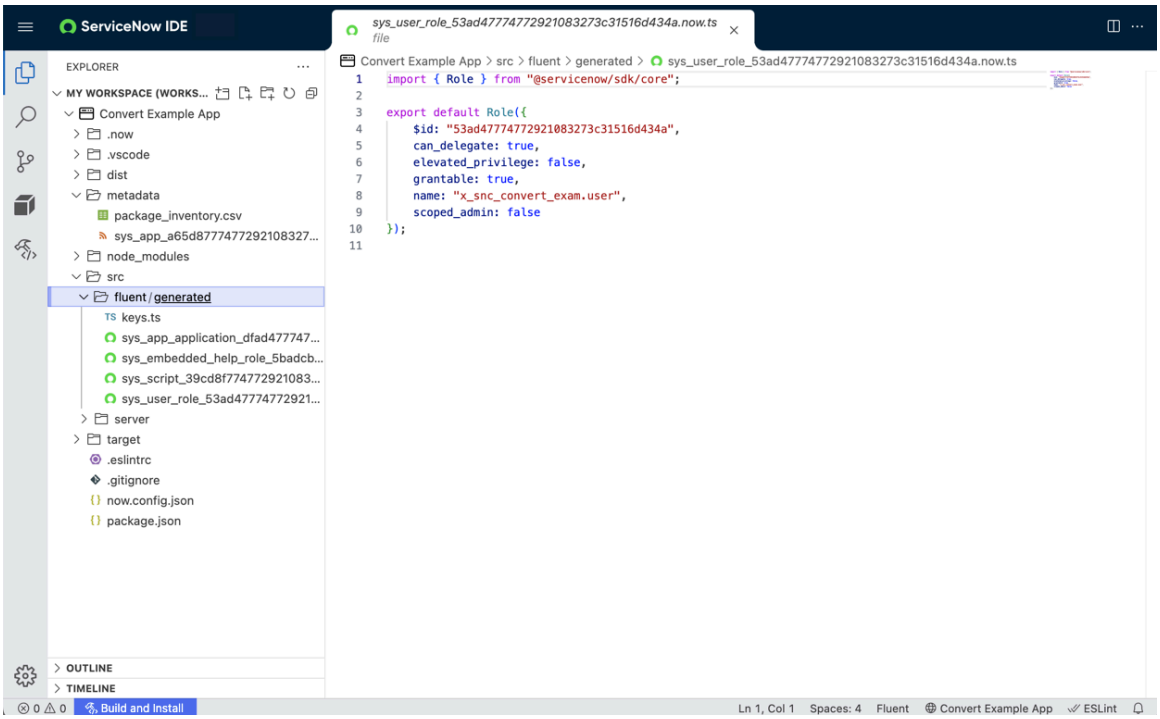
a. From the Activity Bar, select the File Explorer view ( ).

b. Right-click the `metadata` directory for the application and select **Convert Directory to Fluent**.

Example



Application metadata is defined in ServiceNow Fluent code in the `fluent/generated` directory and removed from the `metadata` directory and its sub-directories.



Note:
A limited number of metadata types, such as Metadata Snapshots [sys_metadata_link] and UX Assets [sys_ux_lib_asset], can't be represented as ServiceNow Fluent code and aren't transformed. These metadata types remain as metadata XML files in the `metadata` directory of your application.

7. **Optional:** Build and install your application to compile source code into application metadata and make your changes available across the instance. For more information, see Build and install an application in the ServiceNow IDE.

Result

The converted application is added to your workspace with the necessary files and directories to support development in source code. After installing a converted application, the **Package JSON** field of the custom application record [sys_app] contains the path to the `package.json` file for the application.

Trouble?

Note:
For the ServiceNow IDE to install the required dependencies in an application, the public npm registry must respond with the HTTP `Access-Control-Allow-Origin` header.

What to do next

From your Git provider, create a dedicated Git repository for the application. Initialize a local Git repository for your application and push it to the remote repository. For more information, see Initialize a Git repository with the ServiceNow IDE.

In the ServiceNow IDE, start developing your application in source code with ServiceNow Fluent, writing custom JavaScript modules, or adding third-party libraries.

Clone a Git repository with the ServiceNow IDE

Clone a remote Git repository to collaborate on applications in source control with the ServiceNow IDE.

Before you begin

- Create a workspace for your applications. For more information, see Create a workspace in the ServiceNow IDE.
- Set your basic or OAuth 2.0 credentials for the ServiceNow IDE to connect to your Git repository. For more information, see Connect to a Git provider using basic authentication with the ServiceNow IDE or Configure OAuth 2.0 credentials to connect to a Git provider with the ServiceNow IDE.

Role required: admin

About this task

You can clone a remote Git repository that contains applications created or converted with the ServiceNow IDE or ServiceNow SDK. The repository must contain one or more applications with `now.config.json` and `package.json` files.

Note:
ServiceNow IDE supports cloning from Git servers on Git version 2.3.2 or later.

Cloning is intended for developing an application on multiple non-production instances and managing it in a single repository. To publish an application and deploy it to a production

instance, use the Application Repository. For more information, see ServiceNow application repository.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace to which you want to add the application.
3. Use one of the following keyboard shortcuts to open the command palette:
   - Windows: Ctrl-Shift-P
   - Mac: Cmd-Shift-P
4. Enter `Git: Clone` and press Enter.
5. Enter a remote Git repository URL and press Enter.

Result

The application is added to the instance and your workspace with the files from the remote repository. Only the default branch is cloned initially. To check out another branch, you must fetch the other branches from the remote repository using the `Git: Fetch` command from the command palette.

What to do next

You can check out or create branches in the repository and push changes to the remote repository. For more information, see Using source control in the ServiceNow IDE.

Integrating source control with the ServiceNow IDE

Integrate with remote Git repositories to manage applications in source control with the ServiceNow IDE.

You can connect to a Git provider using basic or OAuth 2.0 authentication. Then, initialize a repository for an application in the ServiceNow IDE and push it to a remote repository or clone a remote repository that contains an application. After setting up authentication and connecting to a repository, you can use common Git commands to manage applications in source control.

Learn how to get started using source control in the ServiceNow IDE in the following topics.

Instance requirements

If an instance includes modified configurations for saving attachments, the following requirements must be met to perform Git operations:

- If the *glide.attachment.extensions* system property is configured, it must contain `txt,gitdata`.
- If the *glide.security.attachment_type.use_blacklist* system property is set to true, the *glide.attachment.blacklisted.extensions* system property must not contain `txt,gitdata` and the *glide.attachment.blacklisted.types* system property must not contain `text/plain,application/octet-stream`.

If you want to use custom extensions for attachments, set the following properties to custom values: *sn_glider.git.attachment.extension.text* and *sn_glider.git.attachment.extension.binary*. For more information, see ServiceNow IDE properties.

Connect to a Git provider using basic authentication with the ServiceNow IDE

Connect to a Git domain using basic authentication credentials to manage applications in source control from the ServiceNow IDE.

Before you begin

- From your Git provider, generate a personal access token to use for basic authentication to the Git domain . You must provide your Git user name and personal access token when configuring your credentials for the ServiceNow IDE.
- Create a dedicated Git repository for an application in a Git provider such as GitHub, GitLab, Bitbucket, or Azure Repos.

Role required: admin

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Use one of the following keyboard shortcuts to open the command palette:
   - Windows: Ctrl-Shift-P
   - Mac: Cmd-Shift-P
3. Enter `Git: Set IDE Git credentials` and press Enter.
4. From the New Git credential form, select **Basic auth**.
5. On the form, fill in the fields.

New Git credential form

Field	Description
Git repository URL	The HTTPS URL of a Git repository associated with your Git credentials.
Git username	Your Git user name.
Personal access token	A personal access token that you generate from your Git provider.

6. Select **Submit**.

Result

Your Git credentials are associated with your user on the instance and used for all repositories in the domain from the Git repository URL. If you add different credentials for a repository in the same domain, the new credentials are used and the previous credentials are set to inactive.

What to do next

After initializing or cloning a repository, you can begin using source control. For more information, see Using source control in the ServiceNow IDE.

To manage existing Git credentials, use the `Git: Manage Git credentials` command from the command palette.

Connect to a Git provider using OAuth 2.0 with the ServiceNow IDE

Set up an OAuth 2.0 application registry and credentials to connect to your Git provider from the ServiceNow IDE.

Configure an OAuth 2.0 application registry for the ServiceNow IDE

Configure how the client ID and secret are sent to the OAuth 2.0 provider associated with your Git provider.

Before you begin

Create an OAuth application with your Git provider and configure it to redirect to your instance. In this OAuth application, use your instance URL as the homepage URL and `https://<instance>/oauth_redirect.do` for the authorization callback URL. GitHub, GitLab, Bitbucket, and Azure Repos are supported by default.

Important:
For Azure Repos, the maximum length of the **Client Secret** field must be updated to 2048 before you add the secret.

1. Navigate to **All** > **System Definition** > **Tables**.
2. Filter the table by entering `oauth_entity` for the **Name** field.
3. Select the Application Registries [oauth_entity] table.
4. In the Columns related list, locate the Client Secret column and enter `2048` as the value of its **Max length** attribute.
5. Select **Update**.

Role required: admin

About this task

To use OAuth 2.0 authentication with the ServiceNow IDE, you must register the OAuth application you created from your Git provider. Follow this procedure to configure an OAuth application registry [oauth_entity] on your instance.

Procedure

1. Navigate to **All** > **System OAuth** > **Application Registry** and then select **New**.
2. On the interceptor page, select **Connect to a third-party OAuth provider**.
3. On the form, fill in the fields.

   For additional information about fields on the form, see Connect to a third-party OAuth provider.

Application Registries form

Field	Description
Name	A unique name for the third-party OAuth connection.
Client ID	The client ID of the OAuth application in your Git provider.
Client Secret	The client secret of the OAuth application in your Git provider.
OAuth API Script	The script used to customize request and response to the external OAuth provider. Select one of the following depending on your Git provider: ○ GitHub: OauthAPIScriptForGitHub ○ GitLab: OauthAPIScriptForGitLab ○ Bitbucket: OauthAPIScriptForBitbucket ○ Azure Repos: OauthAPIScriptForAzureRepos

Field	Description
	Note: To use other Git providers, you can use these scripts as examples to create your own OAuth API script for your OAuth provider. The name of the script must begin with "Oauth".
Default Grant type	The default grant type used to establish the token. Select **Authorization code**. An authorization code is granted to the client to obtain an access token, which is then used to obtain access to the resource.
Refresh Token Lifespan	The time, in seconds, that the refresh token is valid.
Application	The application scope that contains this record.
Accessible from	An option to make the application registry accessible from all application scopes or from this scope only.
Active	An option for turning the OAuth application on or off.
Authorization URL	The OAuth authorization code endpoint for your Git domain. For example: ○ GitHub: `https://github.com/login/oauth/authorize` ○ GitLab: `https://gitlab.com/oauth/authorize` ○ Bitbucket: `https://bitbucket.org/site/oauth2/authorize` ○ Azure Repos: `https://app.vssps.visualstudio.com/oauth2/authorize`
Token URL	The OAuth server token endpoint for your Git domain. For example: ○ GitHub: `https://github.com/login/oauth/access_token` ○ GitLab: `https://gitlab.com/oauth/token` ○ Bitbucket: `https://bitbucket.org/site/oauth2/access_token` ○ Azure Repos: `https://app.vssps.visualstudio.com/oauth2/token`
Redirect URL	The OAuth callback endpoint. If empty, the instance auto-generates a value of `https://<instance>/oauth_redirect.do`.
Use mutual authentication	An option to use mutual authentication. Leave this option cleared to turn off using mutual authentication for token request and revocation.
Send Credentials	The way in which the OAuth client populates the client credentials in the request.

Field	Description
	○ For GitHub, GitLab, or Bitbucket, select **As Basic Authorization header**. ○ For Azure Repos, select **In Request Body (Form URL-Encoded)**.

4. Select **Submit**.

What to do next

Developers using the ServiceNow IDE must configure their own OAuth 2.0 credentials.

Configure OAuth 2.0 credentials to connect to a Git provider with the ServiceNow IDE

Connect to a Git domain using OAuth 2.0 credentials to manage applications in source control from the ServiceNow IDE.

Before you begin

- An administrator must configure how the client ID and secret are sent to the OAuth 2.0 provider associated with your Git provider. For more information, see Configure an OAuth 2.0 application registry for the ServiceNow IDE.
- Create a dedicated Git repository for an application in a Git provider such as GitHub, GitLab, Bitbucket, or Azure Repos.

Role required: admin

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Use one of the following keyboard shortcuts to open the command palette:
   - Windows: Ctrl-Shift-P
   - Mac: Cmd-Shift-P
3. Enter `Git: Set IDE Git credentials` and press Enter.
4. From the New Git credential form, select **OAuth**.
5. On the form, fill in the fields.

New Git credential form

Field	Description
Git repository URL	The HTTPS URL of a Git repository associated with your Git credentials.
Git username	Your Git user name.
Select an OAuth profile	The OAuth 2.0 credentials for your Git provider. The OAuth profile is created with the application registry, which must be configured by an administrator.

6. Select **Submit**.

   The first time you use a Git command, you're prompted to authorize your user from the Git provider.

Result
Your Git credentials are associated with your user on the instance and used for all repositories in the domain from the Git repository URL. If you add different credentials for a repository in the same domain, the new credentials are used and the previous credentials are set to inactive.

What to do next
After initializing or cloning a repository, you can begin using source control. For more information, see Using source control in the ServiceNow IDE.

To manage existing Git credentials, use the `Git: Manage Git credentials` command from the command palette.

Configure a MID Server to use source control with the ServiceNow IDE

Configure a MID Server to use source control with the ServiceNow IDE if your Git provider is behind a firewall.

Before you begin
Install a MID Server with a REST capability. For more information, see Installing the MID Server and Configure MID Server capabilities.

Role required: admin

Procedure
1. Change the application scope of the instance to ServiceNow IDE.
   a. In the Unified Navigation, select the globe icon ( ).
   b. Select **Application scope**.
   c. Select **ServiceNow IDE**.
2. Navigate to **All** > **MID Server** > **Applications**.
3. Select **New**.
4. On the form, fill in the fields.

MID Server Application form

Field	Description
Name	A name for the MID Server Application.
Default MID Server	The MID Server configured with a REST capability.
Application	The ServiceNow IDE.
Included in application ALL	An option to include this MID Server Application in the definition of ALL for a MID Server.

5. Select **Submit**.

What to do next
If you haven't already, configure basic or OAuth 2.0 authentication to connect to a Git domain or repository. For more information, see Connect to a Git provider using basic authentication with the ServiceNow IDE or Connect to a Git provider using OAuth 2.0 with the ServiceNow IDE.

Note:
The MID Server user must have the sn_glider.ide_git_user role or admin role to perform Git operations in the ServiceNow IDE. For more information, see Create the MID Server user and grant the role and ServiceNow IDE MID Server User [sn_glider.ide_git_user].

Initialize a Git repository with the ServiceNow IDE

Initialize a local Git repository for an application and push it to a remote Git repository to manage an application in source control.

Before you begin

- Create or convert an application with the ServiceNow IDE. For more information, see Create an application with the ServiceNow IDE or Convert an application with the ServiceNow IDE.
- Create a dedicated Git repository for the application from your Git provider.
- Set your basic or OAuth 2.0 credentials for the ServiceNow IDE to connect to your Git repository. For more information, see Connect to a Git provider using basic authentication with the ServiceNow IDE or Configure OAuth 2.0 credentials to connect to a Git provider with the ServiceNow IDE.

Role required: admin

About this task

An application on an instance can be connected to only one repository at a time. To clone an application that exists in a remote Git repository, see Clone a Git repository with the ServiceNow IDE.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace with an application that isn't connected to a Git repository.
3. From the Activity Bar, select the Source Control view ( ).
4. Select **Initialize Repository** or use the `Git: Initialize Repository` command from the command palette.
5. Select the application for which you want to initialize a Git repository and press Enter.
6. Select **main** as the default branch name or enter another name and press Enter.
7. Select the Stage All Untracked Changes icon ( ).
8. Enter a commit message and select the Commit icon ( ).
9. Select the More actions menu icon ( ) and select **Push**.
10. Enter a remote repository URL and press Enter.

Result

The application is available in the remote repository.

Trouble?

If your Git credentials aren't configured or are inactive, the application isn't pushed to the remote repository. When prompted to configure your credentials, select **Configure** to configure your Git credentials and then try again.

What to do next

You can check out or create branches in the repository and push changes to the remote repository. For more information, see Using source control in the ServiceNow IDE.

Using source control in the ServiceNow IDE

Use Git commands and other source control features in the ServiceNow IDE to manage changes to an application across a development team.

Role required: admin

Git commands

After initializing a local Git repository in the ServiceNow IDE, you can perform several Git commands from the Source Control view ( ) or command palette, including but not limited to the following commands.

Git commands in the ServiceNow IDE

Command	Description
Checkout to (> **Checkout to...**) Command palette: **Git: Checkout to...**	Check out another branch from the repository. Select the branch from the list or create a branch and push it to the remote repository. Only one branch per repository can be checked out at a time on an instance (or developer sandbox).
Clone (> **Clone**) Command palette: **Git: Clone**	Clone a remote repository to add an application to your workspace. For more information, see Clone a Git repository with the ServiceNow IDE.
Commit ( ) Command palette: **Git: Commit**	Commit your staged changes to the local repository. Enter a commit message to describe your changes.
Create branch (> **Branch** > **Create branch...**) Command palette: **Git: Create branch...**	Add a branch to a repository and check it out.
Discard ( ) Command palette: **Git: Discard Changes**	Discard changes to undo modifications to an application.
Fetch (> **Fetch**) Command palette: **Git: Fetch**	Fetch to sync commits and branches from the remote repository into the local repository.
Pull (> **Pull**) Command palette: **Git: Pull**	Pull to merge the latest changes from the remote repository into the local repository.
Push (> **Push**)	Push your committed changes to the remote repository.

Git commands in the ServiceNow IDE (continued)

Command	Description
Command palette: **Git: Push**	
Stage (+) Command palette: **Git: Stage Changes**	Stage the changes in your working directory that you want to commit. When you stage changes, files move from the Changes list to the Staged Changes list.
Stash (··· > **Stash**) Command palette: **Git: Stash**, **Git: Pop**, **Git: Drop**, **Git: List**, **Git: Apply**, **Git: Clear**	Stash all uncommitted changes to save them in your working directory locally and come back to them later. The pop, drop, list, apply, and clear subcommands are supported when stashing. Select **Stash** > **Apply** to reapply your changes in your working directory.

Merge conflicts

If there's a conflict between local and remote changes in a file, the file is listed under Merge conflicts in the Source Control view ( ). You can review conflicts in the editor and resolve them by accepting the current or incoming change, both, or manually editing the file.

Commit history

The commit history for a branch is listed in the Commits section of the Source Control view ( ) and includes details about the commits. To see the commit history for a file, navigate to the File Explorer view ( ), select the file, and expand the Timeline section. When you select a commit, a Diff editor comparing the changes opens.

Repository changes

To update the remote repository an application is connected to, you can use the `Git: Update remote origin` command from the command palette and enter a different remote repository URL.

Developing applications with the ServiceNow IDE

Create and develop applications in source code with the ServiceNow IDE. Build applications to make your changes available across an instance and collaborate with users of any skill set.

In the ServiceNow IDE, you create scoped applications or convert existing applications and develop them in source code using ServiceNow Fluent to define application metadata [sys_metadata]. Optionally, you can create JavaScript modules and use third-party libraries to organize and reuse code within scoped applications.

When your changes are saved and ready, build an application to compile the source code to application metadata and make your changes available to users across the instance. As other users modify the application metadata, you can synchronize the changes into the source code.

Learn how to get started developing applications with the ServiceNow IDE in the following topics.

System requirements

ServiceNow IDE uses the public npm registry (`https://registry.npmjs.org`) as its default package source. If your network blocks access to this registry, you must have access to an alternate registry to download packages and build applications in the ServiceNow IDE. If access to the public npm registry is blocked on your system, you must configure a private npm registry in your Package Manager user settings in the ServiceNow IDE. For more information, see Install an npm package from a private registry with the ServiceNow IDE.

Related topics

- Integrating source control with the ServiceNow IDE
- ServiceNow Fluent API reference

Synchronizing applications in the ServiceNow IDE

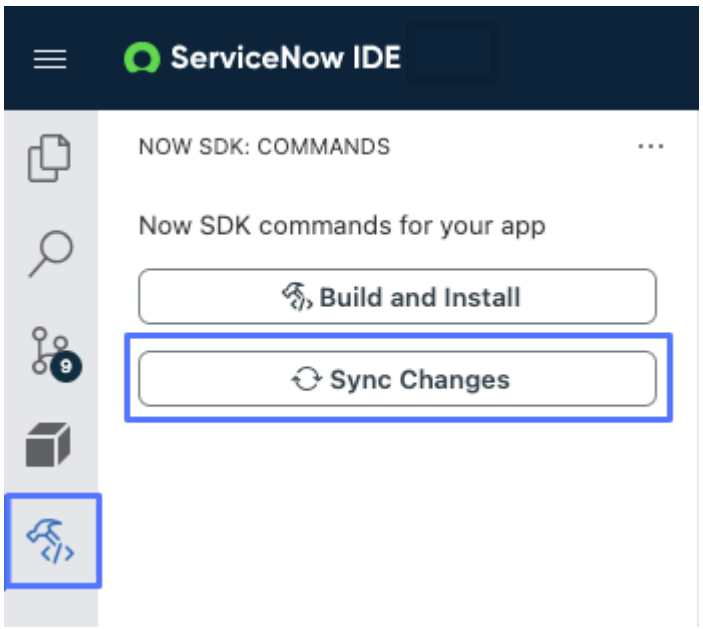
Synchronizing an application in the ServiceNow IDE downloads and transforms application metadata into ServiceNow Fluent code.

ServiceNow Fluent supports two-way synchronization, which allows changes to metadata to be synced from other ServiceNow AI Platform user interfaces into source code. When you build and install an application, changes to source code are compiled into metadata across the instance. This process enables collaborating with users of different skill sets because developers can modify the ServiceNow Fluent code while others modify the application metadata across the platform.

In the ServiceNow IDE, when metadata in an application is changed, you're prompted to sync the application to get the latest changes. If metadata changes are detected in application, syncing is required before you can install the application across the instance.

Note:
If you ignore the notification to sync an application, you aren't prompted to do so again for the remainder of the session. If you try to install the application or refresh the browser, you are prompted to sync again.

You can sync an application on demand from the Now SDK view ( ) or from the command palette with the `Fluent: Sync Fluent App with changed metadata` command.



Synchronization process in the ServiceNow IDE

The following list explains the stages of the synchronization process for applications in the ServiceNow IDE:

1. Before the first time an application is synced:
   - You should initialize a repository.
   - You should stage any changes to the application to retain them when the application is synced.
2. The first time that an application is synced:
   - New metadata that isn't defined in source code is transformed into source code in the `src/fluent/generated` directory.
   - Changes to metadata that is defined in source code are downloaded into source code in the `src/fluent` and `src/fluent/generated` directories.
   - Changes to existing metadata XML that hasn't been converted into source code are downloaded into the `metadata` directory.
3. Subsequent times that an application is synced:
   - Only the changes made since the last time the application was synced are downloaded and transformed.
   - Optionally, you can synchronize all metadata in the application with the `Fluent: Force Sync of Fluent App with all metadata` command from the command palette. Before running this command, you should stage any changes to retain them.

Related topics

ServiceNow Fluent

Define application metadata in code with ServiceNow Fluent in the ServiceNow IDE

Define application metadata in code with ServiceNow Fluent in the ServiceNow IDE.

Before you begin

Create, convert, or clone an application and add it to your workspace. For more information, see Adding applications in ServiceNow IDE.

Role required: admin

About this task

ServiceNow Fluent is a domain-specific language (DSL) based on TypeScript for defining the metadata files [sys_metadata] that make up applications and includes APIs for the different types of metadata, such as tables, roles, ACLs, business rules, and Automated Test Framework tests. For more information about ServiceNow Fluent APIs and examples, see ServiceNow Fluent API reference. The ServiceNow IDE has language processing and validation for ServiceNow Fluent APIs and applications by default.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace with an application.
3. From the Activity Bar, select the File Explorer view ( ).
4. In the `src/fluent` directory, add a file with the `.now.ts` extension.
   You can define application metadata in a single file or as many `.now.ts` files as you want and organize files in directories within the `fluent` directory.

Tip:
In the `fluent` directory, you can refer to an example file named `index.now.ts`.

5. In the `.now.ts` file, use objects in the ServiceNow Fluent APIs to define metadata in the application.

   Example

   For example, to create a table [sys_db_object] in the application, use the Table API. The following example includes a simple table for to-do lists using a *`Table`* object with the necessary properties and values.

```
export const x_snc_example_to_do = Table({
    name: 'x_snc_example_to_do',
    schema: {
        deadline: DateColumn({ label: 'Deadline' }),
        state: StringColumn({
            label: 'State',
            choices: {
                ready: { label: 'Ready' },
                completed: { label: 'Completed' },
                inProgress: { label: 'In Progress' },
            },
        }),
        task: StringColumn({ label: 'Task', maxLength: 120 }),
    },
})
```

Note:
You can use the Record API to define application metadata that doesn't have a dedicated API.

6. Add the required imports for the APIs from `@servicenow/sdk/core`.

   Example

   For example, to define a table with a date column and string column, import the corresponding objects to use from the Table API:

```
import { Table, DateColumn, StringColumn }
 from '@servicenow/sdk/core'
```

7. **Optional:** For ServiceNow Fluent APIs that use server-side scripts, such as business rules, import code from JavaScript modules to call them from an object's *`script`* property.

   Example
   This example imports the `showStateUpdate` function, which can then be referenced from the *`script`* property.

```
import { showStateUpdate } from '../server/script.js'
```

8. Save your changes.
9. **Optional:** Build and install your application to compile source code into application metadata and make your changes available across the instance. For more information, see Build and install an application in the ServiceNow IDE.

Trouble?

Tip:
You can use the following directives in a code comment to help manage your code:

- `@fluent-ignore`: Suppresses ServiceNow Fluent diagnostic warnings and errors in the following line of code.
- `@fluent-disable-sync`: Turns off syncing changes to a ServiceNow Fluent object. Use before a call expression (for example, `Record({ ... })`) to turn off syncing for that object and its child objects. Only use this directive if you want to ignore changes made outside of the source code to the object and never update it when syncing.
- `@fluent-disable-sync-for-file`: Turns off syncing changes to a ServiceNow Fluent file (`.now.ts`). Use in the first line of the file to turn off syncing for all code in the file. Only use this directive if you want to ignore changes made outside of the source code to the file and never update it when syncing.

Example: Defining application metadata in source code with ServiceNow Fluent

In files with the `.now.ts` extension, use objects in the ServiceNow Fluent APIs to define metadata in the application. You must also include the required imports for the APIs from `@servicenow/sdk/core`. For objects with server-side scripts, such as the *`BusinessRule`* object, you can import and use code from JavaScript modules.

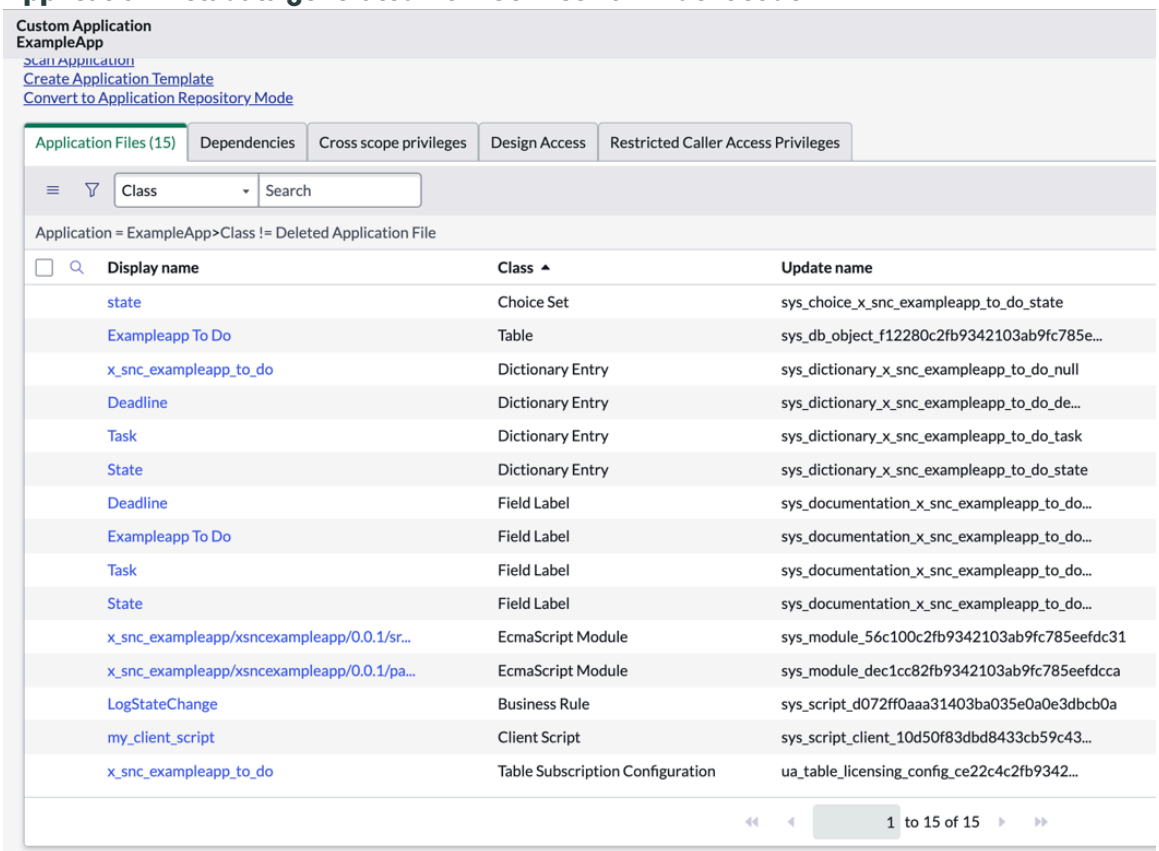
The following example includes the definitions of a table, client script, and business rule in the application. The client script uses a script from the `client-script.js` file. The business rule uses a function from the `script.js` JavaScript module.

```
import '@servicenow/sdk/global'
import { BusinessRule, ClientScript, DateColumn, StringColumn,
 Table } from '@servicenow/sdk/core'
import { showStateUpdate } from '../server/script.js'

//creates todo table, with three columns (deadline, status and task)
export const x_snc_example_to_do = Table({
    name: 'x_snc_example_to_do',
    schema: {
        deadline: DateColumn({ label: 'Deadline' }),
        state: StringColumn({
            label: 'State',
            choices: {
                ready: { label: 'Ready' },
                completed: { label: 'Completed' },
                inProgress: { label: 'In Progress' },
            },
        }),
        task: StringColumn({ label: 'Task', maxLength: 120 }),
    },
})

//creates a client script that pops up 'Table loaded successfully!!' message everytime todo
 record is loaded
ClientScript({
    $id: Now.ID['cs0'],
    name: 'my_client_script',
    table: 'x_snc_example_to_do',
    active: true,
```

```
    appliesExtended: false,
    global: true,
    uiType: 'all',
    description: 'Custom client script generated by Now SDK',
    isolateScript: false,
    type: 'onLoad',
    script: Now.include('../client/client-script.js'),
})

//creates a business rule that pops up state change message whenever a todo record is
 updated
BusinessRule({
    $id: Now.ID['br0'],
    action: ['update'],
    table: 'x_snc_example_to_do',
    script: showStateUpdate,
    name: 'LogStateChange',
    order: 100,
    when: 'after',
    active: true,
})
```

After building the application, this source code generates the following application metadata files on the instance.

Application metadata generated from ServiceNow Fluent code



Related topics

ServiceNow Fluent

Create and use JavaScript modules in applications in the ServiceNow IDE

Optimize your codebase by defining reusable code blocks with JavaScript modules in the ServiceNow IDE.

Before you begin

Create, convert, or clone an application and add it to your workspace. For more information, see Adding applications in ServiceNow IDE.

Note:
To use TypeScript in modules, select TypeScript as the template type when creating an application.

Role required: admin

About this task

To learn about support for using JavaScript modules in scoped applications, including some limitations, see JavaScript modules and third-party libraries. For general information about the syntax used to create JavaScript modules, see the JavaScript modules page on the MDN Web Docs website.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace with an application.
3. From the Activity Bar, select the File Explorer view ( ).
4. In the `src/server` directory of the application, create a JavaScript or TypeScript file to contain the module code you want to reuse.
5. **Optional:** Import server APIs or script includes to call them from your module.

   (Optional) Glide APIs can be imported from the `@servicenow/glide` package or their namespace in the package. Script includes can be imported from their application scope or the global scope in the `@servicenow/glide` package.

   For example:

   ```
   import { API } from "@servicenow/glide";
   import { API } from "@servicenow/glide/<namespace>";
   import { ScriptInclude } from "@servicenow/glide/<scope>";
   import { global } from "@servicenow/glide/global";
   ```

6. In the module, identify the code to export with *`export`* statements.

   You can use a named export or default export. Named exports can be variables, constants, functions, or classes whereas default exports can be functions or classes only.

   Example
   The following example is one way of adding a named export for multiple features (a function and a variable) in a module:

   ```
   export { myFunction, myVariable };
   ```

7. Use code from the exported module in other modules or server-side scripts.
8. Save your changes.

9. From the Status Bar, select **Build and Install**.

The active file that's open in the editor determines which application to build. If no files are open, select the application to build when prompted.

After building, the modules are added to the EcmaScript Module [sys_module] table.

What to do next

To use third-party libraries in a JavaScript module, see Use third-party libraries in applications in the ServiceNow IDE.

Related topics

JavaScript modules and third-party libraries

Use third-party libraries in applications in the ServiceNow IDE

Call third-party libraries in your application to use existing open-source functionality with the ServiceNow IDE.

Before you begin

Create a JavaScript module. For more information, see Create and use JavaScript modules in applications in the ServiceNow IDE.

To install and use packages from private registries, you must configure your Package Manager user settings in the ServiceNow IDE. For more information, see Install an npm package from a private registry with the ServiceNow IDE.

Role required: admin

About this task

Third-party libraries are added to applications as JavaScript modules. For general information about the syntax used to create JavaScript modules, see the JavaScript modules page on the MDN Web Docs website.

Important:
You can't use third-party libraries that rely on unsupported functionality, such as unsupported APIs or ECMAScript features. For more information about unsupported functionality, see JavaScript modules and third-party libraries.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace with an application.
3. From the Activity Bar, select the File Explorer view ( ).
4. Open the `package.json` file for the application.
5. Add the `dependencies` field with the package name and version of any third-party libraries to use.

Example

```
"dependencies": {
      "<package name>": "<version>"
  }
```

6. Install the third-party library packages.

   a. Use one of the following keyboard shortcuts to open the command palette:

      - Windows: Ctrl-Shift-P
      - Mac: Cmd-Shift-P

   b. Enter `Package Manager: Install Dependencies` and press Enter.

   Packages are installed in the `node_modules` directory.

7. In a JavaScript module, import the library using an *`import`* statement.

   Example

   In this example, the module includes a namespace import for the lodash module.

   ```
   import * as lodash from "lodash"
   ```

   In this example, the module includes a named import for the camelCase function in the lodash module.

   ```
   import camelCase from 'lodash'
   ```

8. Call code imported from the library in your module to reuse it.
9. From the Status Bar, select **Build and Install**.

   The active file that's open in the editor determines which application to build. If no files are open, select the application to build when prompted.

   After building, the modules are added to the EcmaScript Module [sys_module] table.

Related topics

JavaScript modules and third-party libraries

Install an npm package from a private registry with the ServiceNow IDE

Install Node Package Manager (npm) packages from a private registry as dependencies in your application to use them as third-party libraries.

Before you begin

Role required: admin

About this task

To install packages from private registries, you must configure your Package Manager user settings in the ServiceNow IDE. Then you can choose in which applications to install packages.

ServiceNow IDE uses the public npm registry (`https://registry.npmjs.org`) as its default package source. If your network blocks access to this registry, you must have access to an alternate registry to download packages and build applications in the ServiceNow IDE. If access to the public npm registry is blocked on your system, you must configure a private npm registry in your Package Manager user settings in the ServiceNow IDE.

Note:
To install packages from a private registry, the registry must respond with the HTTP `Access-Control-Allow-Origin` header.

Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace.
3. Use one of the following keyboard shortcuts to open the command palette:
   - Windows: Ctrl-Shift-P
   - Mac: Cmd-Shift-P
4. Enter `Preferences: Open User Settings (JSON)` and press Enter.
5. Specify a private registry as the default registry or a scoped registry from which you can install packages.

   Note:
   To install packages from public registries, you can specify a public registry as the default registry or a scoped registry in addition to any private registries.

6. Provide your credentials for accessing the private registry.

Option	Description
Basic authentication	Provide a user name and password to access the default registry that you specified. `"package-manager.basicAuth": [ { "registry": "<private-registry-url>", "user": "<user-name>", "pass": "<password>" }, ... ]`
Legacy authentication	Provide a legacy token to access the default registry that you specified. Legacy tokens are basic authentication credentials encoded in Base64 format. `"package-manager.legacyAuth": [ { "registry": "<private-registry-url>", "token": "<legacy-token>" }, ... ]`
Token authentication	Provide a bearer token to access the default registry that you specified. `"package-manager.tokenAuth": [ { "registry": "<private-registry-url>", "token": "<token>" }, ... ]`

7. Save your changes to the `settings.json` file.
8. Install packages from the configured registries.

   a. Use one of the following keyboard shortcuts to open the command palette:
      - Windows: Ctrl-Shift-P
      - Mac: Cmd-Shift-P

   b. Enter `Package Manager: Install Dependencies` and press Enter.

   c. Select an application to install packages in as dependencies.

   Packages are installed in the `node_modules` directory.

Example: User settings for private registry access

In the following example, a user configured access to a private registry using basic authentication. The user also configured their settings to install packages from a public registry with the `@example` scope.

```
{
    "files.autoSave": "off",
    "package-manager.defaultRegistry": "<private-registry-url>",
    "package-manager.basicAuth": [
        {
        "registry": "<private-registry-url>",
            "user": "<user-name>",
            "pass": "<password>"
        }
    ],
    "package-manager.scopedRegistries": [
        {
            "scope": "@example",
            "registry": "<public-registry-url>"
        }
    ]
}
```

What to do next

Use the packages that you installed as third-party libraries in your application. For more information, see Use third-party libraries in applications in the ServiceNow IDE.

Create an application file from the Metadata Explorer

Create an application file in an embedded ServiceNow AI Platform user interface from the Metadata Explorer in the ServiceNow IDE.

Before you begin

Create, convert, or clone an application and add it to your workspace. For more information, see Adding applications in ServiceNow IDE.

Role required: admin

About this task

If you need to create some application metadata outside of source code, you can create application files from the Metadata Explorer within the ServiceNow IDE. To create an application

file from the ServiceNow IDE, an embedded ServiceNow AI Platform user interface opens in the editor.

Procedure

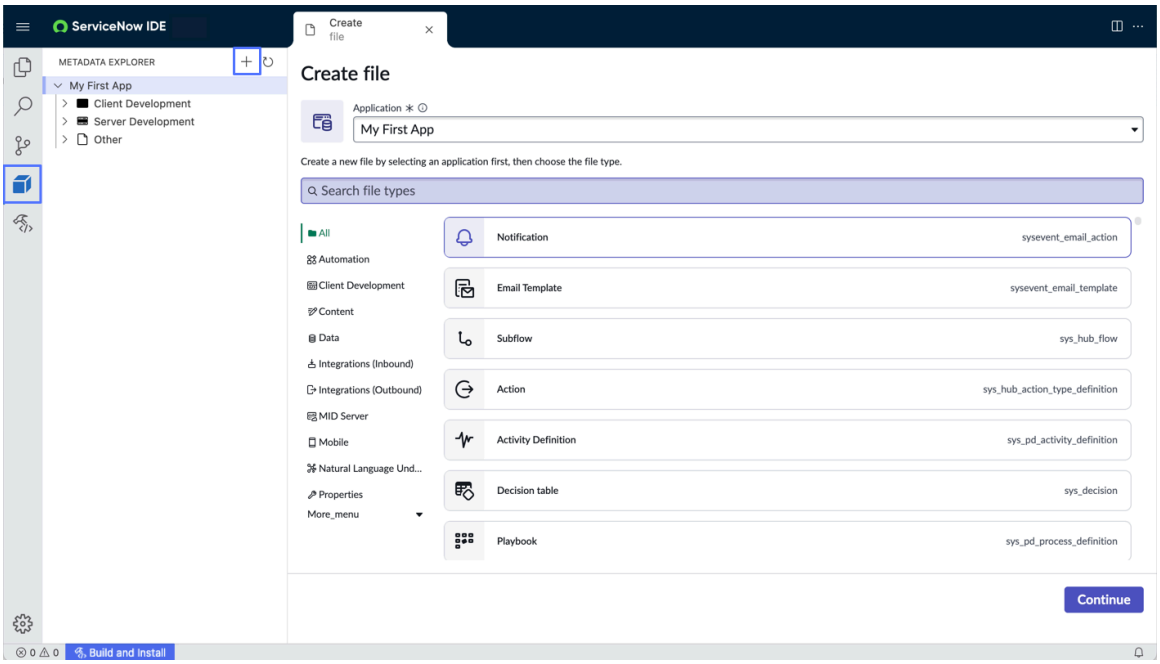
1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace with an application.
3. From the Activity Bar, select the Metadata Explorer view ( ).
4. Select the Create New File icon ( ).

> **Tip:**
> Alternatively, you can right-click an existing metadata category or file type in an application and select **Create New File** to create an application file of that category or type.

5. From the Application menu, select an application from your workspace.
6. From the list of file types, select the type of file to create.

> **Tip:**
> Search for a file type or select a category to find the file type you need.

Example



7. Select **Continue**.
8. In the embedded ServiceNow AI Platform user interface, fill in the fields to create the application file.
   The user interface and fields vary depending on the type of file that you created.
9. Select **Submit**.

Build and install an application in the ServiceNow IDE

Build an application to compile its source code and install application changes across an instance from the ServiceNow IDE.

Before you begin

Create, convert, or clone an application and add it to your workspace. For more information, see Adding applications in ServiceNow IDE.

Role required: admin

About this task

When you build an application, ServiceNow Fluent code is compiled into application metadata, and modules in the application are added to the EcmaScript Module [sys_module] table on the instance.

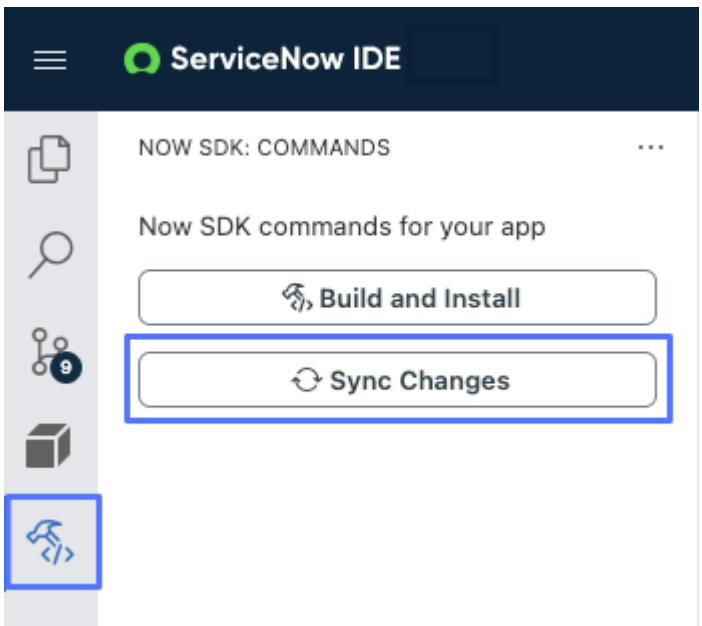
Procedure

1. Navigate to **All** > **App Development** > **ServiceNow IDE**.
2. Open a workspace with an application.
3. Synchronize changes made to the application metadata from across the instance.
   Syncing an application confirms you have the latest changes before making additional changes. Only the changes made since the last time the application was synced are downloaded and transformed. For more information, see Synchronizing applications in the ServiceNow IDE.

   a. From the Activity Bar, select the Now SDK view ( ).

   b. Select **Sync Changes**.
   The active file that's open in the editor determines which application to synchronize. If no files are open, select the application to synchronize when prompted.

   Example



4. From the Activity Bar, select the File Explorer view ( ).

5. Modify your application in one of the following ways and save your changes:
   - Define application metadata in code with ServiceNow Fluent in the ServiceNow IDE
   - Create and use JavaScript modules in applications in the ServiceNow IDE
   - Use third-party libraries in applications in the ServiceNow IDE
6. From the Status Bar, select **Build and Install**.

   The active file that's open in the editor determines which application to build. If no files are open, select the application to build when prompted.

   Trouble?
   In the status bar, you can see a message that confirms whether the build was successful or failed. If the build fails, review the output logs and any problems in your code from the panel.
7. **Optional:** Review your changes as application metadata.

   a. From the Activity Bar, select the Metadata Explorer view ( ).

   b. Open the metadata files that you changed and review your changes from the embedded ServiceNow AI Platform user interfaces.

   Tip:
   To see your changes in source code and metadata side by side, select the Split Editor icon ( ) or right-click a tab and select one of the **Split** options.

Result

The application is updated across the instance to reflect your changes. Other users can modify the application metadata simultaneous with modifications to the source code. Also, other developers can reuse module code in other modules or scripts within an application.

Trouble?

If you need to ensure that the metadata on the instance matches the metadata in the installation package, you can uninstall and reinstall the application on the instance using the `Fluent: Reinstall Fluent App` command from the command palette. For more information, see ServiceNow IDE commands.

Warning:
Changes to metadata on the instance that haven't been synced into your application in the ServiceNow IDE are removed.

What to do next

Use source control to stage, commit, and push your changes to a remote Git repository. For more information, see Using source control in the ServiceNow IDE.

ServiceNow IDE reference

Reference topics include information about ServiceNow Fluent APIs and ServiceNow IDE properties, roles, and more.

ServiceNow Fluent API reference

Use ServiceNow Fluent APIs to define the metadata that makes up scoped applications in source code with the ServiceNow IDE or ServiceNow SDK.

ServiceNow IDE commands

Use commands installed with ServiceNow IDE.

ServiceNow IDE properties

Use system properties installed with ServiceNow IDE.

ServiceNow IDE roles

Use roles installed with ServiceNow IDE.

ServiceNow IDE commands

Use commands from the command palette to perform many actions in the ServiceNow IDE.

The command palette includes commands unique to the ServiceNow IDE for creating and managing applications and commands from Visual Studio Code, such as Git commands for using source control. The following commands are unique to the ServiceNow IDE.

To open the command palette, use one of the following keyboard shortcuts:

- Windows: Ctrl-Shift-P
- Mac: Cmd-Shift-P

To close the command palette, press the Escape key.

ServiceNow IDE commands

Command	Description
Fluent: Apply Template for an existing Fluent App	Adds template files and directories for development in ServiceNow Fluent to an existing application.
Fluent: Build and Install	Build and install the application across the instance. Learn more: Build and install an application in the ServiceNow IDE
Fluent: Build with Now SDK	Build the application into an installable package.
Fluent: Convert a scoped app to Fluent	Convert an existing scoped application to support development in source code. **Note:** With ServiceNow IDE version 3.1.4 and earlier, the sn_glider.ide_fluent_admin role is required to use this command. Learn more: Convert an application with the ServiceNow IDE
Fluent: Create Fluent App	Create a scoped application to develop in source code. Learn more: Create an application with the ServiceNow IDE
Fluent: Download script dependencies	Download script dependencies in your application to get type-ahead support for Glide APIs and script includes. Type definitions are downloaded into the `@types/servicenow` directory based on the scripts in your application. After downloading script dependencies, you can create a `tsconfig.json` file in the `src/fluent` directory that includes the type definitions. For example:

ServiceNow IDE commands (continued)

Command	Description
	<pre>"include": ["./**/*.client.js", "./**/*.server.js", "../../@types/servicenow/*.client.d.ts", "../../@types/servicenow/*.server.d.ts",]</pre> Learn more: Downloading dependencies with the ServiceNow SDK
Fluent: Force Install Fluent App in Instance	Install an existing build artifact, which might overwrite recent changes.
Fluent: Force Sync of Fluent App with all metadata	Synchronize all metadata in the application rather than only the metadata that changed since the last time that the application was synchronized. **Warning:** Force synchronizing all metadata in the application might overwrite any changes in the application that haven't been built and installed on the instance. Learn more: Synchronizing applications in the ServiceNow IDE
Fluent: Install Fluent App in Instance	Install the application across the instance without building it again.
Fluent: Reinstall Fluent App	Uninstall and reinstall the application on the instance to verify that the metadata on the instance matches the metadata in the installation package. The source files in the application aren't changed. **Warning:** Changes to metadata on the instance that haven't been synced into your application in the ServiceNow IDE are removed. If you have previous versions of modules in the EcmaScript Module [sys_module] table that aren't needed, re-installing an application removes previous versions of the application's modules from the table. **Note:** With ServiceNow IDE version 3.1.4 and earlier, the sn_glider.ide_fluent_admin role is required to use this command.
Fluent: Restart Fluent Server	Restart the ServiceNow Fluent language server.
Fluent: Sync Fluent App with changed metadata	Synchronize changes made to the application metadata since the last time that the application was synchronized. Learn more: Synchronizing applications in the ServiceNow IDE

ServiceNow IDE commands (continued)

Command	Description
Git: Manage Git credentials	Manage existing Git credentials for a Git domain and your user on the instance.
Git: Set IDE Git credentials	Configure basic or OAuth 2.0 credentials that are associated with your user to authenticate with a Git provider. Learn more: • Connect to a Git provider using basic authentication with the ServiceNow IDE • Connect to a Git provider using OAuth 2.0 with the ServiceNow IDE
Metadata Explorer: Focus on Metadata Explorer View	Navigate to the Metadata Explorer view. Learn more: Create an application file from the Metadata Explorer
Metadata Explorer: Open in New Browser Tab	Open the Metadata Explorer in a new browser tab.
Metadata Explorer: Reload	Refresh the Metadata Explorer view.
Package Manager: Install Dependencies	Install third-party dependencies into the `node_modules` directory. Learn more: Use third-party libraries in applications in the ServiceNow IDE
Workspaces: Add Application to Workspace	Add an existing application to a workspace.
Workspaces: Add to Workspace	Create, open, or clone an application in a workspace.
Workspaces: Browse Workspaces	Browse through and open your ServiceNow IDE workspaces.
Workspaces: Create a Workspace	Create a ServiceNow IDE workspace. Learn more: Create a workspace in the ServiceNow IDE
Workspaces: Delete a Workspace	Delete a ServiceNow IDE workspace.
Workspaces: Rename a Workspace	Rename a ServiceNow IDE workspace.
Workspaces: Welcome	Navigate to the ServiceNow IDE home page, which lists all of your workspaces.

ServiceNow IDE properties

Manage how the ServiceNow IDE functions on an instance using the following system properties.

Note:
To open the System Properties [sys_properties] table, enter `sys_properties.list` in the navigation filter. To add properties to the table, see Add a system property.

Properties for ServiceNow IDE

Property	Description
sn_glider.default_to_bundled_sdk	Specifies whether to use the bundled version or the latest version of the ServiceNow SDK when creating or converting applications in the ServiceNow IDE. If true, the version of the ServiceNow SDK that's bundled with a version of the ServiceNow IDE is used. If false, the latest version of the ServiceNow SDK is used. • Type: true | false • Default value: false • Location: Add the property to the System Property [sys_properties] table
sn_glider.enable_ide	Enables access to the ServiceNow IDE. If false, access to the ServiceNow IDE is turned off across the instance. • Type: true | false • Default value: true • Location: Add the property to the System Property [sys_properties] table
sn_glider.fluent_convert_enabled	Enables converting existing scoped applications that weren't created with the ServiceNow IDE or ServiceNow SDK to support development in source code from the ServiceNow IDE. • Type: true | false • Default value: true • Location: Add the property to the System Property [sys_properties] table • Learn more: Convert an application with the ServiceNow IDE
sn_glider.git.attachment.extension.binary	Defines a custom extension for attachment files with binary content types (for example, `application/octet-stream`) to override the default `gitdata` extension. The custom extension that you specify with this property must also be specified with the *`glide.attachment.extensions`* system property. **Note:** If you reuse an existing extension, confirm that the value you set passes MIME type validations for binary types. • Type: string • Default value: gitdata

Properties for ServiceNow IDE (continued)

Property	Description
	• Location: Add the property to the System Property [sys_properties] table • Learn more: ServiceNow IDE MID Server User [sn_glider.ide_git_user]
sn_glider.git.attachment.extension.text	Defines a custom extension for attachment files with text content types (for example, `text/plain`) to override the default `txt` extension. The custom extension that you specify with this property must also be specified with the *`glide.attachment.extensions`* system property. **Note:** If you reuse an existing extension, confirm that the value you set passes MIME type validations for text types. • Type: string • Default value: txt • Location: Add the property to the System Property [sys_properties] table • Learn more: ServiceNow IDE MID Server User [sn_glider.ide_git_user]
sn_glider.pinnedSdkVersion	Specifies the version of the ServiceNow SDK to use when creating or converting applications in the ServiceNow IDE. • Type: string • Default value: The latest version of the ServiceNow SDK • Location: Add the property to the System Property [sys_properties] table

ServiceNow IDE roles

The following roles are installed for use with ServiceNow IDE.

Note:
Users with the Creator Studio Restricted User [sn_creatorstudio.restricted_user] role aren't able to access the ServiceNow IDE regardless of any other roles assigned.

ServiceNow IDE Administrator [sn_glider.admin]

Create and develop scoped applications in the ServiceNow IDE.

Contains Roles

List of roles contained within the role.

- sn_glider.create_app_admin
- sn_glider.read_admin
- sn_glider.write_admin

Groups

This role is assigned to no groups by default.

Special considerations

In addition to this role, the following roles are necessary for certain functionality in the ServiceNow IDE:

- credential_admin: Required to perform Git operations.
- sn_udc.admin: Required to modify application files from the Metadata Explorer.
- sn_udc.basic_read: Used for read-only access to application files in the Metadata Explorer.

ServiceNow IDE MID Server User [sn_glider.ide_git_user]

Assign administrator privileges to a MID server user to manage applications in source control from the ServiceNow IDE.

MID Server users must have the sn_glider.ide_git_user role or admin role to perform Git operations in the ServiceNow IDE. For more information, see Create the MID Server user and grant the role and Configure a MID Server to use source control with the ServiceNow IDE.

Contains Roles

List of roles contained within the role:

credential_admin

Groups

This role is assigned to no groups by default.

Special considerations

None.

ServiceNow SDK

Create and develop scoped applications in source code locally in Visual Studio Code Desktop and upload changes to an instance using the ServiceNow software development kit (SDK).

Note:
The ServiceNow SDK doesn't support the Global scope or global applications with instances on the Xanadu release.

Get started

Explore  Learn about developing applications with the ServiceNow SDK.	Configure  Install and configure the ServiceNow SDK locally.	Authenticate  Authenticate to ServiceNow instances.
Create  Create applications in source code.	Develop  Develop applications in source code.	Reference  Get details about the ServiceNow SDK CLI and ServiceNow Fluent APIs.

Which builder should I use to create an app?

Are you a developer who wants to use industry-standard development tools and processes?

The ServiceNow IDE and ServiceNow SDK support developing applications in source code with ServiceNow Fluent, creating JavaScript modules, and using third-party libraries. ServiceNow Fluent is a domain-specific programming language for creating application metadata in code.

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. The ServiceNow SDK uses Visual Studio Code Desktop locally. For more information, see Building applications in source code.

Are you a developer who wants more control in a centralized user interface?

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity. For more information, see Exploring ServiceNow Studio.

Need a more general app but still want low-code options?

App Engine Studio lets you build a broader range of apps than Creator Studio without being a programming pro. For more information, see Exploring App Engine Studio.

Want to build an app easily, without code?

Creator Studio specializes in helping you craft request-fulfillment applications without writing code. For example, an application to request office supplies by filling out a form, and someone approves or denies your request. For more information, see Exploring Creator Studio.

Troubleshoot and get help

- To learn more about what's new in the Xanadu release, see the ServiceNow SDK release notes.
- ServiceNow SDK and Fluent Creator Toolbox video
- ServiceNow SDK examples GitHub repository
- ServiceNow IDE, SDK, and Fluent forum in the ServiceNow Community
- ServiceNow IDE, SDK, and Fluent articles in the ServiceNow Community
- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

Related topics

ServiceNow Fluent

JavaScript modules and third-party libraries

Exploring the ServiceNow SDK

Learn about developing scoped applications in source code locally and installing changes on an instance with the ServiceNow SDK.

ServiceNow SDK overview

With the ServiceNow SDK, you can create and modify scoped applications locally in Visual Studio Code Desktop and install those applications on a non-production instance beginning with the Washington DC release. By developing applications locally, you can work offline and use features that are available in Visual Studio Code but not in the ServiceNow IDE.

In Visual Studio Code, you can develop scoped applications in source code using ServiceNow Fluent to define application metadata [sys_metadata], write custom JavaScript modules, or add third-party libraries.

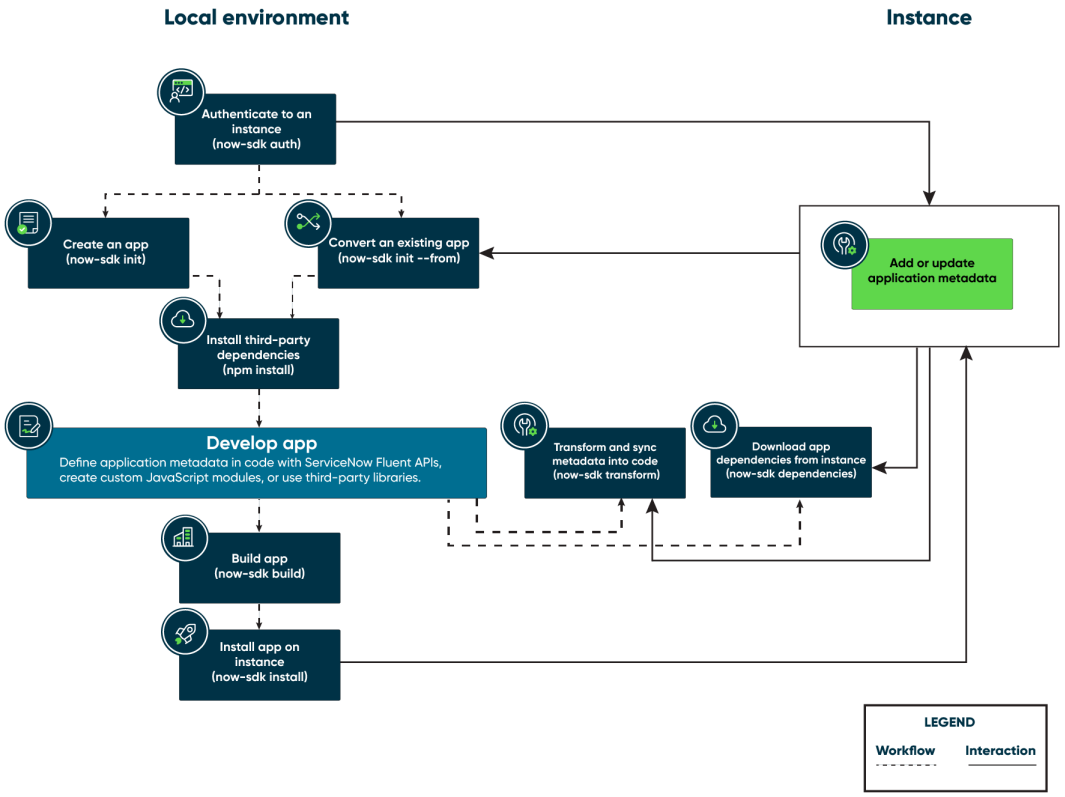
The ServiceNow SDK includes a command-line interface (CLI) for managing changes between an instance and a local application. Using simple CLI commands, you can authenticate to an instance, create or convert an application, transform application metadata (XML) into ServiceNow Fluent code, build and install the application on the instance, and more.

The ServiceNow SDK also serves as the application packaging service that builds applications in the ServiceNow IDE. For more information, see Exploring the ServiceNow IDE.

ServiceNow SDK workflow

The following infographic shows the workflow for developers to get started developing applications with the ServiceNow SDK.

Developing applications with the ServiceNow SDK



1. A developer installs the ServiceNow SDK locally. For more information, see Install the ServiceNow SDK in an application.
2. Using the CLI, the developer authenticates to a non-production instance to interface between their local environment and the instance with the `now-sdk auth` command. The developer must have the admin role on the instance.
3. The developer creates a scoped application (`now-sdk init`) or converts an existing scoped application from the instance (`now-sdk init --from`) for use with the ServiceNow SDK.
4. In Visual Studio Code, the developer can define application metadata in source code using ServiceNow Fluent, create custom JavaScript modules, or use third-party libraries.

   They can also download application and script dependencies from the instance to code against (`now-sdk dependencies`).
5. The developer builds the application, which compiles the source code and transforms it into application metadata for the instance (`now-sdk build`).
6. The developer installs the application on the instance (`now-sdk install`).
7. Other users can continue to modify the application metadata on the instance, and developers can reuse the code from modules in other modules or scripts within the application.
8. The developer downloads and transforms application metadata from the instance to get the latest updates to the application metadata locally (`now-sdk transform`), keeping it synchronized with the application on the instance.

You can manage applications in source control with your preferred Git provider. To develop a scoped application on another system with the ServiceNow SDK, other developers should clone

the application from a remote GIt repository rather than downloading and transforming it from the instance.

ServiceNow SDK benefits

Benefit	Feature	Users
Develop applications offline, outside of an instance in Visual Studio Code and manage them in source control with your preferred Git provider.	Developing applications with the ServiceNow SDK Visual Studio Code documentation	Developers
Write source code to define the metadata that makes up ServiceNow applications	ServiceNow Fluent	Developers
Organize and reuse code within scoped applications with custom JavaScript modules and third-party JavaScript utilities	JavaScript modules and third-party libraries	Developers

Configuring the ServiceNow SDK

Install or upgrade the ServiceNow SDK locally and get language processing and validation for ServiceNow Fluent in Visual Studio Code.

Install the ServiceNow SDK in an application

Install the ServiceNow software development kit (SDK) in a local application using Node Package Manager (npm).

Before you begin

Your system must meet the following minimum requirements to use the latest version of the ServiceNow SDK:

- Node.js v20.18.0
- npm v8.19.3

To install or upgrade Node.js or npm, see the installation instructions on the npm website.

Role required: admin

About this task

The ServiceNow SDK is available as an npm package from the public npm registry. The ServiceNow SDK supports integrating with ServiceNow instances beginning with the Washington DC release.

In the following procedure, you install the ServiceNow SDK within a new application using `npx` rather than globally. To install the ServiceNow SDK globally, use the `npm install --global @servicenow/sdk` command.

Note:
If you use `npx` to install the ServiceNow SDK in application rather than globally, you must use `npx @servicenow/sdk [command]` rather than `now-sdk [command]` with the ServiceNow SDK CLI.

Procedure

1. Open a command-line tool on your system.
2. Verify that your system meets the requirements.
   a. Enter `node -v` to check if you have Node.js installed and which version.
   b. Enter `npm -v` to check if you have npm installed and which version.
3. Create a local directory for your application.
4. Change directories into the directory for your application using the `cd` command.

   Example

   ```
   cd <path/to/directory>
   ```

5. Create an application with the ServiceNow SDK.

What to do next

Install the ServiceNow Fluent Language server to get language processing and validation for ServiceNow Fluent in Visual Studio Code. For more information, see Install the ServiceNow Fluent Language server in Visual Studio Code.

Upgrade the ServiceNow SDK

Upgrade to the latest version of the ServiceNow SDK for use with instances beginning with the Washington DC release.

Before you begin

Your system must meet the following minimum requirements to use the latest version of the ServiceNow SDK:

- Node.js v20.18.0
- npm v8.19.3

To install or upgrade Node.js or npm, see the installation instructions on the npm website.

Role required: admin

Procedure

Complete the following steps depending on whether you're updating a global installation of the ServiceNow SDK or the version used by an application.

What to do next

Install the ServiceNow Fluent Language server to get language processing and validation for ServiceNow Fluent in Visual Studio Code. For more information, see Install the ServiceNow Fluent Language server in Visual Studio Code.

Use the ServiceNow SDK to authenticate to a ServiceNow instance. For more information, see Authenticating to a ServiceNow instance with the ServiceNow SDK.

Install the ServiceNow Fluent Language server in Visual Studio Code

Get language processing and validation for ServiceNow Fluent in Visual Studio Code with the ServiceNow Fluent Language server.

Before you begin
Role required: admin

About this task
The ServiceNow Fluent Language server provides code completion features and validation for ServiceNow Fluent language files (`.now.ts`) and for ServiceNow SDK configuration files (`now.config.json`).

Procedure
1. In Visual Studio Code, select the Extensions view from the Activity Bar.
2. Search for `ServiceNow® Fluent Language`.
3. Select **Install**.

What to do next
To use ServiceNow Fluent in applications with the ServiceNow SDK, see Define application metadata in code with ServiceNow Fluent and the ServiceNow SDK.

Related topics
ServiceNow Fluent

Authenticating to a ServiceNow instance with the ServiceNow SDK
Authenticate to a ServiceNow instance and store user credentials for accessing the instance on your system with the ServiceNow SDK.

You can use basic or OAuth 2.0 credentials to authenticate to an instance. To use OAuth 2.0 authentication, your instance must have ServiceNow IDE (version 1.1 or later) installed or have the required XML configuration imported. By default, instances on the Xanadu release include ServiceNow IDE version 1.1.4 and support using OAuth 2.0 authentication with the ServiceNow SDK.

After authenticating to an instance with the ServiceNow SDK command-line interface (CLI), you can begin developing applications and installing them on your instance. A non-production instance should be used for application development.

Authenticate to a ServiceNow instance using basic authentication with the ServiceNow SDK
Use basic authentication to connect to a ServiceNow instance with the ServiceNow SDK.

Before you begin
Role required: admin

About this task
This procedure uses the ServiceNow SDK command-line interface (CLI). From a command-line tool, enter `now-sdk --help` to get information about the available commands and global options. To get additional information about a command and its parameters, enter the command and `--help` or `-h`. For example, `now-sdk auth --help`. For more information about the CLI, see ServiceNow SDK CLI.

Procedure
1. Create a local directory for your application.
2. In Visual Studio Code, open the directory.
3. From the application directory, open an integrated Terminal window.
4. Specify the instance to authenticate to with the `auth` command.

```
npx @servicenow/sdk auth --add <instance>
```

Example
For example:

```
npx @servicenow/sdk auth --add
 https://myinstance.service-now.com
```

Note:
Using the `npx` command installs the ServiceNow SDK in your application directory instead of globally.

5. Respond to the following series of prompts.

Prompt	Response
`Type of authentication to use`	Select `basic`.
`Alias for these credentials`	Enter an alias for your credentials and the instance. The alias can be used for authentication with the `init`, `transform`, `dependencies`, and `install` commands.
`The username to authenticate with the instance`	Enter your username for the instance. You must be assigned the admin role.
`The password to authenticate with the instance`	Enter your password.

Result
The alias and credentials are stored in the device keychain or credential manager on your system and are set as the default credentials.

What to do next
Add an application to your local system with the ServiceNow SDK. For more information, see Adding applications with the ServiceNow SDK.

Authenticate to a ServiceNow instance using OAuth 2.0 with the ServiceNow SDK

Use OAuth 2.0 authentication to connect to a ServiceNow instance with the ServiceNow SDK.

Before you begin
To use OAuth 2.0 authentication, your instance must have ServiceNow IDE (version 1.1 or later) installed or have the required XML configuration imported. By default, instances on the Xanadu release include ServiceNow IDE version 1.1.4 and support using OAuth 2.0 authentication with the ServiceNow SDK.

Role required: admin

About this task

This procedure uses the ServiceNow SDK command-line interface (CLI). From a command-line tool, enter `now-sdk --help` to get information about the available commands and global options. To get additional information about a command and its parameters, enter the command and `--help` or `-h`. For example, `now-sdk auth --help`. For more information about the CLI, see ServiceNow SDK CLI.

Procedure

1. Create a local directory for your application.
2. In Visual Studio Code, open the directory.
3. From the application directory, open an integrated Terminal window.
4. Specify the instance to authenticate to with the `auth` command.

```
npx @servicenow/sdk auth --add <instance>
```

Example

For example:

```
npx @servicenow/sdk auth --add
 https://myinstance.service-now.com
```

Note:
Using the `npx` command installs the ServiceNow SDK in your application directory instead of globally.

5. Respond to the following series of prompts.

Prompt	Response
`Type of authentication to use`	Select `oauth`.
`Alias for these credentials`	Enter an alias for your credentials and the instance. The alias can be used for authentication with the `init`, `transform`, `dependencies`, and `install` commands.

The ServiceNow SDK opens a web browser to authenticate with the instance.

6. Navigate to the web page that opens and log in to the instance if you aren't currently logged in.
7. Select **Accept** to allow the ServiceNow SDK to connect to the instance.
The page refreshes and includes an authentication code.
8. Select **Copy** to copy the authentication code provided.
9. In the command line, paste the authentication code.

Result

The alias and credentials are stored in the device keychain or credential manager on your system and are set as the default credentials.

What to do next

Add an application to your local system with the ServiceNow SDK. For more information, see Adding applications with the ServiceNow SDK.

Adding applications with the ServiceNow SDK

Add applications to your local system to get started with development in source code with the ServiceNow SDK.

- Create an application with the ServiceNow SDK

  Create a scoped application that supports development in source code.

- Convert an application with the ServiceNow SDK

  Convert an existing scoped application to support development in source code.

You can also clone an existing scoped application that has already been converted to support development in source code from a remote Git repository in Visual Studio Code Desktop. For more information, see Introduction to Git in VS Code in the Visual Studio Code documentation.

Create an application with the ServiceNow SDK

Create a scoped application to develop in source code with the ServiceNow SDK.

Before you begin

Use the ServiceNow SDK to authenticate to a ServiceNow instance. For more information, see Authenticating to a ServiceNow instance with the ServiceNow SDK.

Role required: admin

About this task

This procedure uses the ServiceNow SDK command-line interface (CLI). From a command-line tool, enter `now-sdk --help` to get information about the available commands and global options. To get additional information about a command and its parameters, enter the command and `--help` or `-h`. For example, `now-sdk auth --help`. For more information about the CLI, see ServiceNow SDK CLI.

Procedure

1. Create a local directory for your application.
2. In Visual Studio Code, open the directory.
3. From the application directory, open an integrated Terminal window.
4. Create an application following a guided set of prompts with the `init` command:

   ```
   npx @servicenow/sdk init
   ```

   Note:
   Using the `npx` command installs the ServiceNow SDK in your application directory instead of globally.

5. Respond to the following series of prompts.

Prompt	Response
`Select a template`	Select a template that determines the default application structure, such as whether to create a full-stack application that supports UI development ands whether to use JavaScript or TypeScript to create modules.

Prompt	Response
`Name of ServiceNow Application`	Enter a name for the application.
`NPM package name`	Enter a name for the application package used in the `package.json` file. The package name must adhere to Node Package Manager (npm) package naming standards.
`Create a Global/Scoped App?`	Select whether to create a scoped or global application. ○ Scoped: Create a scoped application that is protected by identifying and restricting access to application files and data. ○ Global: Create an application in the global scope to allow it to be accessible to other global applications. Global applications are supported only with instances beginning with the Australia release.
`Scope name`	For scoped applications, enter the scope of the application. The scope name must be unique on the instance, begin with x_<prefix>, and be 18 characters or fewer. For more information, see Namespace identifier.

Example
In the following example, a scoped application named Example App (x_snc_example_app) is created.

```
$ npx @servicenow/sdk init
[now-sdk] Bootstrapping a new ServiceNow application project...
? Select a template: now-sdk + basic
? Name of ServiceNow Application:  Example App
? NPM package name:  example-app
? Scope name:  x_snc_example_app
[now-sdk] Application created successfully.
          Install the required dependencies with your preferred
 package manager before running "$now-sdk build".
          Ex: Run "npm install" if using npm.
```

6. Install the required third-party dependencies using your preferred package manager.

Example
For example, if you use Node Package Manager (npm), run `npm install`.

7. Build the application with the `build` command:

```
now-sdk build
```

8. **Optional:** Install the application on an instance with the `install` command:

```
now-sdk install --auth <alias>
```

Result
A scoped application with the default application structure is available locally. For information about the application structure, see the Application structure section of the Building applications in source code topic.

If you installed the application successfully, it's available on the instance. For more information about installing applications, see Build and install an application with the ServiceNow SDK.

What to do next

In Visual Studio Code, start developing your application in source code with ServiceNow Fluent, writing custom JavaScript modules, or adding third-party libraries. For more information, see Developing applications with the ServiceNow SDK.

Convert an application with the ServiceNow SDK

Convert an existing scoped application to support development in source code with the ServiceNow SDK.

Before you begin

Use the ServiceNow SDK to authenticate to a ServiceNow instance. For more information, see Authenticating to a ServiceNow instance with the ServiceNow SDK.

Role required: admin

About this task

Existing scoped applications that weren't created with the ServiceNow IDE or ServiceNow SDK must be converted to support development in source code. Converting an application adds the necessary files and directories for developing it in source code. You can choose whether to convert existing application metadata into ServiceNow Fluent code. The application isn't changed on the instance until you build and install it on the instance.

This procedure uses the ServiceNow SDK command-line interface (CLI). From a command-line tool, enter `now-sdk --help` to get information about the available commands and global options. To get additional information about a command and its parameters, enter the command and `--help` or `-h`. For example, `now-sdk auth --help`. For more information about the CLI, see ServiceNow SDK CLI.

Procedure

1. Create a local directory for your application.
2. In Visual Studio Code, open the directory.
3. From the application directory, open an integrated Terminal window.
4. Convert an existing scoped application from an instance or a local directory using the `init` command.

   With the `--from` parameter, provide a sys_id of an application on the instance or a path to a local directory that contains an application to convert to support development in source code.

   ```
   npx @servicenow/sdk init --from <sys_id or path>
   ```

   Example

   For example:

   ```
   npx @servicenow/sdk init --from
    aadfdd904748a6500ff14ee4316d4369
   ```

   Note:
   Using the `npx` command installs the ServiceNow SDK in your application directory instead of globally.

5. At the prompt, enter a name for the application package used in the `package.json` file.

The package name must adhere to Node Package Manager (npm) package naming standards.

Example
In the following example, an application is downloaded from the instance and converted using its sys_id.

```
$ npx @servicenow/sdk init --from
 aadfdd904748a6500ff14ee4316d4369
[now-sdk] Bootstrapping a new ServiceNow application project...
[now-sdk] Please provide the missing required fields:
? NPM package name:  example-app
[now-sdk] Access Token has expired, refreshing token
[now-sdk] Downloading application package for
 aadfdd904748a6500ff14ee4316d4369 from
 https://myinstance.service-now.com
[now-sdk] Unzipping downloaded package...
[now-sdk] Successfully downloaded and unzipped application
 aadfdd904748a6500ff14ee4316d4369
[now-sdk] Summary of downloaded metadata files
Metadata: 13 (Total: 13)
  Copied To Metadata Directory::
    1. dictionary/x_snc_example_app_mytable.xml
    2. package_inventory.csv
    3. sys_app_aadfdd904748a6500ff14ee4316d4369.xml
    4. update/sys_db_object_7030a5d04748a6500ff14ee4316d4338.xml
    5. update/sys_dictionary_x_snc_example_app_mytable_null.xml
    6.
 update/sys_documentation_x_snc_example_app_mytable__en.xml
... and 7 more

[now-sdk] Successfully converted application.
[now-sdk] Install the required dependencies with your preferred
 package manager before running "$now-sdk build".
          Ex: Run "npm install" if using npm.
```

The application is available locally with the default application structure and supports development in source code. Existing application metadata is downloaded into in the `metadata` directory. For information about the application structure, see the Application structure section of the Building applications in source code topic.

6. Install the required third-party dependencies using your preferred package manager.

   Example
   For example, if you use Node Package Manager (npm), run `npm install`.

7. **Optional:** Transform existing application metadata (XML) into ServiceNow Fluent code using the `transform` command.

   (Optional) With the `--from` parameter, provide a path to a local directory or file that contains metadata XML to transform into ServiceNow Fluent code.

   ```
   now-sdk transform [--from <path>] [--directory <package path>]
    [--preview <flag>] [--auth <alias>]
   ```

   Tip:
   You can set the `--preview` parameter to true to preview the transformed ServiceNow Fluent code from the command line without saving the changes.

Example
For example:

```
now-sdk transform --from metadata/update --auth devuser1
```

Application metadata is defined in ServiceNow Fluent code in the `src/fluent/generated` directory and removed from the specified directory.

> **Note:**
> A limited number of metadata types, such as Metadata Snapshots [sys_metadata_link] and UX Assets [sys_ux_lib_asset], can't be represented as ServiceNow Fluent code and aren't transformed. These metadata types remain as metadata XML files in the `metadata` directory of your application.

8. Build the application with the `build` command.

```
now-sdk build
```

9. **Optional:** Update the application on the instance with the `install` command.

```
now-sdk install --auth <alias>
```

Result
The converted application is added to your local directory with the necessary files and directories to support development in source code. If you installed the application successfully, it's updated on the instance. For more information about installing applications, see Build and install an application with the ServiceNow SDK. After installing a converted application, the **Package JSON** field of the custom application record [sys_app] contains the path to the `package.json` file for the application.

New application metadata added after converting an application is automatically transformed into source code in the `src/fluent/generated` directory when you use the `transform` command. If metadata exists in the local application as both XML and source code, the XML version takes precedence when installed on the instance.

What to do next
In Visual Studio Code, start developing your application in source code with ServiceNow Fluent, writing custom JavaScript modules, or adding third-party libraries. For more information, see Developing applications with the ServiceNow SDK.

Developing applications with the ServiceNow SDK

Create and develop applications in source code locally in Visual Studio Code Desktop with the ServiceNow SDK. Build and install applications to make your changes available on an instance.

After installing the ServiceNow SDK, you can authenticate to an instance to create scoped applications or convert existing applications using the ServiceNow SDK command-line interface (CLI).

In Visual Studio Code, start developing your application in source code with ServiceNow Fluent, writing custom JavaScript modules, or adding third-party libraries.

When your changes are ready, build and install the local application on the instance with the ServiceNow SDK CLI. Download and transform changes to the application metadata from the instance to keep the application up to date locally before making additional changes.

Learn how to get started developing applications with the ServiceNow SDK in the following topics.

Related topics

Using TypeScript in JavaScript modules with the ServiceNow SDK

ServiceNow SDK CLI

ServiceNow Fluent API reference

Define application metadata in code with ServiceNow Fluent and the ServiceNow SDK

Define application metadata in code with ServiceNow Fluent and the ServiceNow SDK.

Before you begin

- Install the ServiceNow Fluent Language server to get language processing and validation for ServiceNow Fluent in Visual Studio Code. For more information, see Install the ServiceNow Fluent Language server in Visual Studio Code.
- Add an application to your local system with the ServiceNow SDK. For more information, see Adding applications with the ServiceNow SDK.

Role required: admin

About this task

ServiceNow Fluent is a domain-specific language (DSL) based on TypeScript for defining the metadata files [sys_metadata] that make up applications and includes APIs for the different types of metadata, such as tables, roles, ACLs, business rules, and Automated Test Framework tests. For more information about ServiceNow Fluent APIs and examples, see ServiceNow Fluent API reference.

Procedure

1. In Visual Studio Code, open your scoped application directory.
2. In the `src/fluent` directory, add a file with the `.now.ts` extension.
   You can define application metadata in a single file or as many `.now.ts` files as you want and organize files in directories within the `fluent` directory.

   Tip:
   In the `fluent` directory, you can refer to an example file named `index.now.ts`.

3. In the `.now.ts` file, use objects in the ServiceNow Fluent APIs to define metadata in the application.

   Example

   For example, to create a table [sys_db_object] in the application, use the Table API. The following example includes a simple table for to-do lists using a *`Table`* object with the necessary properties and values.

```
export const x_snc_example_to_do = Table({
    name: 'x_snc_example_to_do',
    schema: {
        deadline: DateColumn({ label: 'Deadline' }),
        state: StringColumn({
            label: 'State',
            choices: {
                ready: { label: 'Ready' },
                completed: { label: 'Completed' },
                inProgress: { label: 'In Progress' },
            },
        }),
```

```
        task: StringColumn({ label: 'Task', maxLength: 120 }),
    },
})
```

Note:
You can use the Record API to define application metadata that doesn't have a dedicated API.

4. Add the required imports for the APIs from `@servicenow/sdk/core`.

   Example

   For example, to define a table with a date column and string column, import the corresponding objects to use from the Table API:

   ```
   import { Table, DateColumn, StringColumn }
    from '@servicenow/sdk/core'
   ```

5. **Optional:** For ServiceNow Fluent APIs that use server-side scripts, such as business rules, import code from JavaScript modules to call them from an object's *`script`* property.

   Example
   This example imports the `showStateUpdate` function, which can then be referenced from the *`script`* property.

   ```
   import { showStateUpdate } from '../server/script.js'
   ```

6. Save your changes.
7. **Optional:** Build and install your application to compile source code into application metadata and make your changes available on the instance. For more information, see Build and install an application with the ServiceNow SDK.

Trouble?

Tip:
You can use the following directives in a code comment to help manage your code:

- `@fluent-ignore`: Suppresses ServiceNow Fluent diagnostic warnings and errors in the following line of code.
- `@fluent-disable-sync`: Turns off syncing changes to a ServiceNow Fluent object. Use before a call expression (for example, `Record({ ... })`) to turn off syncing for that object and its child objects. Only use this directive if you want to ignore changes made outside of the source code to the object and never update it when syncing.
- `@fluent-disable-sync-for-file`: Turns off syncing changes to a ServiceNow Fluent file (`.now.ts`). Use in the first line of the file to turn off syncing for all code in the file. Only use this directive if you want to ignore changes made outside of the source code to the file and never update it when syncing.

Example: Defining application metadata in source code with ServiceNow Fluent

In files with the `.now.ts` extension, use objects in the ServiceNow Fluent APIs to define metadata in the application. You must also include the required imports for the APIs from `@servicenow/sdk/core`. For objects with server-side scripts, such as the *`BusinessRule`* object, you can import and use code from JavaScript modules.

Tip:
You can use content from another file in ServiceNow Fluent APIs by referring to the file from a property using the syntax `Now.include('path/to/file')`. Referring to content in a different file instead of including the content inline can be helpful for records with fields that support JavaScript, HTML, CSS, and so on to make developing different file types easier with the appropriate syntax highlighting. The `Now.include` utility supports two-way synchronization so changes to fields from other ServiceNow AI Platform user interfaces are synced into the referenced file's source code and changes to the code are synced back to metadata across the instance.

The following example includes the definitions of a table, client script, and business rule in the application. The client script uses a script from the `client-script.js` file. The business rule uses a function from the `script.js` JavaScript module.

```
import '@servicenow/sdk/global'
import { BusinessRule, ClientScript, DateColumn, StringColumn,
 Table } from '@servicenow/sdk/core'
import { showStateUpdate } from '../server/script.js'

//creates todo table, with three columns (deadline, status and task)
export const x_snc_example_to_do = Table({
    name: 'x_snc_example_to_do',
    schema: {
        deadline: DateColumn({ label: 'Deadline' }),
        state: StringColumn({
            label: 'State',
            choices: {
                ready: { label: 'Ready' },
                completed: { label: 'Completed' },
                inProgress: { label: 'In Progress' },
            },
        }),
        task: StringColumn({ label: 'Task', maxLength: 120 }),
    },
})

//creates a client script that pops up 'Table loaded successfully!!' message everytime todo
 record is loaded
ClientScript({
    $id: Now.ID['cs0'],
    name: 'my_client_script',
    table: 'x_snc_example_to_do',
    active: true,
    appliesExtended: false,
    global: true,
    uiType: 'all',
    description: 'Custom client script generated by Now SDK',
    isolateScript: false,
    type: 'onLoad',
    script: Now.include('../client/client-script.js'),
})

//creates a business rule that pops up state change message whenever a todo record is
 updated
BusinessRule({
    $id: Now.ID['br0'],
    action: ['update'],
```

```
    table: 'x_snc_example_to_do',
    script: showStateUpdate,
    name: 'LogStateChange',
    order: 100,
    when: 'after',
    active: true,
})
```

The client script referenced from the *ClientScript* object:

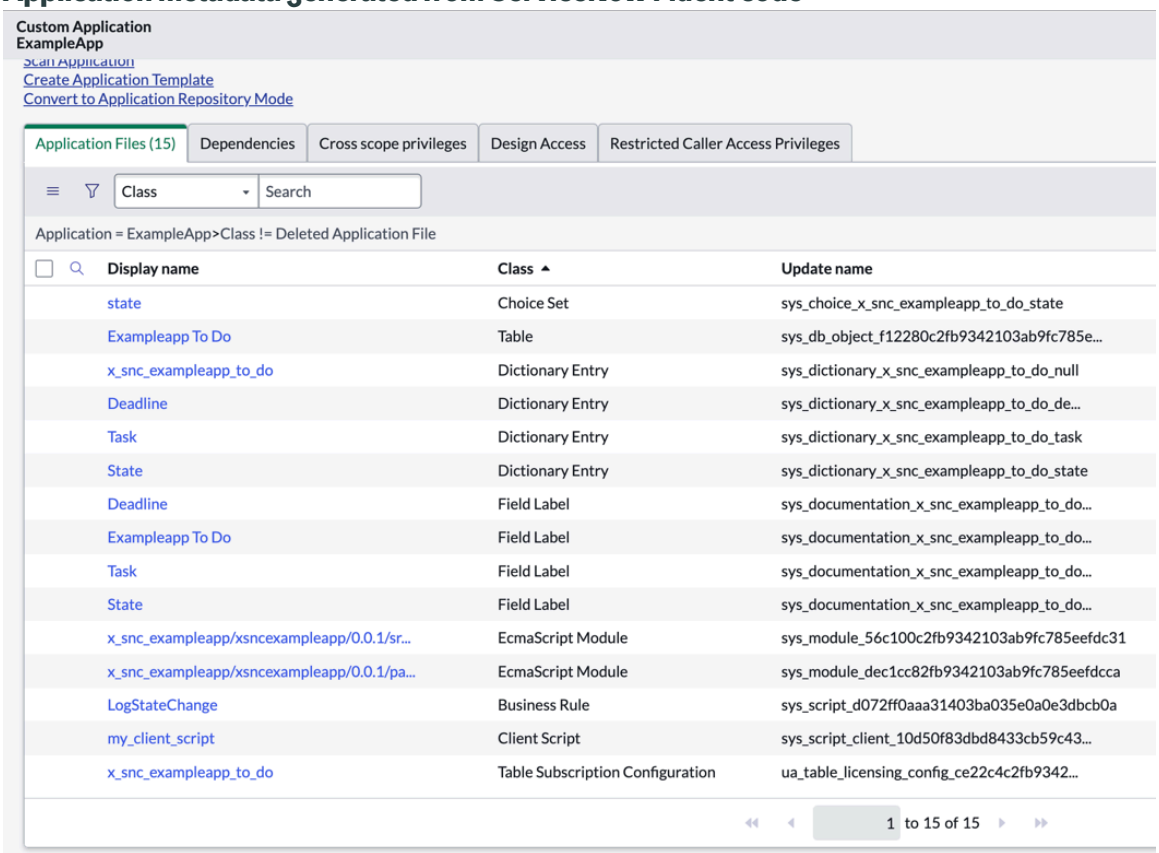
```
function onLoad() {
    g_form.addInfoMessage("Table loaded successfully!!")
}
```

The JavaScript module referenced from the *BusinessRule* object:

```
import { gs } from '@servicenow/glide'

export function showStateUpdate(current, previous) {
    const currentState = current.getValue('state')
    const previousState = previous.getValue('state')

    gs.addInfoMessage(`state updated from "${previousState}"
 to "${currentState}"`)
}
```

After building and installing the application, this source code generates the following application metadata files on the instance.

Application metadata generated from ServiceNow Fluent code



Related topics

ServiceNow Fluent

Create and use JavaScript modules in applications with the ServiceNow SDK

Optimize your codebase by defining reusable code blocks with JavaScript modules and the ServiceNow SDK.

Before you begin

Add an application to your local system with the ServiceNow SDK. For more information, see Adding applications with the ServiceNow SDK.

Role required: admin

About this task

To learn about support for using JavaScript modules in scoped applications, including some limitations, see JavaScript modules and third-party libraries. For general information about the syntax used to create JavaScript modules, see the JavaScript modules page on the MDN Web Docs website.

Procedure

1. In Visual Studio Code, open your scoped application directory.
2. In the `src/server` directory of the application, create a JavaScript or TypeScript file to contain the module code you want to reuse.
3. **Optional:** Import server APIs or script includes to call them from your module.

   (Optional) Glide APIs can be imported from the `@servicenow/glide` package or their namespace in the package. Script includes can be imported from their application scope or the global scope in the `@servicenow/glide` package.

   For example:

   ```
   import { API } from "@servicenow/glide";
   import { API } from "@servicenow/glide/<namespace>";
   import { ScriptInclude } from "@servicenow/glide/<scope>";
   import { global } from "@servicenow/glide/global";
   ```

4. In the module, identify the code to export with *`export`* statements.

   You can use a named export or default export. Named exports can be variables, constants, functions, or classes whereas default exports can be functions or classes only.

   Example

   The following example is one way of adding a named export for multiple features (a function and a variable) in a module:

   ```
   export { myFunction, myVariable };
   ```

5. Use code from the exported module in other modules or server-side scripts.

6. Save your changes.

What to do next

To use third-party libraries in a JavaScript module, see Use third-party libraries in applications with the ServiceNow SDK.

To build your application and add the modules to the EcmaScript Module [sys_module] table, see Build and install an application with the ServiceNow SDK.

Related topics

JavaScript modules and third-party libraries

Using TypeScript in JavaScript modules with the ServiceNow SDK

Use TypeScript when creating JavaScript modules with the ServiceNow SDK.

TypeScript uses static typing and type annotations to support developers catching errors earlier while writing code in Visual Studio Code.

For general information about using TypeScript, see the TypeScript Documentation on the typescriptlang.org website.

Use TypeScript in JavaScript modules

Use TypeScript in JavaScript modules by adding support for TypeScript in your application.

Before you begin

Install TypeScript version 4.8.4 or later. For installation instructions, see Download TypeScript on the typescriptlang.org website.

Role required: admin

About this task

Follow this procedure to update existing applications that weren't created using a TypeScript template to use TypeScript in modules. Beginning with the ServiceNow SDK version 3.0, applications support using TypeScript in JavaScript modules by default using default compiler options. To use a `tsconfig.json` file with custom options for transpiling TypeScript into JavaScript during the build process, configure the `tsconfigPath` parameter in the `now.config.json` file. If you want to use a custom transpilation step before building the application, configure the `modulePaths` parameter in the `now.config.json` file.

Procedure

1. In Visual Studio Code, open your scoped application directory.
2. Open the application's `package.json` file and in the `devDependencies` object, add the TypeScript package and version.

   Example

   ```
   "devDependencies": {
         "typescript": "<version>",
         "@servicenow/sdk": "2.0.0",
         "@servicenow/glide": "26.0.1",
         "eslint": "8.50.0",
         "@servicenow/eslint-plugin-sdk-app-plugin": "2.0.0"
      }
   ```

3. In your application's `src/server` directory, add a `tsconfig.json` file that defines the options for compiling the application.

   Example

   ```
   {
         "compilerOptions": {
               "rootDir": "./",
               "outDir": "../../dist/server",
   ```

```
            "module": "es2022",
            "target": "es2022",
            "moduleResolution": "bundler",
            "allowJs": true,
            "declaration": false,
            "sourceMap": false,
            "skipLibCheck": true,
            "allowImportingTsExtensions": true,
            "noEmit": true
        },
        "include": [
            "./**/*.ts",
    "../../@types/**/*.modules.d.ts"
        ],
        "exclude": [
            "**/*.now.ts"
        ]
}
```

4. In the application's `now.config.json` file, set the `tsconfigPath` parameter to the location of the `tsconfig.json` for JavaScript modules. .

   Example

```
{
   "scope": "x_snc_example_app",
   "scopeId": "2f8400eb07426110f736e28f69d3017a",
   "name": "ExampleApp",
   "tsconfigPath": "./src/server/tsconfig.json"

}
```

5. In the `src/server` directory, add at least one `.ts` file to contain module code.
   For information about creating modules, see Create and use JavaScript modules in applications with the ServiceNow SDK.

What to do next

Build and install the application to compile the TypeScript files into JavaScript modules and add the modules to the EcmaScript Module [sys_module] table. For more information, see Build and install an application with the ServiceNow SDK.

Add type definitions for APIs

Get type-ahead support for APIs and scriptable objects outside of Glide APIs.

Before you begin

Role required: admin

About this task

Note:
You can download type definitions for most APIs, script includes, and other scriptable objects using the `now-sdk dependencies` command. For more information, see Download module and script dependencies.

Follow this procedure to manually add types definitions that the `now-sdk dependencies` doesn't download. You can declare modules directly in the ServiceNow SDK application to stub access to the APIs for type-ahead support. These modules aren't packaged in the application

package, but they can be tracked in a source control repository for the application and shared between developers.

Procedure

1. In Visual Studio Code, open your scoped application directory.
2. In your application, add a TypeScripe (`.ts`) file for type definitions.
3. In your TypeScript file, declare modules for APIs and scriptable objects.

Example

This example declares a module for an API using the API namespace (`sn_app_api`):

```
declare module '@servicenow/glide/sn_app_api' {
 class AppStoreAPI {
  static canUpgradeAnyStoreApp(): boolean
 }
}
```

This example declares a module to access objects defined in script includes using the scope of the script include (`x_1234_scope`):

```
declare module '@servicenow/glide/x_1234_scope' {
 class MyLogItemClass {
  myLogFunction()
 }
}
```

4. In JavaScript modules in your application, import the declared modules.

Example

This example imports the declared module for the AppStoreAPI.

```
import { gs } from '@servicenow/glide'
import { AppStoreAPI } from '@servicenow/glide/sn_app_api'

export const canUpgradeStoreApp = function () {
 var canUpgrade = AppStoreAPI.canUpgradeAnyStoreApp()
 if (canUpgrade) {
  gs.addInfoMessage(`You can upgrade store apps!`)
 } else {
  gs.addInfoMessage(`You cannot upgrade store apps!`)
 }
}
```

This example imports the declared module for the MyLogItemClass object.

```
import { MyLogItemClass } from '@servicenow/glide/x_1234_scope'

export const myLogFunction = function (status) {
 const myLogItem = new MyLogItemClass()
 myLogItem.myLogFunction(status)
}
```

Note:
Modules can access only global scriptable objects or scriptable objects in the same application scope.

Use third-party libraries in applications with the ServiceNow SDK

Call third-party libraries in your application to use existing open-source functionality with the ServiceNow SDK.

Before you begin

Create a JavaScript module. For more information, see Create and use JavaScript modules in applications with the ServiceNow SDK.

Role required: admin

About this task

Third-party libraries are added to applications as JavaScript modules. For general information about the syntax used to create JavaScript modules, see the JavaScript modules page on the MDN Web Docs website.

Important:
You can't use third-party libraries that rely on unsupported functionality, such as unsupported APIs or ECMAScript features. For more information about unsupported functionality, see JavaScript modules and third-party libraries.

Procedure

1. In Visual Studio Code, open your scoped application directory.
2. From the application directory, open an integrated Terminal window.
3. Install the npm package for the library.

   Example

   ```
   npm install <package name>
   ```

   The library's packages are added to your application in the `node_modules` directory, and the library is added as a dependency in the `package.json` file for the application.

   ```
   "dependencies": {
       "<package name>": "<version>"
     }
   ```

4. In a JavaScript module, import the library using an *`import`* statement.

   Example

   In this example, the module includes a namespace import for the lodash module.

   ```
   import * as lodash from "lodash"
   ```

   In this example, the module includes a named import for the camelCase function in the lodash module.

   ```
   import camelCase from 'lodash'
   ```

5. Call code imported from the library in your module to reuse it.

What to do next

Build and install your changes on an instance. For more information, see Build and install an application with the ServiceNow SDK.

Related topics

JavaScript modules and third-party libraries

Downloading dependencies with the ServiceNow SDK

Download application dependencies and TypeScript definitions from an instance for IntelliSense and validation of scripts and ServiceNow Fluent code.

Throughout the development process, you should download dependencies and TypeScript definitions from an instance to support coding against those dependencies in an application.

To download all script and ServiceNow Fluent dependencies for an application, you can use the `dependencies` command with no parameters. If needed, provide the application directory and authentication alias too.

```
now-sdk dependencies [--directory <package path>] [--auth
 <alias>]
```

Download ServiceNow Fluent application dependencies

Download TypeScript definitions for dependencies in other application scopes to get IntelliSense support and validation for ServiceNow Fluent code.

Before you begin

Add an application to your local system with the ServiceNow SDK. For more information, see Adding applications with the ServiceNow SDK.

Role required: admin

About this task

This procedure uses the ServiceNow SDK command-line interface (CLI). From a command-line tool, enter `now-sdk --help` to get information about the available commands and global options. To get additional information about a command and its parameters, enter the command and `--help` or `-h`. For example, `now-sdk auth --help`. For more information about the CLI, see ServiceNow SDK CLI.

Procedure

1. In Visual Studio Code, open your scoped application directory.
2. In the `now.config.json` file, add the items that your application depends on in the `dependencies` object.

   The `dependencies` object has the following structure. You must specify the application scope and the dependency type and names or sys_ids. Only tables and roles can be specified by name. You can use a wildcard (`*`) to add all items from a specified table and scope.

   ```
   "dependencies": {
        "<scope>": {
           "<type>": ["<sys_id or name>"],
           ...
        },
        ...
   }
   ```

   Example

   For example:

   ```
   {
     "dependencies": {
        "global": {
           "tables": ["incident", "problem"],
           "roles": ["admin"],
   ```

```
        "sys_script_client": ["fa776f6d97700100f309124eda2975bc"]
      },
      "x_custom": {
        "tables": ["custom_table"],
        "sys_security_acl": "*",
      }
    }
}
```

3. Save your changes.
4. From the application directory, open an integrated Terminal window.
5. Download dependencies of the application with the `dependencies` command.

Example

```
now-sdk dependencies --auth <alias> --fluent-only
```

Note:
If you want to download script dependencies too, exclude the `--fluent-only` parameter.

The `dependencies` command downloads the dependencies specified in the `now.config.json` file and generates TypeScript definitions for them in the `@types/servicenow/fluent` directory with the `.d.now.ts` file extension. When building the application, these files aren't compiled like source code files.

What to do next

You can reference dependencies in ServiceNow Fluent files using the `#now:{scope}/{category}` subpath import format. For example:

```
// Import roles from global scope
import { role as globalRole } from '#now:global/security'

// Import roles from custom app scope
import { role as xExampleAppRole } from '#now:x_example_app/security'

// Use in ACLs, flows, etc.
Acl({
    $id: Now.ID['my_acl'],
    type: 'record',
    table: 'incident',
    operation: 'read',
    roles: [globalRole.admin, xExampleAppRole.xExampleAppCool],
})
```

Your application's `package.json` file must include the following `imports` configuration to do so:

```
{
  "imports": {
    "#now:*": "./@types/servicenow/fluent/*/index.js"
  }
}
```

Download module and script dependencies

Download TypeScript definitions for module and script dependencies to get IntelliSense support and validation for Glide APIs and script includes.

Before you begin

Add an application to your local system with the ServiceNow SDK. For more information, see Adding applications with the ServiceNow SDK.

Role required: admin

About this task

This procedure uses the ServiceNow SDK command-line interface (CLI). From a command-line tool, enter `now-sdk --help` to get information about the available commands and global options. To get additional information about a command and its parameters, enter the command and `--help` or `-h`. For example, `now-sdk auth --help`. For more information about the CLI, see ServiceNow SDK CLI.

Procedure

1. In Visual Studio Code, open your scoped application directory.
2. From the application directory, open an integrated Terminal window.
3. Download dependencies for scripts in the application with the `dependencies` command.

   Example

   ```
   now-sdk dependencies --auth <alias> --type-defs-only
   ```

   Note:
   If you want to download application dependencies too, exclude the `--type-defs-only` parameter.

   The `dependencies` command downloads type definitions for all Glide APIs and scans the modules and scripts in your application and creates type definitions for the script includes that they use. Type definitions are added in the `@types/servicenow` directory. To add any other type definitions, you can create them manually. For more information, see Add type definitions for APIs.

4. In the `src/fluent` directory, create a `tsconfig.server.json` file for server-side scripts.
   In the `include` object, include the type definitions that you downloaded for server-side Glide APIs (`glide.server.d.ts`) and script includes (`script-includes.server.d.ts`).

   Example

   ```
   {
     "compilerOptions": {
       "lib": [
         "ES2021"
       ],
       "noEmit": true,
       "checkJs": false,
       "allowJs": true,
       "noEmitHelpers": true,
       "esModuleInterop": false,
       "module": "None",
       "types": []
     },
     "include": [
       "./**/*.server.js",
       "../../@types/servicenow/*.server.d.ts",
   ```

```
    ]
}
```

5. In the `src/fluent` directory, create a `tsconfig.client.json` file for client-side scripts. In the `include` object, include the type definitions that you downloaded for client-side Glide APIs (`glide.client.d.ts`).

Example

```
{
  "compilerOptions": {
    "target": "ES6",
    "lib": [
      "DOM",
      "ES6"
    ],
    "checkJs": false,
    "allowJs": true,
    "noEmit": true,
    "noEmitHelpers": true,
    "esModuleInterop": false,
    "module": "None",
    "types": []
  },
  "include": [
    "./**/*.client.js",
    "../../@types/servicenow/*.client.d.ts",
  ]
}
```

6. In the `src/fluent` directory, create a `tsconfig.json` file.

> **Note:**
> This `tsconfig.json` file is separate from a `tsconfig.json` file used for applications that use TypeScript to create JavaScript modules.

In the `references` object, add the paths to the `tsconfig.server.json` and `tsconfig.client.json` files.

Example

```
{
  "files": [],
  "references": [
    {
      "path": "./tsconfig.server.json"
    },
    {
      "path": "./tsconfig.client.json"
    }
  ]
}
```

7. Save your changes.

Result

With this configuration, you can use the `.server.js` file extension for server-side scripts and `.client.js` file extension for client-side scripts and get type-ahead support during development.

Build and install an application with the ServiceNow SDK

Build and install a local application developed with the ServiceNow SDK on a ServiceNow instance.

Before you begin

Add an application to your local system with the ServiceNow SDK. For more information, see Adding applications with the ServiceNow SDK.

Role required: admin

About this task

This procedure uses the ServiceNow SDK command-line interface (CLI). From a command-line tool, enter `now-sdk --help` to get information about the available commands and global options. To get additional information about a command and its parameters, enter the command and `--help` or `-h`. For example, `now-sdk auth --help`. For more information about the CLI, see ServiceNow SDK CLI.

Note:
To build global applications with the ServiceNow SDK, you should use Node Package Manager (npm) as your package manager, especially if your application has more complex dependencies.

Procedure

1. In Visual Studio Code, open your scoped application directory.
2. From the application directory, open an integrated Terminal window.
3. Download changes to the application from the instance with the `transform` command to stay up to date with changes locally.

   Example

   ```
   now-sdk transform --auth <alias>
   ```

   Changes to application metadata are downloaded and synchronized into source code. If metadata exists in the local application as both XML and source code, the XML version takes precedence when installed on the instance.

   Note:
   Updates to JavaScript modules aren't included when downloading application metadata from your instance.

4. Build your application using the `build` command.

   ```
   now-sdk build
   ```

   The build artifacts are output to the `dist/app` directory, including the metadata as XML files in the `dist/app/update` directory.
5. Install your application using the `install` command.

```
now-sdk install --auth <alias>
```

Warning:
If you set the `--reinstall` parameter to true with the `install` command, application metadata created by other developers on your instance can be removed. Any application metadata on your instance that is not present in the local installment package generated during the build process is removed during installment. Before building your application, use the `now-sdk transform` command to synchronize your local metadata with the metadata on your instance.

Result
The application is updated on your instance to reflect your local changes. ServiceNow Fluent code is compiled into application metadata, and modules in the application are added to the EcmaScript Module [sys_module] table on the instance.

Note:
If you have previous versions of modules in the EcmaScript Module [sys_module] table that aren't needed, re-installing an application removes previous versions of the application's modules from the table.

ServiceNow SDK reference

Reference topics provide additional information about ServiceNow SDK CLI commands and ServiceNow Fluent APIs.

ServiceNow SDK CLI

Use the ServiceNow SDK command-line interface (CLI) to manage changes between a local application and the application on an instance.

From the command-line tool on your system, enter `now-sdk` to start the CLI and return a list of available commands or `now-sdk [command]` to begin using the ServiceNow SDK.

Note:
If you use `npx` to install the ServiceNow SDK in application rather than globally, you must use `npx @servicenow/sdk [command]` rather than `now-sdk [command]` with the ServiceNow SDK CLI.

The CLI includes the following commands and global options:

Global options

Option	Description
--version, -v	Return the version of the CLI.
--help, -h	Return information about commands, sub-commands, and parameters.
--debug, -d	Return the debug logs generated with a command.

auth

Authenticate to an instance and store, update, or view user credentials for accessing an instance on your system.

The `auth` command has parameters for adding credentials, deleting credentials, listing credentials, and setting credentials to use by default.

For more information, see Authenticating to a ServiceNow instance with the ServiceNow SDK.

add

Store credentials in the device keychain or credential manager on your system.

The `auth` command has the following structure with the `--add` parameter:

```
npx @servicenow/sdk auth [--add <instance url>] [--type <auth
 method>] [--alias <alias>]
```

Note:
Using the `npx` command installs the ServiceNow SDK in your application directory instead of globally.

Required parameters

Parameter	Type	Description	Default value
--add	String	The URL of the target instance to access and to which you install applications. The instance must be on the Washington DC release or later.	—

Optional parameters

Parameter	Type	Description	Default value
--type	String	The method to use to authenticate with the target instance. Specify `basic` for basic authentication or `oauth` for OAuth 2.0 authentication.	basic
--alias	String	The alias for the instance and user credentials. The alias can be used for authentication with the `init`, `transform`, `dependencies`, and `install` commands.	—

For example:

```
npx @servicenow/sdk auth --add
 https://myinstance.service-now.com --type oauth --alias devuser1
```

delete

Remove credentials in the device keychain or credential manager on your system.

The `auth` command has the following structure with the `--delete` parameter:

```
now-sdk auth [--delete <alias or all>]
```

Required parameters

Parameter	Type	Description	Default value
--delete	String	The alias for the instance and user credentials. To delete all credentials, set this parameter to `--delete all`.	—

For example:

```
now-sdk auth --delete devuser1
```

list

View credentials saved in the device keychain or credential manager on your system. Passwords and authentication codes aren't returned.

The `auth` command has the following structure with the `--list` parameter:

```
now-sdk auth [--list]
```

Required parameters

Parameter	Type	Description	Default value
--list	String	Lists all available authentication credentials.	—

For example:

```
now-sdk auth --list
```

use

Set the credentials to be used by commands by default.

The `auth` command has the following structure with the `--use` parameter:

```
now-sdk auth [--use <alias>]
```

Required parameters

Parameter	Type	Description	Default value
--use	String	The alias for the instance and user credentials.	—

For example:

```
now-sdk auth --use devuser1
```

init

Create a custom scoped application or convert an existing scoped application to support development in source code. The application is added in the current directory.

To create an application with your default credentials, you can run the `init` command without any optional parameters. For example, `now-sdk init`. To convert an existing application, you must include the `--from` parameter to provide the sys_id of an application on an instance or a path to a local directory that contains an application.

After initializing an application, you must install the required third-party dependencies using your preferred package manager before building the application. For example, if you use Node Package Manager (npm), run `npm install`.

The `init` command has the following structure:

```
npx @servicenow/sdk init [--from <sys_id or path>] [--appName
 <name>] [--packageName <name>] [--scopeName <name>] [--auth
 <alias>] [--template <template>]
```

Note:
Using the `npx` command installs the ServiceNow SDK in your application directory instead of globally.

Optional parameters

Parameter	Type	Description	Default value
--from	String	A sys_id of an application on the instance or a path to a local directory that contains an application to convert to support development in source code. Converting an application adds the necessary files and directories for using the ServiceNow SDK locally and downloads the application metadata. The application isn't changed on the instance until you build and install it on the instance. After installing a converted application, the **Package JSON** field of the custom application record [sys_app] contains the path to the `package.json` file for the application.	—
--appName	String	A name for the application.	—
--packageName	String	A name for the application package used in the `package.json` file. The package name must adhere to Node Package Manager (npm) package naming standards.	—
--scopeName	String	The scope of the application. The scope name must be unique on the instance, begin with x_<prefix>, and be 18 characters	—

Optional parameters (continued)

Parameter	Type	Description	Default value
		or fewer. For more information, see Namespace identifier.	
--auth, -a	String	An alias for the credentials to use to authenticate to the instance.	If set, the default alias.
--template	String	A template for the default structure of the application. • base: An application with only the basic structure necessary for development in source code. • javascript.basic: An application configured for development in ServiceNow Fluent and JavaScript. • javascript.react: An application configured for development in ServiceNow Fluent, JavaScript, and React. • typescript.basic: An application configured for development in ServiceNow Fluent and TypeScript. TypeScript source files in the `src/server` directory are transpiled into JavaScript modules. • typescript.react: An application configured for development in ServiceNow Fluent, TypeScript, and React. TypeScript source files in the `src/server` directory are transpiled into JavaScript modules. • partial.javascript.react: Adds template files and directories to an existing application for development in ServiceNow Fluent, JavaScript, and React. • partial.typescript.react: Adds template files and directories to an existing application for development in ServiceNow Fluent, TypeScript, and React.	—

For example:

```
npx @servicenow/sdk init --from dbce0f6a3b3fda107b45b5d355e45af6
 --appName Example App --packageName example-app --scopeName
 x_snc_example --auth devuser1 --template base
```

For more information, see Create an application with the ServiceNow SDK or Convert an application with the ServiceNow SDK.

build

Compile source files and output build artifacts. Third-party library dependencies are converted into XML files that can be installed with the application.

The `build` command has the following structure:

```
now-sdk build <source> [--frozenKeys <flag>]
```

Optional parameters

Parameter	Type	Description	Default value
source	String	The path to the directory that contains the `package.json` file for your application. The `package.json` should be in the base directory of your application.	Current working directory
--frozenKeys	Boolean	An option to validate that the `keys.ts` file is up to date for continuous integration (CI) builds. If true and changes were made to the application's ServiceNow Fluent code, the `keys.ts` file isn't updated and the build fails. The `keys.ts` file is automatically generated in the `src/fluent/generated` directory.	false

For example:

```
now-sdk build /path/to/package --frozenKeys true
```

For more information, see Build and install an application with the ServiceNow SDK.

install

Package the build artifacts and install or update an application on an instance. Before using the `install` command, you must use the `build` command to generate an installable package.

The `install` command has the following structure:

```
now-sdk install [--source <package path>] [--reinstall <flag>]
 [--auth <alias>] [--open-browser <flag>] [--info <flag>]
```

Optional parameters

Parameter	Type	Description	Default value
--source	String	The path to the directory that contains the `package.json` file for your application.	Current working directory

Optional parameters (continued)

Parameter	Type	Description	Default value
		The `package.json` should be in the base directory of your application.	
--reinstall, -r	Boolean	An option to uninstall and reinstall the application on the instance to ensure that the metadata on the instance matches the metadata in the installation package. **Warning:** Metadata that is on the instance but not in your local application is removed. If you have previous versions of modules in the EcmaScript Module [sys_module] table that aren't needed, re-installing an application removes previous versions of the application's modules from the table.	false
--auth, -a	String	An alias for the credentials to use to authenticate to the instance. **Note:** For CI/CD pipelines, you can set the following environment variables to authenticate with an instance at runtime using basic authentication: • SN_SDK_INSTANCE_URL: The URL of the target instance to access and to which you install applications. • SN_SDK_USER: A username for the instance. • SN_SDK_USER_PWD: The password associated with the user. • SN_SDK_NODE_ENV: Set to SN_SDK_CI_INSTALL to enable CI server support.	If set, the default alias.
--open-browser, -b	Boolean	An option to open the application record in your default browser	false

Optional parameters (continued)

Parameter	Type	Description	Default value
		after successfully installing the application.	
--info, -i	Boolean	An option to return details about the most recent installation of this application, such as the status and records updated. When this parameter is used, the application isn't installed.	false

For example:

```
now-sdk install --source /path/to/package --reinstall false
 --auth devuser1 --open-browser true --info true
```

For more information, see Build and install an application with the ServiceNow SDK.

dependencies

Download application dependencies and TypeScript definitions from an instance to enable IntelliSense and code validation for an application.

The `dependencies` command downloads dependencies and TypeScript definitions for scripts and ServiceNow Fluent code:

- For scripts, this command downloads TypeScript definitions for all Glide APIs and scans the modules and scripts in your application and generates type definitions for the script includes that they use. Type definitions are added in the `@types/servicenow` directory. After downloading script dependencies, you must update your `tsconfig.json` file to include the type definitions.
- For ServiceNow Fluent, this command downloads the dependencies listed in an application's `now.config.json` file and generates TypeScript definitions for them in the `@types/servicenow/fluent` directory.

To download all script and ServiceNow Fluent dependencies for an application, you can use the `dependencies` command with no parameters. If needed, provide the application directory and authentication alias too.

The `dependencies` command has the following structure:

```
now-sdk dependencies [--directory <package path>] [--auth
 <alias>] [--type-defs-only <flag>] [--fluent-only <flag>] [--add <table> <sys_ids or names>]
[--scope <name>]
```

Optional parameters

Parameter	Type	Description	Default value
--directory	String	The path to the directory that contains the `package.json` file for your application.	Current working directory

Optional parameters (continued)

Parameter	Type	Description	Default value
		The `package.json` should be in the base directory of your application.	
--auth, -a	String	An alias for the credentials to use to authenticate to the instance.	If set, the default alias.
--type-defs-only	Boolean	An option to download TypeScript definitions for only script dependencies. Script types are downloaded in the `@types/servicenow` directory.	false
--fluent-only	Boolean	An option to download dependencies and TypeScript definitions for only ServiceNow Fluent dependencies from other application scopes. You must list an application's dependencies in its `now.config.json` file. ServiceNow Fluent types are generated in the `@types/servicenow/fluent` directory.	false
--add	String	Adds the dependencies that you want to download to the application's `now.config.json` file. You must specify the type of dependencies to add using the table name and then the names or sys_ids of each item to add, separated by spaces. `--add <table_name> <sys_id1> <sys_id2> <sys_id3>` • To add table dependencies, you can use the `table` alias and names of tables instead of the sys_ids. For example: `--add tables incident problem change_request` • To add role dependencies, you can use the `role` alias and	—

Optional parameters (continued)

Parameter	Type	Description	Default value
		names of roles instead of the sys_ids. For example: `--add roles admin user itil` Use a wildcard (`*`) to add all items from a specified table and scope. For example: `--add sys_security_acl "*"`	
sysIds	Array	A list of sys_ids of dependencies to download and generate TypeScript definitions from. This parameter only applies if you use the `--add` parameter.	—
--scope	String	The application scope from which to download dependencies. This parameter is required if you use the `--add` parameter.	—

For example:

```
now-sdk dependencies --directory /path/to/package --auth devuser1
 --add sys_ui_view fa776f6d97700100f309124eda2975bc --scope global
```

For more information, see Downloading dependencies with the ServiceNow SDK.

transform

Download application metadata (XML) from the instance and transform the metadata into ServiceNow Fluent source code to synchronize the application changes on the instance into your local application.

After initializing an application, you can run the `transform` command without any parameters to transform new application metadata from the instance into source code in the `src/fluent/generated` directory and synchronize changes to metadata into source code in the `src/fluent` directory. To transform metadata that existed when the application was initialized into source code, use the `--from` parameter to provide the path to a local directory or file that contains XML. If metadata exists in the local application as both XML and source code, the XML version takes precedence when installed on the instance.

The `transform` command has the following structure:

```
now-sdk transform [--from <path>] [--directory <package path>]
 [--preview <flag>] [--auth <alias>] [--format <flag>]
```

Optional parameters

Parameter	Type	Description	Default value
--from	String	A path to a local directory or file that contains metadata XML to	—

Optional parameters (continued)

Parameter	Type	Description	Default value
		transform into ServiceNow Fluent code.	
--directory	String	The path to the directory that contains the `package.json` file for your application. The `package.json` should be in the base directory of your application.	Current working directory
--preview	Boolean	An option to preview the transformed ServiceNow Fluent code from the command line without saving the changes.	false
--auth, -a	String	An alias for the credentials to use to authenticate to the instance.	If set, the default alias.
--format, -f	Boolean	An option to format new and updated ServiceNow Fluent source code when it's transformed.	true

For example:

```
now-sdk transform --from metadata/update
 --directory /path/to/package --preview true --auth devuser1
 --format true
```

For more information, see Convert an application with the ServiceNow SDK or Build and install an application with the ServiceNow SDK.

download

Download all application metadata (XML) from an application on an instance to compare with the metadata in your local application.

Updates to JavaScript modules aren't included when downloading application metadata from your instance.

The `download` command has the following structure:

```
now-sdk download <directory> [--source <package path>]
 [--incremental <flag>]
```

Required parameters

Parameter	Type	Description	Default value
directory	String	A path to any directory in which to download the metadata.	—

Required parameters (continued)

Parameter	Type	Description	Default value
		Note: This directory should be a different directory from the `metadata` directory in your application.	

Optional parameters

Parameter	Type	Description	Default value
--source	String	The path to the directory that contains the `package.json` file for your application. The `package.json` should be in the base directory of your application.	Current working directory
--incremental	Boolean	An option to download only changes to application metadata made on the instance and recorded in the Customer Updates [sys_update_xml] table.	false

For example:

```
now-sdk download /path/to/directory --source /path/to/package
 --incremental true
```

clean

Remove the build artifacts that were output with the previous build.

The `clean` command has the following structure:

```
now-sdk clean <source>
```

Optional parameters

Parameter	Type	Description	Default value
source	String	The path to the directory that contains the `package.json` file for your application. The `package.json` should be in the base directory of your application.	Current working directory

For example:

```
now-sdk clean /path/to/package
```

pack

Package the build artifacts that were output with the previous build into an installable ZIP file.

The `pack` command has the following structure:

```
now-sdk pack <source>
```

Optional parameters

Parameter	Type	Description	Default value
source	String	The path to the directory that contains the `package.json` file for your application. The `package.json` should be in the base directory of your application.	Current working directory

For example:

```
now-sdk pack /path/to/package
```

ServiceNow Fluent API reference

Use ServiceNow Fluent APIs to define the metadata that makes up scoped applications in source code with the ServiceNow IDE or ServiceNow SDK.

The ServiceNow Fluent API reference includes details about the APIs and examples. For additional examples of the ServiceNow Fluent APIs in use, see the ServiceNow SDK examples GitHub repository.

Related topics

ServiceNow Fluent

ServiceNow Fluent language constructs

ServiceNow Fluent language constructs provide additional functionality for development in source code with ServiceNow Fluent APIs.

Now.ID

The *`Now.ID`* construct is used to specify human-readable unique IDs for metadata defined in source code. *`Now.ID`* uses the format `Now.ID['String' or Number]` with the *`$id`* property in ServiceNow Fluent APIs. When you build an application, the ID is hashed into a unique sys_id.

You can use *`Now.ID`* only to define IDs for metadata and not to reference other metadata in an application. To reference other metadata within the same application, you can assign the object to a *`const`* variable and reference the variable identifier in other objects.

```
import { Role } from "@servicenow/sdk/core"

const managerRole = Role({
    $id: Now.ID['manager_role'],
    name: 'x_snc_example.manager'
})

Record({
```

```
  table: 'some_table',
  data: {
     role: managerRole //"role" is a reference field to sys_user_role
  }
})
```

Now.ref

The *Now.ref* construct is used to reference metadata in a different application that's not defined in source code. *Now.ref* uses the format Now.ref['table', 'sys_id' or {column: 'value'}, {column: 'value'}] with Reference properties in ServiceNow Fluent APIs.

```
import { Role } from "@servicenow/sdk/core"

Role({
    name: 'x_test.admin',
    contains_roles: [
        'x_test.manager',
        Now.ref('sys_user_role', { name: 'x_test.itil' }),              //
 Coalescing ID reference
        Now.ref('sys_user_role', '${itomId}'),                            //
 GUID-based reference
        Now.ref('sys_user_role', '3D82d1a88947942a90b6d8aa25126d439b',
 { name: 'x_test.csm' }),     // GUID with coalescing ID reference
    ],
})
```

Now.include

The *Now.include* construct is used to refer to text content in another file in the same application rather than including the content inline. *Now.include* uses relative file paths in the format Now.include('./path/to/file') with any property in ServiceNow Fluent APIs. Using *Now.include* allows you to code with the appropriate syntax highlighting for the file type and is especially helpful with properties that support other languages like JavaScript, HTML, CSS, and more.

Now.include supports two-way synchronization so that changes to fields from other ServiceNow AI Platform user interfaces are synced into the referenced file's source code and changes to the source code are synced back to metadata across the instance.

```
import { ScriptInclude } from '@servicenow/sdk/core'

ScriptInclude({
    $id: Now.ID['my-script-include'],
    name: 'MyScriptInclude',
    active: true,
    apiName: 'x_scriptincludes.MyScriptInclude',
    script: Now.include('./MyScriptInclude.server.js')  //The actual content of the
 "script" field is contained in the file specified
})
```

Now.attach

The *Now.attach* construct is used to refer to an image file in the same application and attach it to records associated with metadata defined in source code. *Now.attach* uses relative file paths in the format Now.attach('./path/to/file') with properties that support

user images (the user_image field type) in ServiceNow Fluent APIs. `Now.attach` supports the following file formats: `.jpg`, `.jpeg`, `.png`, `.gif`, `.bmp`, `.ico`, and `.svg`.

`Now.attach` supports two-way synchronization so that changes to image files from other ServiceNow AI Platform user interfaces are synced into the application in source code and changes to image files in source code are synced back to metadata across the instance.

```
import { Record } from '@servicenow/sdk/core'

Record({
    $id: Now.ID['sample-service-portal'],
    table: 'sp_portal',
    data: {
        title: 'Sample Portal',
        url_suffix: 'sample',
        icon: Now.attach('../../assets/servicenow.jpg')
    }
})
```

You can also reuse an image in an application by assigning the `Now.attach` construct to a `const` variable and referencing the variable identifier in other properties. For example:

```
import { Record } from '@servicenow/sdk/core'

const image = Now.attach('../../assets/servicenow.jpg')

Record({
    $id: Now.ID['test'],
    table: 'sp_portal',
    data: {
        title: 'Test Portal',
        url_suffix: 'test',
        icon: image,
        logo: image,
    }
})
```

Access Control List API - ServiceNow Fluent

The Access Control List API defines access control lists [sys_security_acl] that secure parts of an application.

For general information about access control lists (ACLs), see Access Control List Rules.

Related topics

ServiceNow Fluent

ACL object

Configure a custom ACL rule [sys_security_acl] to secure access to new objects or to change the default security behavior.

ACLs must include one or more roles, a security attribute, a condition, or a script.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
operation	String	Required. The operation that this ACL rule secures. An ACL rule can only secure one operation. To secure multiple operations, create a separate ACL rule for each. The operation must be `execute` if the *`type`* property is client_callable_flow_object, client_callable_script_include, graphql, processor, or rest_endpoint. Valid values: • execute: Allow users to execute scripts on a record or UI page. • create: Allow users to insert new records (rows) into a table. • read: Allow users to display records from a table. • write: Allow users to update records in a table. • delete: Allow users to remove records from a table or drop a table. • conditional_table_query_range: Enables users to give partial ACL-access based on read ACLs. Created for the tables that have the read ACLs without Data condition and script. • data_fabric: Enable users to allow Data Fabric connectors to access data on a particular table. • query_match: Enables users to submit match queries (such as "is", "is not", and "is empty"). • query_range: Enables users to submit range queries (such as "starts with", "ends with", and "contains") and sorting is unrestricted. • edit_task_relations: Allow users to extend the Task [task] table. • edit_ci_relations: Allow users to extend the Configuration Item [cmdb_ci] table. • save_as_template: Allow users to save a record as a template. • add_to_list: Allow users from viewing or personalizing specific columns in the list mechanic. Conditions and scripts aren't supported. • report_on: Allow users to report on tables.

Properties (continued)

Name	Type	Description
		• list_edit: Allow users to update records (rows) from a list. • report_view: Allow users to report on field ACLs. • personalize_choices: Allow users to configure the table or field.
type	String	Required. The type of object that this ACL rule secures. The type determines which operations are available. After creating an ACL rule, if you want to change the type, you must delete the ACL and create a new one with the correct type. Valid values: • record • rest_endpoint • ui_page • processor • graphql • pd_action • ux_data_broker • ux_page • ux_route • client_callable_flow_object • client_callable_script_include Default: record
active	Boolean	Flag that indicates whether the ACL rule is enforced. Valid values: • true: The ACL rule is enforced. • false: The ACL rule isn't enforced. Default: true
adminOverrides	Boolean	Flag that indicates whether users with the admin role automatically pass the permissions check for this ACL rule. Valid values: • true: Administrators automatically pass the permissions check. If true, admin users pass the permissions check regardless of what script or role restrictions apply.

Properties (continued)

Name	Type	Description
		However, the nobody role, which only ServiceNow personnel can assign, takes precedence over the admin override option. If an ACL is assigned the nobody role, admin users cannot access the resource even when *adminOverrides* is true. For more information, see Base system roles. • false: Administrators must meet the permissions defined in this ACL rule to gain access to the secured object. Use the *condition* or *script* properties to create a permissions check that administrators must pass. Default: true
script	Script	A custom script that defines the permissions required to access the object. This property supports a function from a JavaScript module, a reference to another file in the application that contains a script, or inline JavaScript. ACLs must include one or more roles, a security attribute, a condition, or a script. **Note:** If the *type* property is `graphql`, scripts aren't supported. The script can use the values of the `current` and `previous` global variables and system properties. The script must generate a true or false response in one of two ways: • return an `answer` variable set to a value of true or false • evaluate to true or false In either case, users only gain access to the object when the script evaluates to true and the user meets any conditions the ACL rule has. Both the conditions and the script must evaluate to true for a user to access the object. **Note:** If the evaluated item is in a related list, `current` points to the item the related list is on, not to the current item the ACL is for. However, If the item you are evaluating the ACL for is not in a related list, `current` points to the actual item. Format:

Properties (continued)

Name	Type	Description
		• For functions, use the name of a function, function expression, or default function exported from a JavaScript module and import it into the `.now.ts` file. For information about JavaScript modules, see JavaScript modules and third-party libraries. • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script'` or `` `Script` ``.
description	String	A description of the object or permissions this ACL rule secures.
localOrExisting	String	The type of security attribute to apply. Valid values: • Local: A security attribute based on the *`condition`* property that is saved only for the ACL it is created in. • Existing: An existing security attribute to reference in the *`security_attribute`* property. Default: Local
decisionType	String	An option for whether the ACL should allow or deny access. Valid values: • allow: The ACL allows access. • deny: The ACL denies access. Default: allow
condition	String	A filter query that specifies the fields and values that must be true for users to access the object. For more information, see Operators available for filters and queries. ACLs must include one or more roles, a security attribute, a condition, or a script.
roles	Array	A list of variable identifiers of *`Role`* objects or sys_ids of roles that a user must have to access the object. For more information, see Role API - ServiceNow Fluent. ACLs must include one or more roles, a security attribute, a condition, or a script.

Properties (continued)

Name	Type	Description
		Note: Users with the admin role always pass this permissions check because the admin role automatically grants users all other roles.
securityAttribute	String	Pre-defined conditions for the ACL to use. For example, whether a user is impersonating another user. For more information about security attributes, see OOB (Out-of-Box) Security Attributes. ACLs must include one or more roles, a security attribute, a condition, or a script. **Note:** For security attributes with the *Is localized* field set to true, the *localOrExisting* property of the ACL should be set to Local. If the *Is localized* field is false, the *localOrExisting* property should be set to Existing.
table	String	The name of the table to which the ACL applies. This property only applies and is required if the *type* property is one of the following values: ux_data_broker, ux_page, ux_route, pd_action, or record.
field	String	The name of a field on the table to secure. You can use the wildcard character ("*") to select all fields.
name	String	The name of the ACL. This property only applies and is required if the *type* property is one of the following values: rest_endpoint, ui_page, processor, graphql, client_callable_flow_object, or client_callable_script_include.
protectionPolicy	String	A policy that determines whether someone can view or edit the script include after the application is installed on their instance. If undefined, other application developers can customize the script include. Valid values: • read: Allows anyone to read values from this downloaded or installed script include. No one can change script values on the instance on which they download or install the script include. • protected: Provides intellectual property protection for application developers. Customers who download the script include cannot see the contents of the script field. The script is encrypted in memory to prevent unauthorized users from seeing it in plain text.

Properties (continued)

Name	Type	Description
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { Acl } from "@servicenow/sdk/core";

export default Acl({
    $id: Now.ID['task_delete_acl'],
    active: true,
    adminOverrides: true,
    type: 'record',
    table: 'task',
    field: 'description',
    operation: 'delete',
    roles: [adminRole, managerRole],
})
```

The roles referenced are defined using the *Role* object:

```
import { Role } from "@servicenow/sdk/core";

const managerRole = Role({
   $id: Now.ID['manager_role'],
   name: 'x_snc_example.manager'
})

const adminRole = Role({
   $id: Now.ID['admin_role'],
   name: 'x_snc_example.admin',
   containsRoles: [managerRole]
})
```

Application Menu API - ServiceNow Fluent

The Application Menu API defines menus in the application navigator [sys_app_application].

For general information about application menus, see Create an application menu.

Related topics

ServiceNow Fluent

ApplicationMenu object

Create a menu for an application [sys_app_application].

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
title	String	Required. The label for the menu in the application navigator.
active	Boolean	Flag that indicates whether the menu appears in the application navigator. Valid values: • true: The menu appears. • false: The menu is hidden. Default: true
roles	Array	A list of variable identifiers of *`Role`* objects or names of roles that can access the menu. For more information, see Role API - ServiceNow Fluent.
category	Reference	The variable identifier of a menu category [sys_app_category] that defines the navigation menu style. To define a menu category, use the Record API - ServiceNow Fluent. For general information about menu categories, see Customize menu categories.
hint	String	A short description of the menu that displays as tooltip when hovering over it.
description	String	Additional information about what the application does.
name	String	An internal name to differentiate between applications with the same title.
order	Number	The relative position of the application menu in the application navigator. Default: 100
$meta	Object	Metadata for the application metadata.

Properties (continued)

Name	Type	Description
		With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { ApplicationMenu } from "@servicenow/sdk/core";

ApplicationMenu({
   $id: Now.ID['my_app_menu'],
   title: 'My App Menu',
   hint: 'This is a hint',
   description: 'This is a description',
   category: appCategory,
   roles: ['admin'],
   active: true,
})
```

The category referenced is defined using the *Record* object:

```
import { Record } from "@servicenow/sdk/core";

export const appCategory = Record({
   table: 'sys_app_category',
   $id: Now.ID[9],
   data: {
      name: 'example',
      style: 'border-color: #a7cded; background-color: #e3f3ff;',
   },
})
```

Automated Test Framework Test API - ServiceNow Fluent

The Automated Test Framework Test API defines automated tests [sys_atf_test] that you can run to confirm that your instance works after making a change.

For general information about Automated Test Framework tests, see Test your apps with the ATF.

Related topics

ServiceNow Fluent

Test object

Create an automated test [sys_atf_test] containing a series of steps to execute.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	A unique name for the test.
description	String	A description of what the test does.
active	Boolean	Flag that indicates whether the test is active. Valid values: • true: The test is active. • false: The test isn't active. Default: true
failOnServerError	Boolean	Flag that indicates whether to fail when a server error occurs during the test. Valid values: • true: The test fails when a server error occurs. • false: The test doesn't fail when a server error occurs. Default: true
configurationFunction	Function	The steps of the test. Test steps are passed as statements within the *atf* function. For example: `(atf) => { atf.form.openNewForm({ table: 'sn_example_table', formUI: 'standard_ui', view: '', }) }` For more information about test steps, see Supported test steps.
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances.

Properties (continued)

Name	Type	Description
		`$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

The output values of test steps with output variables can be saved as variables and used as inputs to other steps using the syntax `output.<output-variable>`. The output variables can be used both directly as inputs on appropriate fields or inside of a template string, such as with `atf.server.log` in the following example.

```
import { Test } from '@servicenow/sdk/core'

Test({
        active: true,
        failOnServerError: true,
        name: 'Simple example',
        description: 'An illustrative test written in fluent',
        $id: Now.ID[1],
    },
    (atf) => {
        atf.form.openNewForm({
            table: 'sn_table_app_reptile_table',
            formUI: 'standard_ui',
            view: '',
        })
        atf.form.setFieldValue({
            table: 'sn_table_app_reptile_table',
            formUI: 'standard_ui',
            fieldValues: {
                reptiles: 'lizard' as any,
            },
        })
        const output =
 atf.form.submitForm({ assertType: 'form_submitted_to_server',
 formUI: 'standard_ui' })
        atf.server.recordValidation({
            recordId: output.record_id,
            table: 'sn_table_app_reptile_table',
            assertType: 'record_validated',
            enforceSecurity: true,
            fieldValues: 'reptiles=lizard',
        })
        atf.server.log({
```

```
            log: `Submitted record with sys_id:
 ${output.record_id} to table ${output.table}`
        })
    }
)
```

Supported test steps

The following test steps are supported. For information about step properties, see the Test step categories documentation.

Note:
Some fields available for test steps on forms aren't available as properties in ServiceNow Fluent.

Test steps

Category	Steps
Application Navigator category	• atf.applicationNavigator.applicationMenuVisibility • atf.applicationNavigator.moduleVisibility • atf.applicationNavigator.navigateToModule
Email category	• atf.email.generateInboundEmail • atf.email.generateInboundReplyEmail • atf.email.generateRandomString • atf.email.validateOutboundEmail • atf.email.validateOutboundEmailGeneratedByFlow • atf.email.validateOutboundEmailGeneratedByNotification
Form category	• atf.form.addAttachmentsToForm • atf.form.clickDeclarativeAction • atf.form.clickModalButton • atf.form.clickUIAction • atf.form.declarativeActionVisibility • atf.form.fieldStateValidation • atf.form.fieldValueValidation • atf.form.openExistingRecord • atf.form.openNewForm • atf.form.setFieldValue • atf.form.submitForm • atf.form.uiActionVisibility

Test steps (continued)

Category	Steps
Forms in Service Portal category	• atf.form_SP.addAttachmentsToForm • atf.form_SP.clickUIAction_SP • atf.form_SP.fieldStateValidation_SP • atf.form_SP.fieldValueValidation_SP • atf.form_SP.openForm_SP • atf.form_SP.openServicePortalPage • atf.form_SP.setFieldValue_SP • atf.form_SP.submitForm_SP • atf.form_SP.uiActionVisibilityValidation_SP
Quick start tests for Dashboards	• atf.reporting.responsiveDashboard • atf.reporting.responsiveDashboardSharing
REST category	• atf.rest.assertJsonResponsePayloadElement • atf.rest.assertResponseHeader • atf.rest.assertResponseJSONPayloadIsValid • atf.rest.assertResponsePayload • atf.rest.assertResponseTime • atf.rest.assertResponseXMLPayloadIsWellFormed • atf.rest.assertStatusCode • atf.rest.assertStatusCodeName • atf.rest.assertXMLResponsePayloadElement • atf.rest.sendRestRequest
Server category	• atf.server.addAttachmentsToExistingRecord • atf.server.checkoutShoppingCart • atf.server.createUser • atf.server.impersonate • atf.server.log • atf.server.recordDelete • atf.server.recordInsert • atf.server.recordQuery • atf.server.recordUpdate • atf.server.recordValidation • atf.server.replayRequestItem

Test steps (continued)

Category	Steps
	• atf.server.runServerSideScript • atf.server.searchForCatalogItem • atf.server.setOutputVariables
Service Catalog category	• atf.catalog.addItemToShoppingCart • atf.catalog.openCatalogItem • atf.catalog.openRecordProducer • atf.catalog.orderCatalogItem • atf.catalog.setCatalogItemQuantity • atf.catalog.setVariableValue • atf.catalog.submitRecordProducer • atf.catalog.validatePriceAndRecurringPrice • atf.catalog.variableStateValidation • atf.catalog.validateVariableValue
Service Catalog in Service Portal category	• atf.catalog_SP.addItemtoShoppingCart_SP • atf.catalog_SP.addOrderGuidetoShoppingCart_SP • atf.catalog_SP.addRowToMultiRowVariableSet_SP • atf.catalog_SP.navigatewithinOrderGuide_SP • atf.catalog_SP.openCatalogItem_SP • atf.catalog_SP.openOrderGuide_SP • atf.catalog_SP.openRecordProducer_SP • atf.catalog_SP.orderCatalogItem_SP • atf.catalog_SP.reviewIteminOrderGuide_SP • atf.catalog_SP.reviewOrderGuideSummary_SP • atf.catalog_SP.saveCurrentRowOfMultiRowVariableSet_SP • atf.catalog_SP.setCatalogItemQuantity_SP • atf.catalog_SP.setVariableValue_SP • atf.catalog_SP.submitOrderGuide_SP • atf.catalog_SP.submitRecordProducer_SP • atf.catalog_SP.validateOrderGuideItem_SP • atf.catalog_SP.validatePriceAndRecurringPrice_SP • atf.catalog_SP.variableStateValidation_SP • atf.catalog_SP.validateVariableValue_SP

Business Rule API - ServiceNow Fluent

The Business Rule API defines server-sides scripts [sys_script] that run when a record is displayed, inserted, updated, or deleted, or when a table is queried.

For general information about business rules, see Classic Business rules.

Related topics

ServiceNow Fluent

BusinessRule object

Create a business rule [sys_script] that automatically changes values in form fields when certain server-side conditions are met.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A name for the business rule.
table	String	Required. The name of the table on which the business rule runs.
script	Script	A custom script runs when the defined conditions are true. This property supports a function from a JavaScript module, a reference to another file in the application that contains a script, or inline JavaScript. Format: • For functions, use the name of a function, function expression, or default function exported from a JavaScript module and import it into the `.now.ts` file. For information about JavaScript modules, see JavaScript modules and third-party libraries. • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or `Script``.
order	Number	A number indicating the sequence in which the business rule should run. If there are multiple rules on a particular activity, the rules run in the order specified, from lowest to highest. Default: 100
active	Boolean	Flag that indicates whether the business rule is enabled.

Properties (continued)

Name	Type	Description
		Valid values: • true: The business rule is enforced. • false: The business rule isn't enforced. Default: true
when	String	The time that the business rule should execute. For more information about when business rules run, see How business rules work. Valid values: • before • after • async • display Default: before
action	Array	The record options that the business rule applies to. For more information about business rule actions, see How business rules work. Valid values: • insert • update • delete • query
addMessage	Boolean	Flag that indicates whether to add a message that appears when the business rule runs. Valid values: • true: Displays a message. • false: Doesn't display a message. Default: false
abortAction	Boolean	Flag that indicates whether to abort the current database transaction. For example, on a before insert business rule, if the conditions are met, don't insert the record into the database. Valid values: • false: Doesn't abort the current database transaction. • true: Aborts the current database transaction. You can't perform additional actions on the record,

Properties (continued)

Name	Type	Description
		such as setting field values and running scripts. You can still display a message to users with the *addMessage* and *message* properties. Default: false
message	String	A message that appears when the business rule runs.
roleConditions	Array	A list of variable identifiers of *Role* objects that users who are modifying records in the table must have for the business rule to run. For more information, see Role API - ServiceNow Fluent.
condition	String	A JavaScript conditional statement that specifies the fields and values that must be true for the script to run. This property supports inline JavaScript or a reference to another file in the application that contains a script. **Note:** Don't use this property if you include the condition statement with the *filterCondition* or *script* properties. By specifying the condition statement with this property, the condition is evaluated separately, and the business rule runs only if the condition is true. To have the condition statement reevaluated a second time before running an async business rule, set the *glide.businessrule.async_condition_check* system property to true. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script'` or `` `Script` ``.
filterCondition	String	A filter query that specifies the fields and values that must be true for the business rule to run. For more information, see Operators available for filters and queries.
setFieldValue	String	The values to set for fields in the table. This can be provided as an encoded query, such as `setFieldValue: 'sec_created=false^EQ'`.
description	String	A description of what the business rule does.
protectionPolicy	String	A policy that determines whether someone can view or edit the script include after the application is installed

Properties (continued)

Name	Type	Description
		on their instance. If undefined, other application developers can customize the script include. Valid values: • read: Allows anyone to read values from this downloaded or installed script include. No one can change script values on the instance on which they download or install the script include. • protected: Provides intellectual property protection for application developers. Customers who download the script include cannot see the contents of the script field. The script is encrypted in memory to prevent unauthorized users from seeing it in plain text.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { BusinessRule } from '@servicenow/sdk/core'
import { FunctionExport, FunctionExpression } from '../server/scripts.js'
import DefaultExportFunction from '../server/scripts.js'

const BR1 = BusinessRule({
    name: 'exportedFunction',
    table: 'x_snc_table',
    when: 'after',
    action: ['update', 'delete', 'insert'],
    script: FunctionExport,
    order: 100,
    active: true,
    addMessage: false,
    message: '<p>message</p>',
    abortAction: false,
    $id: Now.ID[0],
})

const BR2 = BusinessRule({
```

```
    name: 'businessrule2',
    table: 'x_snc_table',
    script: FunctionExpression,
    when: 'after',
    action: ['update'],
    $id: Now.ID[1],
})

const BR3 = BusinessRule({
    name: 'businessrule3',
    table: 'x_snc_table',
    script: DefaultExportFunction,
    when: 'after',
    action: ['update'],
    filterCondition:
 `sys_updated_onSTARTSWITHb^sys_updated_bySTARTSWITHm^EQ
    <item goto="false" or="false" field="sys_updated_on" endquery="false"
 value="b" operator="STARTSWITH" newquery="false"/>
    <item goto="false" or="false" field="sys_updated_by" endquery="false"
 value="m" operator="STARTSWITH" newquery="false"/>
    <item goto="false" or="false" field="" endquery="true" value=""
 operator="=" newquery="false"/>`,
    $id: Now.ID[2],
})

const BR4 = BusinessRule({
    name: 'templateBR',
    action: ['insert'],
    when: 'after',
    table: 'x_snc_table',
    roleConditions: [admin],
    order: 100,
    active: true,
    addMessage: true,
    message: '<p>message</p>',
    script: `gs.info('info')`,
    abortAction: false,
    $id: Now.ID[3],
})
```

Client Script API - ServiceNow Fluent

The Client Script API defines client-side scripts [sys_script_client] that run JavaScript on the client (web browser) when client-based events occur, such as when a form loads, after form submission, or when a field changes value.

For general information about client scripts, see Client scripts.

Related topics

ServiceNow Fluent

ClientScript object

Create a client script [sys_script_client] to configure forms, form fields, and field values while the user is using the form.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
table	String	Required. The name of the table on which the client script runs.
name	String	Required. The name of the client script.
active	Boolean	Flag that indicates whether the client script is enabled. Valid values: • true: The script is enabled. • false: The script isn't enabled. Default: true
appliesExtended	Boolean	Flag that indicates whether the client script applies to tables extended from the specified table. Valid values: • true: The script applies to extended tables. • false: The script doesn't apply to extended tables. Default: false
uiType	String	The user interface to which the client script applies. Valid values: • desktop • mobile_or_service_portal • all Default: desktop
description	String	A description of the functionality and purpose of the client script.
messages	String	Text strings that are available to the client script as localized messages using `getmessage('[message]')`. For more information, see Translate a client script message.
isolateScript	Boolean	Flag that indicates whether the script runs in strict mode, with access to direct DOM, jQuery, prototype, and the window object turned off. Valid values:

Properties (continued)

Name	Type	Description
		• true: Isolate the script and don't run it in strict mode. • false: Run the script in strict mode. Default: false
script	Script	A client-side script that runs in the browser. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script'` or `` `Script` ``.
global	Boolean	Flag that indicates on which views of the table the client script runs. Valid values: • true: The script runs on all views. • false: The script runs only on the specified views. Default: true
view	String	The views of the table on which the client script runs. This property applies only when the *`global`* property is set to false.
type	String	The type of client script, which defines when it runs. For more information about the supported types, see Client scripts. Valid values: • onCellEdit: Runs when the list editor changes a cell value. • onChange: Runs when a particular field value changes on the form. • onLoad: Runs when the system first renders the form and before users can enter data. Typically, *`onLoad()`* client scripts perform client-side-manipulation of the current form or set default record values. • onSubmit: Runs when a form is submitted. Typically, *`onSubmit()`* scripts validate things on the form

Properties (continued)

Name	Type	Description
		and ensure that the submission makes sense. An *onSubmit()* client script can cancel form submission by returning a value of false.
field	String	A field on the table that the client script applies to. This property applies only when the *type* property is set to onChange or onCellEdit.
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { ClientScript } from '@servicenow/sdk/core'

export const cs = ClientScript({
    $id: Now.ID['my_scripts'],
    name: 'my_scripts',
    table: 'incident',
    active: true,
    appliesExtended: false,
    global: true,
    uiType: 'all',
    messages: '',
    isolateScript: false,
    type: 'onLoad',
    script: Now.include('../client/client-script.js'),
})
```

The client script is defined in the `client-script.js` file referenced from the *script* property. For example:

```
function onLoad() {
     const x = 'util' g_form.addInfoMessage(x)
}
```

Cross-Scope Privilege API - ServiceNow Fluent

The Cross-Scope Privilege API defines cross-scope privileges [sys_scope_privilege] for runtime access tracking.

Runtime access tracking allows administrators to manage script access to application resources by creating a list of script operations and targets that the system authorizes to run. For general information about cross-scope privileges, see Cross-scope privilege record.

Related topics

ServiceNow Fluent

CrossScopePrivilege object

Configure cross-scope privileges [sys_scope_privilege] that determine which script operations and targets the system allows to run in the application.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
status	String	Required. The authorization for this record. Valid values: • requested • allowed • denied
operation	String	Required. The operation that the script performs on the target. The target type determines the available operations. Tables [sys_db_object] support the read, write, create, and delete operations. Script includes [sys_script_include] and script objects [sys_db_object] only support the execute operation. Valid values: • create • delete • read • write • execute
targetName	String	Required. The name of the table, script include, or script object being requested.
targetScope	String	Required. The application scope from which resources are requested.
targetType	String	Required. The type of request: script include, script object, or table. Valid values:

Properties (continued)

Name	Type	Description
		• sys_script_include • scriptable • sys_db_object
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { CrossScopePrivilege } from '@servicenow/sdk/core'

CrossScopePrivilege({
   $id: Now.ID['cross_1'],
   status: 'allowed',
   operation: 'execute',
   targetName: 'Script type',
   targetScope: 'x_snc_example',
   targetType: 'scriptable',
})
```

Dashboard API - ServiceNow Fluent

The Dashboard API defines dashboards [par_dashboard] for organizing and sharing data visually.

A dashboard consists of tabs, widgets, visibilities, and permissions. Each tab contains widgets that display data visualizations, headings, rich text, and other components.

Dashboards can be used as the home page of a workspace by referring to one or more workspaces from the *visibilities* array of the *Dashboard* object. To create a workspace, see Workspace API - ServiceNow Fluent.

For general information about dashboards, see Dashboards in Platform Analytics.

Related topics

ServiceNow Fluent

Dashboard object

Create a shareable dashboard [par_dashboard] with data visualizations, filters, tabs, widgets, permissions, and visibility rules.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A name to display for the dashboard.
active	Boolean	Flag that indicates whether the dashboard is active. Default: true
tabs	Array	A list of tabs to display in the dashboard. For more information, see tabs array.
permissions	Array	A list of user permissions required to access the dashboard. For more information, see permissions array.
visibilities	Array	A list of visibility rules that control which UX experiences display the dashboard. For more information, see visibilities array. Default: A default visibility rule with sys_id 08c73d60537101100834ddeeff7b1287 is used.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { Dashboard } from '@servicenow/sdk/core'

Dashboard({
    $id: Now.ID['incident-dashboard'],
    name: 'Incident Management Dashboard',
    tabs: [
        {
```

```
            $id: Now.ID['overview'],
            name: 'Overview',
            widgets: [
                {
                    $id: Now.ID['incident-count-chart'],
                    component: 'vertical-bar', // Vertical bar chart
                    componentProps: {
                        title: 'Incident Count',
                        dataSource: 'incident',
                        aggregation: 'count'
                    },
                    height: 8,
                    width: 6,
                    position: { x: 0, y: 0 },
                },
                {
                    $id: Now.ID['recent-incidents-list'],
                    component: 'list', // List component
                    componentProps: {
                        table: 'incident',
                        filter: 'active=true',
                        limit: 10,
                        columns:
['number', 'short_description', 'priority', 'state']
                    },
                    height: 8,
                    width: 6,
                    position: { x: 6, y: 0 },
                }
            ],
        },
        {
            $id: Now.ID['analytics'],
            name: 'Analytics',
            widgets: [
                {
                    $id: Now.ID['metrics-widget'],
                    component: 'single-score', // Single score component
                    componentProps: {
                        title: 'Key Metrics',
                        scoreSize: 'large'
                    },
                    height: 12,
                    width: 12,
                    position: { x: 0, y: 0 },
                }
            ],
        }
    ],
    permissions: [
        {
            $id: Now.ID['admin-user-permission'],
            user: '6816f79cc0a8016401c5a33be04be441', // sys_id of admin user
            canRead: true,
            canWrite: true,
            canShare: true,
            owner: true,
```

```
        },
        {
            $id: Now.ID['itil-role-permission'],
            role: '2831a114c611228501d4ea6c309d626d', // sys_id of itil role
            canRead: true,
        },
        {
            $id: Now.ID['support-group-permission'],
            group: '287ebd7da9fe198100f92cc8d1d2154e', // sys_id of IT Support
group
            canRead: true,
        }
    ],
    visibilities: [
        {
            $id: Now.ID['workspace-visibility'],
            experience: myWorkspace,
        }
    ]
})
```

The workspace referenced is defined using the *Workspace* object:

```
export const myWorkspace = Workspace({
  $id: Now.ID["my_workspace"],
  title: "My Workspace",
  path: "my-workspace",
  tables: ["incident"],
  listConfig: myListConfig
})
```

tabs array

Create tabs [par_dashboard_tab] that contain widgets for a dashboard.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A name to display on the tab.
active	Boolean	Flag that indicates whether the tab is active. Default: true
widgets	Array	A list of widgets to display in the tab. For more information, see widgets array.

Within a dashboard, tabs are ordered using their position in the array.

```
tabs: [
        {
            $id: Now.ID['overview-tab'],
```

```
            name: 'Overview',
            widgets: [
                {
                    $id: Now.ID['chart-widget'],
                    component: 'line', // Line chart component
                    componentProps: {
                        selectedElements: [],
                        chartVariation: 'stacked',
                        yAxisPosition: 'bottom'
                    },
                    height: 12,
                    width: 12,
                    position: { x: 0, y: 0 },
                },
                {
                    $id: Now.ID['header-widget'],
                    component: 'heading', // Heading component
                    componentProps: {
                        variant: 'header-primary',
                        label: 'Dashboard Metrics',
                        level: '1'
                    },
                    height: 4,
                    width: 12,
                    position: { x: 0, y: 12 },
                }
            ],
        },
        {
            $id: Now.ID['details-tab'],
            name: 'Details',
            widgets: []
        }
]
```

widgets array

Create widgets [par_dashboard_widget] within a tab in a grid layout.

Dashboards use a 48-point grid system for positioning widgets.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
component	Reference or String	Required. The name of a component (such as line or single-score) or the sys_id of a component from the UX Macroponent Definition [sys_ux_macroponent] table to display as a widget. Component names aren't case sensitive and resolve to their sys_ids during the build process.

Properties (continued)

Name	Type	Description
height	Number	Required. The height of the widget in grid units.
width	Number	Required. The width of the widget in grid units. Maximum value: 48
position	Object	Required. The position of the widget in the grid layout with `x` and `y` properties. For example, a value of `{ x: 0, y: 0 }` positions the widget in the top-left corner of the grid.
componentProps	Object	The property configuration of a component. For more information, see componentProps object.

```
widgets: [
                {
                    $id: Now.ID['incident-count-chart'],
                    component: 'vertical-bar',  // Vertical bar chart
                    componentProps: {
                        title: 'Incident Count',
                        dataSource: 'incident',
                        aggregation: 'count'
                    },
                    height: 8,
                    width: 6,
                    position: { x: 0, y: 0 },
                },
                {
                    $id: Now.ID['recent-incidents-list'],
                    component: 'list',  // List component
                    componentProps: {
                        table: 'incident',
                        filter: 'active=true',
                        limit: 10,
                        columns:
 ['number', 'short_description', 'priority', 'state']
                    },
                    height: 8,
                    width: 6,
                    position: { x: 6, y: 0 },
                }
            ],
        },
        {
            $id: Now.ID['analytics'],
            name: 'Analytics',
            widgets: [
                {
                    $id: Now.ID['metrics-widget'],
                    component: 'single-score',  // Single score component
                    componentProps: {
                        title: 'Key Metrics',
                        scoreSize: 'large'
                    },
```

```
                height: 12,
                width: 12,
                position: { x: 0, y: 0 },
            }
]
```

componentProps object

Configure the properties of widgets [par_dashboard_widget].

The available properties are determined by the component specified with the *component* property of the *Dashboard* object.

- Trend data components require the *dataSources*, *metrics*, and *trendBy* properties. The *groupBy* property is optional.
- Group data components require the *dataSources*, *metrics*, and *groupBy* properties. The *trendBy* property isn't supported in these visualizations.
- Simple data components require the *dataSources* and *metrics* properties. The *groupBy* and *trendBy* properties aren't supported in these visualizations.

Properties

Name	Type	Description
dataSources	Array	A list of data sources for the component. For example: `dataSources: [ { label: "Incident", // Human-readable label sourceType: "table", // Type of data source tableOrViewName: "incident", // ServiceNow table name filterQuery: "", // Optional encoded query filter id: "data_source_1" // Unique data source ID }, ... ]`
headerTitle	String	A title to display with the widget.
metrics	Array	A list of metrics to measure for the data source. For example: `metrics: [{ dataSource: 'data_source_1', // Must match dataSource id id: 'metric_1', // Unique metric ID aggregateFunction: 'AVG', // COUNT, SUM, AVG, MIN, MAX, COUNT_DISTINCT aggregateField: 'business_duration' // Field to be used for aggregation. axisId: 'primary', // Which axis to display the series }, ...]`

Properties (continued)

Name	Type	Description
groupBy	Array	A list of configurations for grouping and organizing data by data source. For example: `groupBy: [ { groupBy: [ { dataSource: "data_source_1", // Must match dataSource id groupByField: "state" // Field to group by } ], maxNumberOfGroups: 10, // Maximum categories to show showOthers: false // Show "Others" category }, ... ]`
trendBy	Object	A configuration for trend charts. For example: `trendBy: { "trendByFrequency": "year", // Frequency of the trend (date, week, month, year) "trendByFields": [ { "field": "sys_created_on", // Field to trend on (from the table of the dataSource) "metric": "metric_1" // ID of the metric } ] }`
sortBy	String	The method of sorting data. Valid values: • value • label • field

In the following example, the `line` component, which is a trend data type visualization, is used to show how a metric changes over time.

```
{
    component: 'line',
    componentProps: {
        dataSources: [
            {
                label: 'Incident',
                sourceType: 'table',
                tableOrViewName: 'incident',
                filterQuery: '',
```

```
                id: 'data_source_1',
            },
        ],
        headerTitle: 'Incidents by State',
        metrics: [
            {
                dataSource: 'data_source_1',
                id: 'metric_1',
                aggregateFunction: 'COUNT',
                axisId: 'primary',
            },
        ],
        trendBy: {
            trendByFrequency: "year",
            trendByFields: [
                {
                    field: "sys_created_on",
                    metric: "metric_1"
                }
            ]
        },
    },
    height: 14,
    width: 17,
    position: { x: 0, y: 0 },
}
```

In the following example, the `single-score` component, which is a simple data type visualization, is used to show a single count metric.

```
{
    component: 'single-score',
    componentProps: {
        dataSources: [
            {
                label: 'Incident',
                sourceType: 'table',
                tableOrViewName: 'incident',
                filterQuery: '',
                id: 'data_source_1',
            },
        ],
        headerTitle: 'Open Incidents',
        metrics: [
            {
                dataSource: 'data_source_1',
                id: 'metric_1',
                aggregateFunction: 'COUNT',
                axisId: 'primary',
            },
        ]
    },
    height: 14,
    width: 14,
    position: { x: 0, y: 0 },
}
```

permissions array

Define permissions [par_dashboard_permission] to read, edit, and share a dashboard.

At least one of the *user*, *group*, or *role* properties must be specified for each permission in the array.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
user	Reference or String	The variable identifier or sys_id of a user [sys_user] to which to grant permissions. To define a user, use the Record API - ServiceNow Fluent.
group	Reference or String	The variable identifier or sys_id of a user group [sys_user_group] to which to grant permissions. To define a user, use the Record API - ServiceNow Fluent.
role	Reference or String	The variable identifier or sys_id of a role [sys_user_role] to which to grant permissions. To define a user, use the Role API - ServiceNow Fluent.
canRead	Boolean	Flag that indicates whether the user, group, or role can view the dashboard. Default: true
canWrite	Boolean	Flag that indicates whether the user, group, or role can edit the dashboard. Default: false
canShare	Boolean	Flag that indicates whether the user, group, or role can share the dashboard. Default: false
owner	Boolean	Flag that indicates whether the user, group, or role is the owner of the dashboard. For at least one user, the *owner* property should be set to true. Default: false

```
permissions: [
        {
            $id: Now.ID['manager-user-permission'],
            user: 'a8f98bb0eb32010045e1a5115206fe3a', // sys_id of manager
user
            canRead: true,
            canWrite: true,
            owner: true,
        },
```

```
        {
            $id: Now.ID['itil-role-permission'],
            role: '2831a114c611228501d4ea6c309d626d', // sys_id of itil role
            canRead: true,
            canWrite: false,
        },
        {
            $id: Now.ID['support-group-permission'],
            group: 'd625dccec0a8016700a222a0f7900d06', // sys_id of Service
Desk group
            canRead: true,
            canWrite: false,
        }
]
```

visibilities array

Define visibility rules [par_dashboard_visibility] for which UX experiences display the dashboard.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
experience	Reference or String	Required. The variable identifier of a *`Workspace`* object or sys_id of a UX application [sys_ux_page_registry]. For more information, see Workspace API - ServiceNow Fluent.

```
visibilities: [
  {
    $id: Now.ID["dashboard_visibility_1"],
    experience: myWorkspace // Reference to Workspace object
  }
]
```

Email Notification API - ServiceNow Fluent

The Email Notification API defines notifications [sysevent_email_action] that send automated emails based on database operations, custom events, or manual triggers.

For general information about email notifications, see Email and SMS notifications.

Related topics

ServiceNow Fluent

EmailNotification object

Create an email notification [sysevent_email_action] specifying when to send it, who receives it, what it contains, and if it can be delivered in an email digest.

For general information about creating email notifications, see Create an email notification.

Properties

Name	Type	Description
table	Reference or String	Required. The variable identifier or name of a table to which the notification applies. To define a table, use the Table API - ServiceNow Fluent. **Note:** Don't select the Task [task] table, which is for extending other tables. Notifications that run on the Task table directly aren't supported.
triggerConditions	Object	Required. The conditions that trigger the notification. For more information, see triggerConditions object.
name	String	A unique name for the email notification.
description	String	A description of the purpose of the email notification.
category	Reference or String	The variable identifier or name of a notification category for grouping notifications. To define a notification category, use the Record API - ServiceNow Fluent. Default: The default email category (c97d83137f4432005f58108c3ffa917a)
notificationType	String	The type of notification. Valid values: • email: A standard email. • vcalendar: A meeting invitation. Meeting invitations aren't supported with email digests. Default: email
active	Boolean	Flag that indicates whether the notification is active. Default: true
mandatory	Boolean	Flag that indicates whether the notification is required. Default: false
enableDynamicTranslation	Boolean	Flag that indicates whether to enable dynamic translation for the notification. Default: false
emailContent	Object	The email content and formatting. For more information, see emailContent object.
recipientDetails	Object	The email recipients. For more information, see recipientDetails object.
digest	Object	The email digest content and formatting. For more information, see digest object.
$meta	Object	Metadata for the application metadata.

Properties (continued)

Name	Type	Description
		With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { EmailNotification } from '@servicenow/sdk/core'

EmailNotification({
    table: 'incident',
    name: 'Custom Event Notification',
    description: 'Triggered by custom event',
    category: 'c97d83137f4432005f58108c3ffa917a',    // Default email category sys_id
    triggerConditions: {
        generationType: 'event',
        eventName: 'custom.incident.escalated',
        order: 100
    },
    recipientDetails: {
        recipientUsers: ['6816f79cc0a8016401c5a33be04be441'],    // Admin user sys_id
        eventParm1WithRecipient: true,    // Event param 1 contains recipient
        isSubscribableByAllUsers: true
    },
    emailContent: {
        contentType: 'text/html',
        subject: 'Incident Escalated',
        messageHtml: '<p>An incident has been escalated.</p>'
    }
})
```

triggerConditions object

Configure the conditions that must be met for the notification to be sent.

Note:
If the same trigger generates multiple notifications, the system sends only one notification. The system considers all other notifications, even if they have a different subject and body, as duplicates. The Ignore Duplicates business rule controls this functionality.

Properties

Name	Type	Description
generationType	String	Required. The method of triggering the email notification. Valid values: • engine: Sends the notification when a record is inserted or updated. • event: Sends the notification when custom events occur. • triggered: Manually triggered notifications. The following properties don't apply if this property is set to `triggered`.
onRecordInsert	Boolean	Flag that indicates whether to send the notification when a record is inserted. **Note:** Either this property or the *`onRecordUpdate`* property must be true. Default: false
onRecordUpdate	Boolean	Flag that indicates whether to send the notification when a record is updated. **Note:** Either this property or the *`onRecordInsert`* property must be true. Default: false
eventName	String	The name of a custom event to trigger sending the notification. This property is required if the value of the *`generationType`* property is `event`.
affectedFieldOnEvent	String	The event parameter that contains the affected field. This property applies only if the *`generationType`* property is set to `event`. Valid values: • parm1 • parm2
weight	Number	The notification priority relative to duplicate notifications. Notifications that have the same target table and recipients are considered duplicates if the weights are different. If weights are same, an additional evaluation is done to check if the subject and the body (excluding watermark) are the same to qualify as duplicate notification. When there are duplicate notifications, the system only sends the notification with the highest weight. All other notifications are moved from the Outbox to the Skipped mailbox. The default value of 0 causes the system to send the notification (assuming the conditions are met).

Properties (continued)

Name	Type	Description
		Maximum value: 1000 Default: 0
condition	String	A filter query that specifies the fields and values that must be true for users to access the object. For more information, see Operators available for filters and queries.
advancedCondition	String	A JavaScript conditional statement that must return true or set a global answer variable to true to send the notification. For more information, see Advanced conditions for email notifications. **Note:** This property is evaluated in addition to other conditions that you set on the notification. Both the condition and advanced condition must evaluate to true to send the notification.
itemTable	Reference or String	The variable identifier or name of the table to which the notification item refers. To define a table, use the Table API - ServiceNow Fluent.
item	String	The item to use for the notification context.
order	Number	The execution order of the notification. Maximum value: 9999 Default: 100

```
triggerConditions: {
    generationType: "engine",
    onRecordInsert: false,
    onRecordUpdate: true,
    weight: 100,
    condition: "priority=1^ORpriority=2^state!=6^state!=7",  // High/Critical priority,
not resolved/closed
    order: 100
}
```

emailContent object

Configure the contents of an email notification.

Properties

Name	Type	Description
contentType	String	The type of email content. Valid values:

Properties (continued)

Name	Type	Description
		• text/html • text/plain • multipart/mixed Default: text/html
template	Reference or String	The variable identifier or sys_id of an email template [sysevent_email_template]. To define an email template, use the Record API - ServiceNow Fluent. You can only specify an email template that meets one of the following conditions: • The template has the same scope and table as the notification. • The template has the same scope but has no specified table. • The template has the same table and is in the global scope.
style	Reference or String	The variable identifier or sys_id of an email style [sys_email_style]. To define an email style, use the Record API - ServiceNow Fluent.
subject	String	A subject line for the email. If empty, the system uses the Subject value from the email template. You can use the `\${variable}` format for variable references. Variables map to column names available from the notification table, its parent tables, and reference tables. Use variables to include values from a record in the table, such as the short description or comments and work notes.
smsAlternate	String	A notification message to send specifically to SMS devices. The SMS alternate message is limited to 140 characters. If empty, the system uses the SMS alternate value from the email template. You can use the `\${variable}` format for variable references. Variables map to column names available from the notification table, its parent tables, and reference tables. Use variables to include values from a record in the table, such as the short description or comments and work notes.
importance	String	The level of importance of the message. Valid values: • low • high
includeAttachments	Boolean	Flag that indicates whether to include attachments from the notification trigger record with the email.

Properties (continued)

Name	Type	Description
		Default: false
omitWatermark	Boolean	Flag that indicates whether to omit the watermark attached to each email. For more information, see Watermarks on notification emails. Default: false
from	String	The email address from which to send the notification.
replyTo	String	The email address to which recipients can reply.
pushMessageOnly	Boolean	Flag that indicates whether to send the notification only as a push notification to a mobile device. Default: false
pushMessageList	Array	A list of variable identifiers or sys_ids of push messages [sys_push_notif_msg] to associate with the notification. To define a push notification, use the Record API - ServiceNow Fluent. **Note:** The push message and notification must be for the same table.
forceDelivery	Boolean	Flag that indicates whether to bypass user notification preferences that would prevent a notification and send the notification anyway. Default: false
messageHtml	String	The HTML content of the notification message. If empty, the system uses the message HTML from the email template. You can use the `\${variable}` format for variable references. Variables map to column names available from the notification table, its parent tables, and reference tables. Use variables to include values from a record in the table, such as the short description or comments and work notes. This property is required if the value of the *`contentType`* property is `text/html` or `multipart/mixed`.
messageText	String	The plain text content of the notification message. If empty, the system uses the message text from the email template. You can use the `\${variable}` format for variable references. Variables map to column names available from the notification table, its parent tables, and reference tables. Use variables to include values from a record in the table, such as the short description or comments and work notes.

Properties (continued)

Name	Type	Description
		This property is required if the value of the *contentType* property is `text/plain` or `multipart/mixed`.
message	String	Deprecated. Message contents.

```
emailContent: {
    contentType: "text/html",
    subject: "CRITICAL: Incident \${number} - \${short_description}",
    messageHtml: `
            <div style="background-color: #ff4444; color: white; padding: 10px;
border-radius: 5px;">
                <h2>CRITICAL INCIDENT ALERT</h2>
                <p><strong>Incident:</strong> \${number}</p>
                <p><strong>Priority:</strong> \${priority}</p>
                <p><strong>Description:</strong>
 \${short_description}</p>
                <p><strong>Assigned To:</strong>
 \${assigned_to.name}</p>
                <p><strong>Created:</strong>
 \${sys_created_on}</p>
                <p><a href="\${instance_url}/incident.do?sys_id=\${sys_id}"
 style="color: #ffffff; text-decoration: underline;">View Incident</a></p>
            </div>
        `,
    smsAlternate:
      "CRITICAL: Incident \${number} - \${short_description}. Priority: \${priority}.
Assigned: \${assigned_to.name}",
    pushMessageList: ["mobile_push_notification_sys_id"],
    forceDelivery: true,
    importance: "high"
}
```

digest object

Configure the contents of an email digest that summarizes the activity for a selected notification and its target records during a specified time interval.

For general information about email digests, see Email digests.

Properties

Name	Type	Description
allow	Boolean	Flag that indicates whether users to receive this notification as a digest. If false, all other properties of the *digest* object are ignored. Default: false
default	Boolean	Flag that indicates whether digest mode is enabled by default for this notification. Default: false

Properties (continued)

Name	Type	Description
type	String	The type of digest. Valid values: • single: Sends the notification in an email digest when it's triggered multiple times during the selected interval for a single target record, for example INC001 only. • multiple: Sends the notification in an email digest when it's triggered multiple times during the selected interval for a multiple target record, for example INC001, INC002, and more.
defaultInterval	Reference or String	The variable identifier or sys_id of a digest interval [sys_email_digest_interval] to use as the default time interval for digest delivery. To define a digest interval, use the Record API - ServiceNow Fluent.
subject	String	A subject line for the email digest. If empty, the system uses the Subject value from the email template. You can use the `\${variable}` format for variable references. Variables map to column names available from the notification table, its parent tables, and reference tables. Use variables to include values from a record in the table, such as the short description or comments and work notes.
html	String	The HTML content of the email digest. You can use the `\${variable}` format for variable references. Variables map to column names available from the notification table, its parent tables, and reference tables. Use variables to include values from a record in the table, such as the short description or comments and work notes.
text	String	The plain text content of the email digest. You can use the `\${variable}` format for variable references. Variables map to column names available from the notification table, its parent tables, and reference tables. Use variables to include values from a record in the table, such as the short description or comments and work notes.
template	String	The variable identifier or sys_id of an email template [sysevent_email_template]. To define an email template, use the Record API - ServiceNow Fluent.
separatorHtml	String	An HTML separator that appears between each item summarized in the digest. You can use the `\${variable}` format for variable references. Variables map to column names available from the notification table, its parent tables, and reference tables. Use variables to include values from a record in the table, such as the short description or comments and work notes.

Properties (continued)

Name	Type	Description
		Default: `<p> </p>\n<hr>\n<p> </p>`
separatorText	String	A plain text separator that appears between each item summarized in the digest. You can use the `\${variable}` format for variable references. Variables map to column names available from the notification table, its parent tables, and reference tables. Use variables to include values from a record in the table, such as the short description or comments and work notes. Default: `----------` `\n`
from	String	The email address from which to send the notification.
replyTo	String	The email address to which recipients can reply.

```
digest: {
    allow: true,
    defaultInterval: hourlyDigest,
    template: escalationTemplate,
    subject: "Incident Escalation Digest - \${digest_count} incidents require attention",
    html: `<div>
           <div>Incident Escalation Digest</div>
           </div>`,
    separatorHtml: '<hr style="margin: 15px 0;">',
    from: "noreply@company.com",
    replyTo: "itsupport@company.com"
}
```

recipientDetails object

Configure who receives an email notification.

Notification recipients must be active users defined as active users in the User [sys_user] table. They must also have a valid email address defined for their primary channel (device) in the Notification Device [cmn_notif_device] table. Your notification recipients must also have the appropriate notification preferences enabled.

Properties

Name	Type	Description
recipientUsers	Array	A list of variable identifiers or sys_ids of users [sys_user] or a list of email addresses to receive the notification. To define a user, use the Record API - ServiceNow Fluent.
recipientFields	Array	A list of fields that reference the users or user groups to receive the notification. The fields must be reference fields. For example, if a notification uses the Incident [incident] table, you can specify the `opened_by` field to send the notification to users or groups who opened the incident. This

Properties (continued)

Name	Type	Description
		list of users or groups is variable and depends on the values of the associated task record. You can also select a field that includes an email address string to send a notification to that address.
recipientGroups	Array	A list of variable identifiers or sys_ids of user groups [sys_user_group] to receive the notification. To define a user group, use the Record API - ServiceNow Fluent. **Note:** Group members receive individual notifications only if **Include members** is selected in the group record.
excludeDelegates	Boolean	Flag that indicates whether to exclude delegated users. Set this property to true to help prevent the instance from sending email notifications to delegates of the users and members of the groups you selected. Default: false
isSubscribableByAllUsers	Boolean	Flag that indicates whether to allow all users to subscribe to the notification. For more information, see Subscription-based notifications. **Note:** If the record contains sensitive or protected data, consider restricting the recipient list only to those users and groups who normally have access to it, leaving the value of this property as false. You can also configure your notification content so that private or sensitive data isn't exposed. For example, you could insert a link back to the associated record, so that details aren't revealed in the notification. Default: false
sendToCreator	Boolean	Flag that indicates whether to send the notification to the person who performed the action that started the notification process, if the person is also a recipient. If the event creator isn't specified in one of the recipient fields, the event creator doesn't receive a notification regardless of the setting in this field. Default: false
eventParm1WithRecipient	Boolean	Flag that indicates whether the event parameter 1 contains one or more notification recipients. This property only applies to event-based notifications.
eventParm2WithRecipient	Boolean	Flag that indicates whether the event parameter 2 contains one or more notification recipients.

Properties (continued)

Name	Type	Description
		This property only applies to event-based notifications.

```
recipientDetails: {
    recipientGroups: ["d625dccec0a8016700a222a0f7900d06"], // IT Support
group sys_id
    recipientFields: ["assigned_to", "caller_id"],
    sendToCreator: false,
    isSubscribableByAllUsers: false
}
```

Flow API - ServiceNow Fluent

The Flow API defines flows and subflows [sys_hub_flow], which automate business processes with reusable multiple-step components.

Create a flow using the Flow object. For more information, see Flow object.

Create a subflow using the Subflow object. For more information, see Subflow object.

For general information about flows, see Exploring flows. For general information about subflows, see Exploring subflows.

Related topics

ServiceNow Fluent

Flow object

Create a flow [sys_hub_flow] to run a sequence of actions and flow logic when a set of trigger conditions occur.

Flows only run when their trigger conditions are met. Use flows for event-driven automation that requires a consistent and predefined trigger.

A Flow object consists of these components.

- One configuration object that defines the flow configuration properties.
- One wfa.trigger function that defines when to run the flow.
- One Body function that defines what actions and flow logic to run.
  - Zero or more wfa.action functions
  - Zero or more wfa.flow_logic functions

Properties

Name	Type	Description
configuration	Object	Required. An object containing the metadata configuration properties for the Fluent object or function.
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs.

Properties (continued)

Name	Type	Description
		Format: `Now.ID['String' or Number]`
name	String	Required. Display name of the flow. The flow name should be meaningful and descriptive to easily identify its purpose.
description	String	Description of what the flow does.
runAs	String	Determines under which user context the flow actions are run. The system user setting runs as a privileged user that bypasses role-based ACLs. The user setting runs with the privileges and roles of the user who started the flow. Valid values: system, user Default: user
runWithRoles	String	Specify the roles that the flow uses while running. For more information about running a flow with roles, see Flow roles.
flowPriority	String	Defines the execution priority of the flow in the Flow Engine. Valid values: LOW, MEDIUM, HIGH Default: MEDIUM
protection	String	Defines whether the flow is read-only (read) or editable (empty string). Valid values: read, '' Default: ''
access	String	Defines whether a flow is public or private. Valid values: public, package_private Default: public
flowVariables	Object	Object defining data variables available to the flow using Column types. Use the get_table_schema function to fetch field definitions.
wfa.trigger	Function	Defines when to run the flow. When the trigger conditions are met, the system runs the flow using the data provided by the trigger. For more information about the *`wfa.trigger`* function, see wfa.trigger function.
Flow body	Function	*`Flow body`* is an *`Arrow function`* in TypeScript that represents the execution steps in the flow. The Flow Body receives the *`params`* parameter as its input, which contains the *`wfa.trigger`* and *`flowVariables`* objects.The steps in the *`Flow body`* consist of these function types:

Properties (continued)

Name	Type	Description
		• `wfa.action` function: Execute specific tasks and operations within a flow, such as creating records, sending emails, requesting approvals, or integrating with external systems. For more information about the `wfa.action` function, see wfa.action function. • `wfa.flow_logic` function: Control how flows execute, providing conditional branching, iteration, and flow structuring capabilities. For more information about `wfa.flow_logic` function, see wfa.flow_logic function. You can use these parameter values in the Flow body function. **params.trigger.current** Data type: Record. Parameter that stores the current record that started a create record, update record, or create or update record flow. **params.trigger.previous** Data type: Record. Parameter that stores the previous record values that started an update record flow. **params.trigger.table_name** Data type: String. Parameter that stores the table name of the current record that started the flow. **params.trigger.flowVariables** Data type: Object. Parameter that stores the flow variables defined in this flow.

This example creates a flow that runs when an incident is created with an empty origin field value. The flow alerts managers and team members based on the incident severity.

```
import { action, Flow, wfa, trigger } from '@servicenow/sdk/automation'

export const incidentSeverityAlertFlow = Flow(
    {
        $id: Now.ID['incident_severity_alert_flow'],
        name: 'Incident Severity Alert Flow',
        description: 'Alerts managers and team members based on incident severity
for incidents with empty origin',
    },
    wfa.trigger(
        trigger.record.created,
        { $id: Now.ID['empty_origin_incident_trigger'] },
        {
            table: 'incident',
            condition: 'origin=NULL',
            run_flow_in: 'background',
            run_on_extended: 'false',
            run_when_setting: 'both',
            run_when_user_setting: 'any',
```

```
            run_when_user_list: [],
        }
    ),
    (params) => {
        wfa.action(
            action.core.log,
            { $id: Now.ID['log_incident_short_description'] },
            {
                log_level: 'info',
                log_message: `Incident created:
 ${wfa.dataPill(params.trigger.current.short_description, 'string')
}`,
            }
        )

        wfa.flowLogic.if(
            {
                $id: Now.ID['check_severity_high'],
                condition:
 `${wfa.dataPill(params.trigger.current.severity, 'string')}=1`,
                annotation: 'High severity (1)',
            },
            () => {
                const assignmentGroup = wfa.action(
                    action.core.lookUpRecord,
                    { $id: Now.ID['lookup_assignment_group'] },
                    {
                        table: 'sys_user_group',
                        conditions:
 `sys_id=${wfa.dataPill(params.trigger.current.assignment_group,
 'reference')}`,
                        sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
                    }
                )

                const manager = wfa.action(
                    action.core.lookUpRecord,
                    { $id: Now.ID['lookup_manager_details'] },
                    {
                        table: 'sys_user',
                        conditions:
 `sys_id=${wfa.dataPill(assignmentGroup.Record.manager, 'reference')
}`,
                        sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
                    }
                )

                wfa.action(
                    action.core.sendNotification,
                    { $id: Now.ID['send_urgent_email_to_manager'] },
                    {
                        table_name: 'incident',
```

```
                        record:
 wfa.dataPill(params.trigger.current.sys_id, 'reference'),

 notification: 'high_severity_incident_manager_notification',
                    }
                )

                wfa.flowLogic.forEach(

 wfa.dataPill(params.trigger.current.additional_assignee_list, 'a
rray.string'),
                    { $id: Now.ID['foreach_additional_assignees_high'] },
                    () => {
                        const assignee = wfa.action(
                            action.core.lookUpRecord,
                            { $id:
 Now.ID['lookup_assignee_details_high'] },
                            {
                                table: 'sys_user',
                                conditions:
 `sys_id=${wfa.dataPill(params.trigger.current.additional_assign
ee_list, 'array.string')}`,
                                sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
                            }
                        )

                        wfa.action(
                            action.core.sendSms,
                            { $id:
 Now.ID['send_sms_to_assignee_high'] },
                            {
                                recipients:
 `${wfa.dataPill(assignee.Record.phone, 'string')}`,
                                message: `High severity
 incident:
 ${wfa.dataPill(params.trigger.current.short_description, 'string')
}`,
                            }
                        )
                    }
                )

                wfa.action(
                    action.core.updateRecord,
                    { $id: Now.ID['update_work_notes_high'] },
                    {
                        table_name: 'incident',
                        record:
 wfa.dataPill(params.trigger.current.sys_id, 'reference'),
                        values: TemplateValue({
                            work_notes: `Manager
 ${wfa.dataPill(manager.Record.name, 'string')} notified via email
 and team notified via SMS.`,
                        }),
                    }
```

```
                )
            }
        )

        wfa.flowLogic.elseIf(
            {
                $id: Now.ID['check_severity_medium'],
                condition:
 `${wfa.dataPill(params.trigger.current.severity, 'string')}=2`,
                annotation: 'Medium severity (2)',
            },
            () => {
                wfa.flowLogic.forEach(

 wfa.dataPill(params.trigger.current.additional_assignee_list, 'a
rray.string'),
                    { $id:
 Now.ID['foreach_additional_assignees_medium'] },
                    () => {
                        const assignee = wfa.action(
                            action.core.lookUpRecord,
                            { $id:
 Now.ID['lookup_assignee_details_medium'] },
                            {
                                table: 'sys_user',
                                conditions:
 `sys_id=${wfa.dataPill(params.trigger.current.additional_assign
ee_list, 'array.string')}`,
                                sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
                            }
                        )

                        wfa.action(
                            action.core.sendSms,
                            { $id:
 Now.ID['send_sms_to_assignee_medium'] },
                            {
                                recipients:
 `${wfa.dataPill(assignee.Record.phone, 'string')}`,
                                message: `Medium severity
 incident:
 ${wfa.dataPill(params.trigger.current.short_description, 'string')
}`,
                            }
                        )
                    }
                )
            }
        )

        wfa.action(
            action.core.updateRecord,
            { $id: Now.ID['update_incident_to_in_progress'] },
            {
                table_name: 'incident',
```

```
                record:
 wfa.dataPill(params.trigger.current.sys_id, 'reference'),
                values: TemplateValue({ state: '2' }),
            }
        )
    }
)
```

This example runs when a change request is approved and notifies the requester.

```
import { action, Flow, wfa, trigger } from '@servicenow/sdk/automation'

export const changeRequestApprovalNotificationFlow = Flow(
    {
        $id: Now.ID['change_request_approval_notification_flow'],
        name: 'Change Request Approval Notification Flow',
        description: 'Sends formatted notification to requester when change
 request is approved',
    },
    wfa.trigger(
        trigger.record.updated,
        { $id: Now.ID['change_request_approved_trigger'] },
        {
            table: 'change_request',
            condition: 'approval=approved',
            run_flow_in: 'background',
            trigger_strategy: 'unique_changes',
            run_when_user_list: [],
            run_when_setting: 'both',
            run_on_extended: 'false',
            run_when_user_setting: 'any',
        }
    ),
    (params) => {
        const requester = wfa.action(
            action.core.lookUpRecord,
            { $id: Now.ID['lookup_requester_details'] },
            {
                table: 'sys_user',
                conditions:
 `sys_id=${wfa.dataPill(params.trigger.current.requested_by, 'refer
ence')}`,
                sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
            }
        )

        wfa.action(
            action.core.sendEmail,
            { $id: Now.ID['send_approval_notification_email'] },
            {
                table_name: 'change_request',
                watermark_email: true,
                ah_subject: `Change Request
 ${wfa.dataPill(params.trigger.current.number, 'string')} -
 Approved`,
```

```
                ah_body: `Your change request has been
approved.`,
                record:
wfa.dataPill(params.trigger.current.sys_id, 'reference'),
                ah_to:
wfa.dataPill(requester.Record.email, 'string'),
            }
        )

        wfa.action(
            action.core.updateRecord,
            { $id: Now.ID['update_work_notes_notification_sent'] },
            {
                table_name: 'change_request',
                record:
wfa.dataPill(params.trigger.current.sys_id, 'reference'),
                values: TemplateValue({
                    work_notes: `Approval notification
sent to ${wfa.dataPill(requester.Record.name, 'string')}
(${wfa.dataPill(requester.Record.email, 'string')})`,
                }),
            }
        )
    }
)
```

Subflow object

Create a subflow [sys_hub_flow] to run a reusbale sequence of actions and flow logic when called by a flow or API.

Subflows run when called by a flow or an API. Use subflows for on-demand automation that can be called by multiple flows.

Subflows consist of these components.

- One configuration object that defines the subflow configuration properties.
  - Zero or one inputs object that defines subflow inputs.
  - Zero or one outputs object that defines subflow outputs.
  - Zero or one flow variables object that defines flow variables.
- One Body function that defines what actions, flow logic, and subflows to run.
  - Zero or more wfa.action functions
  - Zero or more wfa.flow_logic functions

Properties

Name	Type	Description
configuration	Object	Required. An object containing the metadata configuration properties for the Fluent object or function.
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs.

Properties (continued)

Name	Type	Description
		Format: `Now.ID['String' or Number]`
name	String	Required. Display name of the subflow. The subflow name should be meaningful and descriptive to easily identify its purpose. This property is part of the Subflow configuration object.
description	String	Description of what the flow does. This property is part of the Subflow configuration object.
runAs	String	Determines under which user context the subflow actions are executed. Valid values: system, user Default: user This property is part of the Subflow configuration object.
runWithRoles	Array	Specify the roles that the subflow uses while running. This property is part of the Subflow configuration object.
flowPriority	String	Defines the execution priority of the subflow in the Flow Engine. Valid values: LOW, MEDIUM, HIGH Default: Medium This property is part of the Subflow configuration object.
protection	String	Defines whether the subflow is read-only (read) or editable (empty string). Valid values: read, `''` This property is part of the Subflow configuration object.
access	String	Defines the visibility/accessibility of the subflow to other subflow authors. Valid values: public, package_private Default: public This property is part of the Subflow configuration object.
category	String	Category for organizing the subflow. This property is part of the Subflow configuration object.
inputs	Object	Object defining input parameters using Column types. This property is part of the Subflow configuration object.

Properties (continued)

Name	Type	Description
outputs	Object	Object defining output parameters using Column types. This property is part of the Subflow configuration object.
flowVariables	Object	Object defining data variables available to the subflow using Column types. This property is part of the Subflow configuration object.
Flow body	Function	*`Flow body`* is an *`Arrow function`* in TypeScript that represents the execution steps in the flow. The Flow Body receives the *`_params`* parameter as its input, which contains the *`inputs`* and *`flowVariables`* objects.The steps in the *`Flow body`* consist of these function types: • *`wfa.action`* function: Execute specific tasks and operations within a flow, such as creating records, sending emails, requesting approvals, or integrating with external systems. For more information about the *`wfa.action`* function, see wfa.action function. • *`wfa.flow_logic`* function: Control how flows execute, providing conditional branching, iteration, and flow structuring capabilities. For more information about *`wfa.flow_logic`* function, see wfa.flow_logic function. You can use these parameter values in the Flow body function. **_params.trigger.current** Data type: Record. Parameter that stores the current record that started a create record, update record, or create or update record flow. **_params.trigger.previous** Data type: Record. Parameter that stores the previous record values that started an update record flow. **_params.trigger.table_name** Data type: String. Parameter that stores the table name of the current record that started the flow. **_params.trigger.flowVariables** Data type: Object. Parameter that stores the flow variables defined in this flow.

This example uses a new user's name and office location to look up a User record, send a welcome notification, and look up an unassigned hardware asset and an available office location. If there is an available laptop asset and office location, then each are assigned to the new user.

```
import { Subflow, action, wfa } from '@servicenow/sdk/automation'
import { BooleanColumn, ReferenceColumn, StringColumn }
 from '@servicenow/sdk/core'

export const newUserOnboardingSubflow = Subflow(
    {
```

```
        $id: Now.ID['new_user_onboarding_subflow'],
        name: 'New User Onboarding Subflow',
        description: 'Sends welcome notification, assigns laptop and desk, returns
assignment results',
        inputs: {
            user_sys_id: ReferenceColumn({
                label: 'User',
                referenceTable: 'sys_user',
                mandatory: true,
            }),
            office_location: ReferenceColumn({
                label: 'Office Location',
                referenceTable: 'cmn_location',
                mandatory: true,
            }),
        },
        outputs: {
            laptop_assigned: BooleanColumn({ label: 'Laptop
Assigned' }),
            desk_assigned: BooleanColumn({ label: 'Desk
Assigned' }),
            laptop_number: StringColumn({ label: 'Laptop Asset
Number', maxLength: 40 }),
            desk_number: StringColumn({ label: 'Desk Asset Number',
 maxLength: 40 }),
        },
        flowVariables: {
            laptop_found: BooleanColumn({ label: 'Laptop Found
Flag', default: false }),
            desk_found: BooleanColumn({ label: 'Desk Found
Flag', default: false }),
        },
    },
    (params) => {
        const user = wfa.action(
            action.core.lookUpRecord,
            { $id: Now.ID['lookup_user'] },
            {
                table: 'sys_user',
                conditions:
 `sys_id=${wfa.dataPill(params.inputs.user_sys_id, 'reference')}`,
                sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
            }
        )

        wfa.action(
            action.core.sendNotification,
            { $id: Now.ID['send_welcome_notification'] },
            {
                table_name: 'sys_user',
                record:
 wfa.dataPill(params.inputs.user_sys_id, 'reference'),
                notification: 'new_user_welcome_notification',
            }
        )
```

```
        const availableLaptop = wfa.action(
            action.core.lookUpRecord,
            { $id: Now.ID['lookup_available_laptop'] },
            {
                table: 'alm_hardware',

 conditions: 'assigned_to=NULL^install_status=1^substatus=available',
                sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
            }
        )

        wfa.flowLogic.if(
            {
                $id: Now.ID['check_laptop_available'],
                condition:
 `${wfa.dataPill(availableLaptop.Record.sys_id, 'reference')}!
=NULL`,
                annotation: 'Available laptop found',
            },
            () => {
                wfa.action(
                    action.core.updateRecord,
                    { $id: Now.ID['assign_laptop'] },
                    {
                        table_name: 'alm_hardware',
                        record:
 wfa.dataPill(availableLaptop.Record.sys_id, 'reference'),
                        values: TemplateValue({
                            assigned_to:
 wfa.dataPill(params.inputs.user_sys_id, 'reference'),
                            install_status: '2',
                            substatus: 'in_use',
                        }),
                    }
                )
            }
        )

        wfa.flowLogic.else(
            { $id: Now.ID['no_laptop_available'] },
            () => {
                wfa.action(
                    action.core.log,
                    { $id: Now.ID['log_no_laptop'] },
                    {
                        log_level: 'warn',
                        log_message: `No available laptop
 found for ${wfa.dataPill(user.Record.name, 'string')} — manual
 assignment required.`,
                    }
                )
            }
        )
```

```
        const availableDesk = wfa.action(
            action.core.lookUpRecord,
            { $id: Now.ID['lookup_available_desk'] },
            {
                table: 'alm_asset',
                conditions:
 `assigned_to=NULL^location=${wfa.dataPill(params.inputs.office_
location, 'reference')}^install_status=1^substatus=available`,
                sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
            }
        )

        wfa.flowLogic.if(
            {
                $id: Now.ID['check_desk_available'],
                condition:
 `${wfa.dataPill(availableDesk.Record.sys_id, 'reference')}!=NULL`,
                annotation: 'Available desk found at office location',
            },
            () => {
                wfa.action(
                    action.core.updateRecord,
                    { $id: Now.ID['assign_desk'] },
                    {
                        table_name: 'alm_asset',
                        record:
 wfa.dataPill(availableDesk.Record.sys_id, 'reference'),
                        values: TemplateValue({
                            assigned_to:
 wfa.dataPill(params.inputs.user_sys_id, 'reference'),
                            install_status: '2',
                            substatus: 'in_use',
                        }),
                    }
                )
            }
        )

        wfa.flowLogic.else(
            { $id: Now.ID['no_desk_available'] },
            () => {
                wfa.action(
                    action.core.log,
                    { $id: Now.ID['log_no_desk'] },
                    {
                        log_level: 'warn',
                        log_message: `No available desk at
 office location for ${wfa.dataPill(user.Record.name, 'string')} —
 manual assignment required.`,
                    }
                )
            }
        )

        wfa.action(
```

```
            action.core.sendNotification,
            { $id: Now.ID['send_onboarding_complete'] },
            {
                table_name: 'sys_user',
                record:
wfa.dataPill(params.inputs.user_sys_id, 'reference'),
                notification: 'user_onboarding_complete_notification',
            }
        )

        wfa.flowLogic.assignSubflowOutputs(
            {
                $id: Now.ID['assign_outputs'],
                annotation: 'Return laptop and desk assignment results',
            },
            params.outputs,
            {
                laptop_assigned: true,
                desk_assigned: true,
                laptop_number:
wfa.dataPill(availableLaptop.Record.asset_tag, 'string'),
                desk_number:
wfa.dataPill(availableDesk.Record.asset_tag, 'string'),
            }
        )
    }
)
```

wfa.trigger function

Run a flow when the start conditions of a specific trigger type are met. Triggers determine when a flow runs and what data is available from the flow start conditions.

Add a *wfa.trigger* function to the *Flow Body* function of a *Flow* object.

The following types of action instances are supported:

- *trigger.record.created*
- *trigger.record.updated*
- *trigger.record.createdOrUpdated*
- *trigger.scheduled.daily*
- *trigger.scheduled.weekly*
- *trigger.scheduled.monthly*
- *trigger.scheduled.repeat*
- *trigger.scheduled.runOnce*
- *trigger.application.inboundEmail*
- *trigger.application.slaTask*
- *trigger.application.knowledgeManagement*
- *trigger.application.remoteTableQuery*

.

For more information about available actions, see Workflow Studio flow trigger types.

Properties

Name	Type	Description
trigger	String	Name of the specific trigger to run. All action names use a dot notation that starts with `trigger`. For example, `trigger.record.created`, `trigger.scheduled.daily`, or `trigger.application.serviceCatalog`.
configuration	Object	Required. An object containing the metadata configuration properties for the Fluent object or function.
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
inputs	Object	An object containing any input parameters required by the trigger such as `table` or `condition`.

This example runs a flow when a high priority incident record is created.

```
// Trigger definition - defines when the flow starts
wfa.trigger(
    trigger.record.created,  // TriggerDefinition - record creation event
    {
        $id: Now.ID['incident_trigger'],  // string | guid, mandatory - unique identifier
        annotation: 'When high priority incident is created'  // string, optional -
describes trigger purpose
    },
    {
        table: 'incident',  // string, mandatory - table name to monitor
        condition: 'priority=1^ORpriority=2',  // string, optional - encoded query filter
        run_on_extended: 'false',  // string, optional, default: 'false' - run on child
tables
        run_flow_in: 'any',  // string, optional, default: 'any' - 'any', 'background', or
'foreground'
        run_when_setting: 'both',  // string, optional, default: 'both' - 'both',
'non_interactive', or 'interactive'
        run_when_user_list: [],  // Array, optional, default: [] - need to use
runQuery to get user sys_ids from sys_user table
        run_when_user_setting: 'any'  // string, optional, default: 'any' - 'any',
'one_of', or 'not_one_of'
    }
)
```

This examples starts a flow daily at 9 AM.

```
// # CORRECT - Daily trigger at specific time
wfa.trigger(
    trigger.scheduled.daily,
    {
        $id: Now.ID['daily_report_trigger'],
        annotation: 'Runs daily at 9 AM'
    },
    {
```

```
        time: Time({ hours: 9, minutes: 0, seconds: 0 })
    }
)
```

This examples starts a flow when there is an email reply to an incident record.

```
// Trigger when email arrives for incident table
wfa.trigger(
    trigger.application.inboundEmail, // TriggerDefinition - inbound email
event
    {
        $id: Now.ID['incident_email_trigger'], // string | guid, mandatory
        annotation: 'Process emails for incident table' // string, optional
    },
    {
        target_table: 'incident', // string, optional - table for email replies/forwards
        email_conditions: '', // string, optional - filter which emails trigger the flow
        order: 100, // number, optional - execution order when multiple email triggers
exist
        stop_condition_evaluation: true // boolean, optional - stop
evaluating other triggers if this one matches
    }
)

// Access email data in flow body:
(_params) => {
    const emailSubject = _params.trigger.email.subject // Email subject
    const emailBody = _params.trigger.email.body // Email body
    const emailFrom = _params.trigger.email.from // Sender email address
}
```

wfa.action function

Run a specific action instance from a flow or subflow. Actions determine what data is generated, updated, or retrieved.

Add *wfa.action* functions to the *Flow Body* function of a *Flow* or *Subflow* object.

The following types of action instances are supported:

- *action.core.createRecord*
- *action.core.createOrUpdateRecord*
- *action.core.deleteRecord*
- *action.core.lookUpRecord*
- *action.core.lookUpRecords*
- *action.core.updateRecord*
- *action.core.updateMultipleRecords*
- *action.core.sendNotification*
- *action.core.sendEmail*
- *action.core.sendSms*
- *action.core.askForApproval*
- *action.core.waitForApproval*
- *action.core.createTask*

- `action.core.log`
- `action.core.waitForCondition`
- `action.core.waitForMessage`
- `action.core.slaPercentageTimer`

.

For more information about available actions, see Workflow Studio actions.

Properties

Name	Type	Description
action	String	Name of the specific action to run. All action names use a dot notation that starts with `action.core`. For example, `action.core.lookUpRecord`, `action.core.lookUpRecords`, `action.core.createRecord`, or `action.core.updateRecord`.
configuration	Object	Required. An object containing the metadata configuration properties for the Fluent object or function.
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
inputs	Object	An object containing any input parameters required by the action such as `table_name` or `values`.

This example runs when a change request is approved and notifies the requester. The flow uses the Look Up Records, Send Email, and Update Record actions.

```
import { action, Flow, wfa, trigger } from '@servicenow/sdk/automation'

export const changeRequestApprovalNotificationFlow = Flow(
    {
        $id: Now.ID['change_request_approval_notification_flow'],
        name: 'Change Request Approval Notification Flow',
        description: 'Sends formatted notification to requester when change
request is approved',
    },
    wfa.trigger(
        trigger.record.updated,
        { $id: Now.ID['change_request_approved_trigger'] },
        {
            table: 'change_request',
            condition: 'approval=approved',
            run_flow_in: 'background',
            trigger_strategy: 'unique_changes',
            run_when_user_list: [],
            run_when_setting: 'both',
            run_on_extended: 'false',
            run_when_user_setting: 'any',
```

```
        }
    ),
    (params) => {
        const requester = wfa.action(
            action.core.lookUpRecord,
            { $id: Now.ID['lookup_requester_details'] },
            {
                table: 'sys_user',
                conditions:
 `sys_id=${wfa.dataPill(params.trigger.current.requested_by, 'refer
ence')}`,
                sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
            }
        )

        wfa.action(
            action.core.sendEmail,
            { $id: Now.ID['send_approval_notification_email'] },
            {
                table_name: 'change_request',
                watermark_email: true,
                ah_subject: `Change Request
 ${wfa.dataPill(params.trigger.current.number, 'string')} -
 Approved`,
                ah_body: `Your change request has been
 approved.`,
                record:
 wfa.dataPill(params.trigger.current.sys_id, 'reference'),
                ah_to:
 wfa.dataPill(requester.Record.email, 'string'),
            }
        )

        wfa.action(
            action.core.updateRecord,
            { $id: Now.ID['update_work_notes_notification_sent'] },
            {
                table_name: 'change_request',
                record:
 wfa.dataPill(params.trigger.current.sys_id, 'reference'),
                values: TemplateValue({
                    work_notes: `Approval notification
 sent to ${wfa.dataPill(requester.Record.name, 'string')}
 (${wfa.dataPill(requester.Record.email, 'string')})`,
                }),
            }
        )
    }
)
```

This example runs a flow when a change request record is created or updated as high impact. The flow runs the Update Record to mark the change request as high risk, the Look Up Record to find the group with the name CAB, the Send Notification to send a high risk change notification to the CAB group, and the Log action to document both the record update and the notification sent.

```
import { action, Flow, wfa, trigger } from '@servicenow/sdk/automation'

export const changeRiskTaggingFlow = Flow(
    {
        $id: Now.ID['change_risk_tagging_flow'],
        name: 'Change Risk Tagging Flow',
        description: 'Tags change requests with high-risk label when created or
updated with high impact',
    },
    wfa.trigger(
        trigger.record.createdOrUpdated,
        { $id: Now.ID['change_risk_trigger'] },
        {
            table: 'change_request',
            condition: 'active=true^impact=1',
            run_flow_in: 'background',
            run_on_extended: 'false',
            run_when_setting: 'both',
            run_when_user_setting: 'any',
            run_when_user_list: [],
        }
    ),
    (params) => {
        wfa.action(
            action.core.updateRecord,
            { $id: Now.ID['tag_high_risk'] },
            {
                table_name: 'change_request',
                record:
wfa.dataPill(params.trigger.current.sys_id, 'reference'),
                values: TemplateValue({
                    risk: 'high',
                    work_notes: 'Automatically tagged as high-risk due to high
impact.',
                }),
            }
        )

        const cab = wfa.action(
            action.core.lookUpRecord,
            { $id: Now.ID['lookup_cab_group'] },
            {
                table: 'sys_user_group',
                conditions: 'name=CAB^active=true',
                sort_type: 'sort_asc',

if_multiple_records_are_found_action: 'use_first_record',
            }
        )

        wfa.action(
            action.core.sendNotification,
            { $id: Now.ID['notify_cab_high_risk'] },
            {
                table_name: 'change_request',
```

```
                record:
wfa.dataPill(params.trigger.current.sys_id, 'reference'),
                notification: 'high_risk_change_cab_notification',
            }
        )

        wfa.action(
            action.core.log,
            { $id: Now.ID['log_risk_tagged'] },
            {
                log_level: 'warn',
                log_message: `Change
${wfa.dataPill(params.trigger.current.number, 'string')} tagged
as high-risk. CAB group ${wfa.dataPill(cab.Record.name, 'string')}
notified.`,
            }
        )
    }
)
```

wfa.flow_logic function

Run a specific flow logic instance from a flow or subflow. Flow logic determines how and when data is used.

Add `wfa.flow_logic` functions to the `Flow Body` function of a `Flow` or `Subflow` object.

The following types of flow logic instances are supported:

- `if`
- `elseIf`
- `else`
- `forEach`
- `waitForADuration`
- `exitLoop`, `endFlow`, `skipIteration`, `setFlowVariables`, and `assignSubflowOutputs`

.

For more information about available flow logic, see Workflow Studio flow logic.

Properties

Name	Type	Description
configuration	Object	Required. An object containing the metadata configuration properties for the Fluent object or function.
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
annotation	String	Describe what this instance of flow logic does. For example, `annotation: 'Priority is critical'`.

Properties (continued)

Name	Type	Description
condition	String	Required in a *wfa.flow_logic.if* and *wfa.flow_logic.elseIf* function. Data values that must be true to run this flow logic. For example, `condition: `\${_params.trigger.current.priority}=1`,`.
item	Array	Required in a *wfa.flow_logic.forEach* function. An array or a collection of records to iterate over. Typically, this list of records is dynamically generated from a *wfa.action* function call to the *action.core.lookUpRecords* action. For example, a `problems.Records` array can be generated from an *action.core.lookUpRecords* action. `(_params) => { //Find all the newly created problem records for the past day const problems = wfa.action(action.core.lookUpRecords, { $id: Now.ID['daily_unassigned_new_problems_triage_flow_step1'], annotation: 'Find all the newly created problem records for the past day', }, { table: 'problem', conditions: 'sys_created_onONYesterday@javascript:gs.beginningOfYesterday()@javascript:gs.endOfYesterday()', }) //Iterate over the problems wfa.flow_logic.forEach(problems.Records, { ... }`
label	String	Display value describing this instance of flow logic. For example, `label: 'If Priority is critical',`.
variables	Schema	Required in a *wfa.flow_logic.assignSubflowOutputs* and *wfa.flow_logic.setFlowVariables* function. A schema produced by the FlowVariables API that defines the data structure of the flow variables. For example, the `flowVars` schema can be generated from FlowVariables. `import { FlowVariables } from '@servicenow/sdk/automation' // Define flow variables const flowVars = FlowVariables({`

Properties (continued)

Name	Type	Description
		<pre> status: StringColumn({ label: 'Status' }), count: IntegerColumn({ label: 'Count' }), isActive: BooleanColumn({ label: 'Is Active' }), }) // Flow body (_params) => { // Set flow variables with literal values wfa.flow_logic.setFlowVariables({ $id: Now.ID['initialize_variables'], annotation: 'Initialize flow variables with default values', }, _params.flowVariables, { ... }</pre> **Important:** Always use `_params.flowVariables` as the argument for this property. Passing the wrong schema will cause validation errors.
Flow logic body	Function	*Flow logic body* is an *Arrow function* in TypeScript that represents the execution steps in the flow logic. The Flow Body receives the *_params* parameter as its input, which contains the *wfa.trigger* and *flowVariables* objects. The steps in the *Flow logic body* consist of these functions calls: • *wfa.action* function: Run a specific action instance from a flow or subflow. Actions determine what data is generated, updated, or retrieved.. For more information about the *wfa.action* function, see wfa.action function. • *wfa.flow_logic* function: Run a specific flow logic instance from a flow or subflow. Flow logic determines how and when data is used. For more information about the wfa.flow_logic function, see wfa.flow_logic function.

This example shows checking for conditions in the trigger record. If the current record has a priority of 1, it assigns the record to a critical priority team. Else If the current record has a priority value of 2, it assigns the record to a high priority team. If neither condition is met, the record is assigned to a general team.

```
// Flow body showing if/elseIf/else usage
(_params) => {
    wfa.flow_logic.if(
        {
            $id: Now.ID['check_critical_priority'],
            condition: `\${_params.trigger.current.priority}=1`,
            annotation: 'Priority is critical',
```

```
        },
        () => {
            wfa.action(
                action.core.updateRecord,
                { $id: Now.ID['assign_critical_team'],
 annotation: 'Assign to critical team' },
                { record: _params.trigger.current,
 table_name: 'incident', values: 'assignment_group=critical_team' }
            )
        }
    )
    wfa.flow_logic.elseIf(
        { $id: Now.ID['check_high_priority'], condition:
 `\${_params.trigger.current.priority}=2`, annotation: 'Priority is
high' },
        () => {
            wfa.action(
                action.core.updateRecord,
                { $id: Now.ID['assign_high_team'], annotation: 'Assign
to high team' },
                { record: _params.trigger.current,
 table_name: 'incident', values: 'assignment_group=high_team' }
            )
        }
    )
    wfa.flow_logic.else(
        { $id: Now.ID['assign_default'], annotation: 'Default assignment' },
        () => {
            wfa.action(
                action.core.updateRecord,
                { $id: Now.ID['assign_general_team'],
 annotation: 'Assign to general team' },
                { record: _params.trigger.current,
 table_name: 'incident', values: 'assignment_group=general_team' }
            )
        }
    )
}
```

This example iterates through a list of problem records to find unassigned records, assign them to a problem triage group, and send an email to the group.

```
// Flow body showing forEach usage with lookUpRecords
(_params) => {
    //Find all the newly created problem records for the past day
    const problems = wfa.action(
        action.core.lookUpRecords,
        {
            $id: Now.ID['daily_unassigned_new_problems_triage_flow_step1'],
            annotation: 'Find all the newly created problem records for the past
day',
        },
        {
            table: 'problem',
            conditions:
```

```
  'sys_created_onONYesterday@javascript:gs.beginningOfYesterday()@javascript:gs.endOf
Yesterday()',
        }
    )

    //Iterate over the problems
    wfa.flow_logic.forEach(
        problems.Records,
        { $id: Now.ID['daily_unassigned_new_problems_triage_flow_step2'],
 annotation: 'Iterate each problem' },
        (item) => {
            //Check if the problem is not assigned
            wfa.flow_logic.if(
                {
                    $id:
 Now.ID['daily_unassigned_new_problems_triage_flow_step3'],
                    condition:
 `\${item.assignment_group}ISEMPTY`,
                    annotation: 'Check if problem is not assigned',
                },
                () => {
                    wfa.action(
                        action.core.updateRecord,
                        {
                            $id:
 Now.ID['daily_unassigned_new_problems_triage_flow_step4'],
                            annotation: 'Update the problem record state',
                        },
                        { table_name: 'problem', record: item,
 values: 'state=1' }
                    )

                    //Send notification to the group
                    wfa.action(
                        action.core.sendNotification,
                        {
                            $id:
 Now.ID['daily_unassigned_new_problems_triage_flow_step5'],
                            annotation: 'Send notification to the group',
                        },
                        {
                            table_name: 'problem',
                            record: item,
                            notification: '',
                        }
                    )
                }
            )
        }
    )
}
```

This example sets flow variables to dynamic values stored in data pills.

```
import { FlowVariables } from '@servicenow/sdk/automation'
```

```
const flowVars = FlowVariables({
    incidentPriority: IntegerColumn({ label: 'Incident Priority' }),
    assignedGroup: StringColumn({ label: 'Assigned Group' }),
    threshold: IntegerColumn({ label: 'Threshold' }),
    actionResult: StringColumn({ label: 'Action Result' }),
})

(_params) => {
    // Lookup action result
    const lookupResult = wfa.action(
        action.core.lookUpRecord,
        { $id: Now.ID['lookup_config'], annotation: 'Get configuration' },
        { table: 'sys_properties', conditions: 'name=incident.threshold' }
    )

    // Set flow variables using datapills from trigger, action outputs, and other flow variables
    wfa.flow_logic.setFlowVariables(
        {
            $id: Now.ID['set_variables_with_datapills'],
            annotation: 'Set variables from trigger and action outputs',
        },
        _params.flowVariables,
        {
            incidentPriority: _params.trigger.current.priority,
       // Trigger datapill
            assignedGroup:
 _params.trigger.current.assignment_group,    // Trigger datapill
            threshold: lookupResult.Record.value,
       // Action output datapill
            actionResult: _params.flowVariables.status,
       // Flow variable datapill
        }
    )
}
```

This exampe assign subflow Outputs using flow variables.

```
// Flow body showing assignSubflowOutputs usage with flow variables
import { FlowVariables } from '@servicenow/sdk/automation'

const flowVars = FlowVariables({
    userEmail: StringColumn({ label: 'User Email' }),
    userName: StringColumn({ label: 'User Name' }),
    isApproved: BooleanColumn({ label: 'Is Approved' }),
})

(_params) => {
    // Call a subflow that returns user information
    const subflowResult = wfa.subflow(
        userInfoSubflow,
        { $id: Now.ID['get_user_info'], annotation: 'Get user
information' },
        { userId: _params.trigger.current.caller_id }
    )

    // Assign subflow outputs to flow variables in one operation
    wfa.flow_logic.assignSubflowOutputs(
```

```
        { $id: Now.ID['assign_outputs'], annotation: 'Assign user info to flow
variables' },
        flowVars,
        {
            userEmail: subflowResult.email,
            userName: subflowResult.name,
            isApproved: subflowResult.approved,
        }
    )

    // Now flow variables can be used throughout the flow
    // Access via _params.flowVariables.userEmail, etc.
}
```

wfa.dataPill function

Reference a specific runtime data pill value from an action or flow logic input.

Add `wfa.dataPill` functions to the input parameters of a `action` or `flow_logic` function. Define a constant for each data pill you want to use. Typically, this constant stores the output of a wfa.action or wfa.flow_logic function.

Properties

Name	Type	Description
expression	String	Required. Dot notation reference to the runtime data that you want use as an input value. Commonly used runtime values include trigger data, action outputs, and subflow outputs. **Trigger record data** Field values from a trigger record. For example, `_params.trigger.current.fieldName`. **Trigger record table** Table name of the trigger record. For example, `_params.trigger.table_name`. **Action output values** Output values from previously run actions. For example, `actionResult.fieldName` **Subflow output values** Output values from previously run subflows. For example, `subflowResult.fieldName`
Type	String	Required. Data type of the data such as string, boolean, integer, or reference. For a list of available field data types, see Field types.

This example uses an unbound email record to create either an incident or a task based on the values of the inbound email. This example uses the params object to refer to data in the trigger. This example also defines the constants sender, senderManager, p3Incident, and attachments to store the output of actions and flow logic for use in later wfa.dataPill functions.

```
import { action, Flow, wfa, trigger } from '@servicenow/sdk/automation'

export const emailIncidentTaskFlow = Flow(
    {
```

```
        $id: Now.ID['email_incident_task_flow'],
        name: 'Email Incident/Task Creation Flow',
        description: 'Creates incidents or tasks based on email content with "raise
incident or task" in subject',
    },
    wfa.trigger(
        trigger.application.inboundEmail,
        { $id: Now.ID['inbound_email_trigger'] },
        {
            email_conditions: 'subjectLIKEraise incident or task',
            target_table: 'incident',
        }
    ),
    (params) => {
        wfa.action(
            action.core.log,
            { $id: Now.ID['log_email_received'] },
            {
                log_level: 'info',
                log_message: `Email received from:
 ${wfa.dataPill(params.trigger.from_address, 'string')}, Subject:
 ${wfa.dataPill(params.trigger.subject, 'string')}`,
            }
        )

        const sender = wfa.action(
            action.core.lookUpRecord,
            { $id: Now.ID['lookup_sender'] },
            {
                table: 'sys_user',
                conditions:
 `email=${wfa.dataPill(params.trigger.from_address, 'string')}`,
                sort_type: 'sort_asc',

 if_multiple_records_are_found_action: 'use_first_record',
            }
        )

        // Internal P1 path — two tasks
        wfa.flowLogic.if(
            {
                $id: Now.ID['check_internal_p1'],
                condition:
 `
${wfa.dataPill(sender.Record.email, 'string')}LIKEservicenow^${wfa.d
ataPill(params.trigger.subject, 'string')}LIKEP1`,
                annotation: 'Internal sender with P1 subject',
            },
            () => {
                const senderManager = wfa.action(
                    action.core.lookUpRecord,
                    { $id: Now.ID['lookup_sender_manager'] },
                    {
                        table: 'sys_user',
                        conditions:
 `sys_id=${wfa.dataPill(sender.Record.manager, 'reference')}`,
                        sort_type: 'sort_asc',
```

```
if_multiple_records_are_found_action: 'use_first_record',
                    }
                )

                wfa.action(
                    action.core.createTask,
                    { $id: Now.ID['task_for_manager'] },
                    {
                        task_table: 'incident',
                        field_values: TemplateValue({
                            priority: 1,
                            assigned_to:
wfa.dataPill(senderManager.Record.sys_id, 'reference'),
                            short_description:
wfa.dataPill(params.trigger.inbound_email.body, 'reference'),
                            urgency: 1,
                            impact: 1,
                        }),
                    }
                )

                wfa.action(
                    action.core.createTask,
                    { $id: Now.ID['task_for_sender'] },
                    {
                        task_table: 'incident',
                        field_values: TemplateValue({
                            priority: 1,
                            assigned_to:
wfa.dataPill(sender.Record.sys_id, 'reference'),
                            short_description:
wfa.dataPill(params.trigger.subject, 'string'),
                            urgency: 1,
                            impact: 1,
                        }),
                    }
                )
            }
        )

        // External / non-P1 path — P3 incident + copy attachments
        wfa.flowLogic.else({ $id: Now.ID['create_p3_incident_branch'] },
() => {
            const p3Incident = wfa.action(
                action.core.createRecord,
                { $id: Now.ID['create_p3_incident'] },
                {
                    table_name: 'incident',
                    values: TemplateValue({
                        priority: 4,
                        short_description:
wfa.dataPill(params.trigger.subject, 'string'),
                        description: `From:
${wfa.dataPill(params.trigger.from_address, 'string')}\n\n
${wfa.dataPill(params.trigger.inbound_email.body, 'reference')}`,
                        contact_type: 'email',
```

```
                        caller_id:
wfa.dataPill(sender.Record.sys_id, 'reference'),
                    }),
                }
            )

            const attachments = wfa.action(
                action.core.getAttachmentsOnRecord,
                { $id: Now.ID['get_email_attachments'] },
                { source_record:
wfa.dataPill(params.trigger.inbound_email.sys_id, 'reference') }
            )

            wfa.flowLogic.forEach(
                wfa.dataPill(attachments.parameter, 'records'),
                { $id: Now.ID['copy_attachments_loop'] },
                () => {
                    wfa.action(
                        action.core.copyAttachment,
                        { $id: Now.ID['copy_attachment'] },
                        {
                            target_record:
wfa.dataPill(p3Incident.record, 'reference'),
                            attachment_record:
wfa.dataPill(attachments.parameter, 'records'),
                            table: 'incident',
                        }
                    )
                }
            )

            wfa.action(
                action.core.sendEmail,
                { $id: Now.ID['confirm_p3_email'] },
                {
                    table_name: 'incident',
                    ah_to:
wfa.dataPill(params.trigger.from_address, 'string'),
                    ah_subject: `Incident Created:
${wfa.dataPill(params.trigger.subject, 'string')}`,
                    ah_body: `Your request has been received
and a P3 incident has been created. Our team will be in touch
shortly.`,
                    record:
wfa.dataPill(p3Incident.record, 'reference'),
                }
            )
        })
    }
)
```

Import Sets API - ServiceNow Fluent

The Import Sets API defines transform maps [sys_transform_map] that specify how to transform and map data from the import set staging table to target tables.

Every import operation to a production table requires at least one transform map associated with an import set. The transform map specifies the data relationships between the import set and the target table.

To create an import set in ServiceNow Fluent code, you must define the required metadata in the following order:

1. Define the staging table [sys_db_object] using the Table API. The table must extend the Import Set Row [sys_import_set_row] table. For more information, see Table API - ServiceNow Fluent.

   The staging table defines all columns that receive imported data.

2. Define the data source [sys_data_source] using the Record API. The data source must reference the staging table from its *`import_set_table_name`* property. For more information, see Record API - ServiceNow Fluent.

   The data source defines the connection to external systems (files, databases, APIs) and how to load data into import staging tables. For more information, see Data sources.

3. Define the transform map [sys_transform_map] using the Import Sets API. The transform map must reference the staging table from its *`sourceTable`* property.

Important:
The string `NULL` is a reserved word. It shouldn't be used as a field value in import set transform maps or anywhere in the **First name** or **Last name** fields. The reserved word is `NULL` in all capital letters. A field with the value `Null` or `null`, for example, is acceptable. `NULL` should be used only to clear out a particular field.

For general information about import sets, see Import sets.

Related topics

ServiceNow Fluent

ImportSet object

Create a transform map [sys_transform_map] to define the relationships between fields in an import set table and fields in an existing table.

For general information about creating transform maps, see Create a transform map.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. An internal name for the transform map.
targetTable	String	Required. The name of the table in which you want the transformed data to be inserted. You can specify only tables within the application scope or the global scope, or tables that grant write access to other applications.

Properties (continued)

Name	Type	Description
sourceTable	String	Required. The name of the table containing the raw import set data to transform. An import table is any table that extends the Import Set Row [sys_import_set_row] table. You can specify only tables within the application scope. **Note:** The value of this property must match the value of the *name* property in the staging table definition and the *import_set_table_name* property in the data source definition.
order	Number	The execution order in which to apply transform maps if more than one map fits the conditions. Default: 100
active	Boolean	Flag that indicates whether the transform map is active. Default: false
runBusinessRules	Boolean	Flag that indicates whether to run business rules, workflows, approval engines, auditing, and field normalization while the transformation inserts or updates data into the target table. If false, *GlideRecord.setWorkflow()* runs with a value of false. Default: true
enforceMandatoryFields	String	An option to enforce mandatory fields on the target table. Valid values: • no: Don't enforce mandatory fields. • onlyMappedFields: Enforce mandatory mapped fields only. • allFields: Enforce all mandatory fields. Default: no
copyEmptyFields	Boolean	Flag that indicates whether to copy empty fields from source and override existing target field values. Default: false
createOnEmptyCoalesce	Boolean	Flag that indicates whether to create a record when coalesce fields are empty, instead of ignoring the record or overwriting an existing record. If *createOnEmptyCoalesce* is true for any field in the record, the record is coalesced. Default: false

Properties (continued)

Name	Type	Description
runScript	Boolean	Flag that indicates whether to execute the transform map script. The system runs the transform map script in addition to any field maps. Default: false
script	Script	A script that transforms field values in the source table to the target table. This property supports a function from a JavaScript module, a reference to another file in the application that contains a script, or inline JavaScript. Expects a function of type `(source, target, map, log, isUpdate) => void`. Format: • For functions, use the name of a function, function expression, or default function exported from a JavaScript module and import it into the `.now.ts` file. For information about JavaScript modules, see JavaScript modules and third-party libraries. • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``.
fields	Object	Key and value pairs of field mappings where the key is target field name and the value is the source field name or the fields object. Each target field name (key) must be unique within the fields object to avoid mapping conflicts. Format: • To map a target field name to a source field object, see fields object. • To map a target field name to a source field name, use the following format: `fields: { targetFieldName: 'sourceFieldName', ... }`
scripts	Array	A list of transform scripts for additional processing. For more information, see scripts array.

```
import '@servicenow/sdk/global'
import { Table, Record, ImportSet } from '@servicenow/sdk/core'

// STEP 1: Create Staging Table Definition (REQUIRED - MUST BE FIRST)
// This creates the actual table structure to hold imported data
export const userStagingTable = Table({
```

```
    $id: Now.ID['user-staging-table'],
    name: 'u_user_import_staging',
    label: 'User Import Staging',
    extends: 'sys_import_set_row',  // All staging tables extend this
    columns: [
        {
            name: 'u_email_address',
            type: 'email',
            max_length: 100,
            label: 'Email Address'
        },
        {
            name: 'u_full_name',
            type: 'string',
            max_length: 100,
            label: 'Full Name'
        },
        {
            name: 'u_username',
            type: 'string',
            max_length: 40,
            label: 'Username'
        },
    ]
})

// STEP 2: Create Data Source (REQUIRED - MUST BE SECOND)
// The data source defines HOW to get data from external systems
export const userDataSource = Record({
    $id: Now.ID['user-csv-datasource'],
    table: 'sys_data_source',
    data: {
        name: 'User CSV Data Source',
        type: 'File',
        format: 'CSV',
        file_retrieval_method: 'Attachment',
        csv_delimiter: ',',
        header_row: 1,
        // CRITICAL: This must match the table name from STEP 1
        import_set_table_name: 'u_user_import_staging',
        import_set_table_label: 'User Import Staging',
        batch_size: 500,
        active: true,
    },
})

// STEP 3: Create Import Set (Transform Map) (REQUIRED - MUST BE THIRD)
// The import set defines HOW to transform data from staging to target table
export const userImportSet = ImportSet({
    $id: Now.ID['user-import-transform'],
    name: 'User Import Transform',
    targetTable: 'sys_user',
    // CRITICAL: This must match import_set_table_name in Data Source
    sourceTable: 'u_user_import_staging',
    active: true,
    runBusinessRules: true,
    fields: {
```

```
        email: {
            sourceField: 'u_email_address',
            coalesce: true,
        },
        name: 'u_full_name',
        user_name: 'u_username',
    }
})
```

fields object

Define field mappings [sys_transform_entry] from the source fields of an import set to the fields of the target table.

Use the `fields` object within the `ImportSet` object. In the `fields` object, each key for the target field name must be unique to avoid mapping conflicts.

For general information about creating field maps, see Create a field map.

Properties

Name	Type	Description
sourceField	String	The name of the source field from the import table to be transformed. This property is required unless you use the `sourceScript` property or a coalesce-only configuration. Field on the source table to be transformed. If the `sourceTable` contains only raw data, the value can be empty.
choiceAction	String	The action to take if the import set contains a reference or choice value other than those available. This property applies if the target field is a choice list or reference field. Valid values: • create: Create a choice or record in the reference table. • ignore: Ignore the new value from the source table. • reject: Skip the entire row (record) containing the new value and continue to the next row.
sourceScript	Script	A script that transforms field values in the source table to the target table. This property supports a function from a JavaScript module, a reference to another file in the application that contains a script, or inline JavaScript. Expects a function of type `(source) => any`. Format: • For functions, use the name of a function, function expression, or default function exported from a JavaScript module and import it into the `.now.ts` file. For information about JavaScript modules, see JavaScript modules and third-party libraries. • To use text content from another file, refer to a file in the application using the following format:

Properties (continued)

Name	Type	Description
		`Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``.
useSourceScript	Boolean	Flag that indicates whether to use the source script instead of the source field. Default: false
dateFormat	String	The format for transformations when the target field is a Date or Date/Time field. **Note:** To learn more about allowable date formats, see Global date and time field format. Format: `'dd-MM-yyyy'`, `'yyyy-MM-dd'`, `'yyyy-dd-MM'`, `'MM-dd-yyyy HH:mm:ss z'`, `'yyyy-MM-dd HH:mm:ss'`, `'HH:mm:ss'`, `'MM-dd-yyyy HH:mm:ss'`, `'dd-MM-yyyy HH:mm:ss z'`, `'MM-dd-yyyy'`, or `'dd-MM-yyyy HH:mm:ss'`
referenceValueField	String	The reference value field when the target field is a reference field. The transform map needs a way to match incoming source values to existing records in the reference field's source table. Because most imports don't provide a 32-character sys_id value, you must specify a column from the reference field's source table that contains values that match the incoming source values. For more information, see Create a field map.
coalesce	Boolean	Flag that indicates whether the field is used for record matching. Configuring a target field to coalesce causes the import set to treat the field as a unique key. For more information, see Create a field map. Default: false
coalesceCaseSensitive	Boolean	Flag that indicates whether to make case-sensitive coalesce values result in the creation of new records. By default, coalesce fields are used in a case-insensitive lookup for existing records. Case-insensitive records update existing records only and don't cause the creation of new records. Default: false
coalesceEmptyFields	Boolean	Flag that indicates whether to match an empty source field value to an empty target field value. The *`coalesce`* property must be set to true for this property to apply.

Properties (continued)

Name	Type	Description
		For example, the User transform map coalesces on the email field. If this property is set to true, a source record containing an empty email address coalesces to a target record containing an empty email address. Default: false

```
fields: {
        email: {
            sourceField: 'email_address',
            coalesce: true,
            useSourceScript: true,
            sourceScript: `answer = (function
 transformEntry(source) {
                return source.email_address.toLowerCase().trim();
            })(source);`
        },
        department: {
            sourceField: 'dept_code',
            choiceAction: 'create'
        }
}
```

scripts array

Define transform scripts [sys_transform_script] that run at different stages of the import process.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
active	Boolean	Flag that indicates whether the script is active and executes. Default: true
order	Number	The execution order in which scripts run if more than one script fits the conditions. Default: 100
when	String	The stage of the import process in which to execute the script. For more information, see Map with transformation event scripts. Valid values: onBefore, onAfter, onReject, onStart, onForeignInsert, onComplete, onChoiceCreate

Properties (continued)

Name	Type	Description
		Default: onAfter
script	Script	A script that modifies the transformation behavior at the stage specified with the *when* property. This property supports a function from a JavaScript module, a reference to another file in the application that contains a script, or inline JavaScript. Expects a function of type `(source, map, log, target) => void`. Format: • For functions, use the name of a function, function expression, or default function exported from a JavaScript module and import it into the `.now.ts` file. For information about JavaScript modules, see JavaScript modules and third-party libraries. • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``.

```
scripts: [
        {
            $id: Now.ID['validate-email'],
            active: true,
            order: 100,
            when: 'onBefore',
            script: `(function runTransformScript(source, map,
 log, target) {
                if (!source.email_address ||
 source.email_address.indexOf('@') === -1) {
                    log.error('Invalid email address: ' +
 source.email_address);
                    return;
                }
            })(source, map, log, target);`
        },
        {
            $id: Now.ID['validate-user-script'],
            active: true,
            order: 100,
            when: 'onBefore',
            // Using imported function for lifecycle script
            script: validateUserData
        }
    ]
```

List API - ServiceNow Fluent

The List API defines list views [sys_ui_list] for tables.

For general information about lists, see List administration.

Related topics

ServiceNow Fluent

List object

Configure lists [sys_ui_list] and their views.

Properties

Name	Type	Description
table	String	Required. The name of the table to which the list applies.
view	Reference or String	Required. The variable identifier or name of the UI view [sys_ui_view] which applies, or the default view. To define a UI view, use the Record API - ServiceNow Fluent. To use the default view (`default_view`), you must import it: `import { default_view } from '@servicenow/sdk/core'`
columns	Array	Required. A list of columns in the table to display in the list, specified using the column name and position in the list.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { List } from "@servicenow/sdk/core";

List({
    $id: Now.ID["app_task_view_list"],
    table: "cmdb_ci_server",
    view: app_task_view,
    columns: [
        { element: "name", position: 0 },
```

```
        { element: "business_unit", position: 1 },
        { element: "vendor", position: 2 },
        { element: "cpu_type", position: 3 },
    ],
});
```

The UI view definition referenced is defined using the *Record* object:

```
import { Record } from "@servicenow/sdk/core";

const app_task_view = Record({
   $id: Now.ID['app_task_view'],
   table: 'sys_ui_view',
   data: {
        name: 'app_task_view',
        title: 'app_task_view'
   }
})
```

Property API - ServiceNow Fluent

The Property API defines system properties [sys_properties] that control instance behavior.

For general information about system properties, see Add a system property.

Related topics

ServiceNow Fluent

Property object

Add a system property [sys_properties] for configuring an aspect of an application.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. The name of the property beginning with the application scope in the following format: `<scope>.<name>`.
value	Any	A value for the property. The value must be the correct data type. All property values are stored as strings. When retrieving properties via the *`gs.getProperty()`* method, treat the results as strings. For example, a true|false property returns 'true' or 'false' (strings), not the Boolean equivalent.
type	String	A data type for the property value.

Properties (continued)

Name	Type	Description
		Valid values: string, integer, boolean, choicelist, color, date_format, image, password, password2, short_string, time_format, timezone, uploaded_image
description	String	A description of what the property does.
choices	Array	A comma-separated list of choice values. This property only applies if the *`type`* property is set to `choicelist`. If you need a different choice label and value, use an equal sign (=) to separate the label from the value. For example, `['Blue=0000FF', 'Red=FF0000', 'Green=00FF00']` displays **Blue**, **Red**, and **Green** in the list, and saves the corresponding hex value in the property value field.
roles	Object	The variable identifiers of *`Role`* objects or names of roles that have read or write access to the property. For example: `roles: {` `  read: [activity_admin, 'app_user'],` `  write: [admin]` `}` For more information, see Role API - ServiceNow Fluent.
ignoreCache	Boolean	Flag that indicates whether to cache flush when the value of the property is set. The system stores system property values in server-side caches to avoid querying the database for configuration settings. When you change a system property value, the system flushes the cache for the System Properties [sys_properties] table. Use this field to determine whether to flush this property's value from all other server-side caches. Valid values: • true: The system ignores flushing some server-side caches, thus flushing only the cache for the System Properties [sys_properties] table and preserving the prior property value in all other caches. This option avoids the performance cost of flushing all caches and retrieving new property values. Generally, you should only set this property to `true` when you have a system property that changes more frequently than once a month, and the property value is only stored in the System Properties [sys_properties] table table.

Properties (continued)

Name	Type	Description
		• false: The system flushes all server-side caches and retrieves the current property value from the database. Set this property to `false` for all caches to have the current property value. Default: false
isPrivate	Boolean	Flag that indicates whether to exclude the property from being imported via update sets. Keeping system properties private helps prevent settings in one instance from overwriting values in another instance. For example, you might not want a system property in a development instance to use the same value as a production instance. Valid values: • true: The property isn't included in update sets. • false: The property is included in update sets. Default: false
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { Property } from '@servicenow/sdk/core'

Property({
   $id: Now.ID['1234'],
   name: 'x_snc_app.some.new.prop',
   type: 'string',
   value: 'hello',
   description: 'A new property',
   roles: {
      read: ['admin'],
```

```
      write: [adminRole, managerRole],
   },
   ignoreCache: false,
   isPrivate: false,
})
```

The roles referenced are defined using the *Role* object:

```
import { Role } from "@servicenow/sdk/core";

const managerRole = Role({
   $id: Now.ID['manager_role'],
   name: 'x_snc_example.manager'
})

const adminRole = Role({
   $id: Now.ID['admin_role'],
   name: 'x_snc_example.admin',
   containsRoles: [managerRole]
})
```

Record API - ServiceNow Fluent

The Record API defines records in any table. Use the Record API to define application metadata that doesn't have a dedicated ServiceNow Fluent API.

Related topics

ServiceNow Fluent

Record object

Add data to any table with a record.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
table	String	Required. The name of the table to which the record belongs.
data	Object	Fields and their values in the table. For example: `data: { state: 'Ready', task: 'Add demo data' }` To use text content from another file, refer to a file in the application using the `Now.include` syntax. For more information, see ServiceNow Fluent language constructs.

Properties (continued)

Name	Type	Description
		`data: { script: Now.include('./script-file.js'), html: Now.include('./html-file.html'), css: Now.include('./css-file.css') }`
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

In this example, a record defining a menu category is added to the Menu Category [sys_app_category] table. The menu category style is defined in the `css-file.css` file.

```
import { Record } from "@servicenow/sdk/core";

export const appCategory = Record({
    table: 'sys_app_category',
    $id: Now.ID[9],
    data: {
        name: 'example',
        style: Now.include('./css-file.css'),
    },
})
```

In this example, a record defining an incident is added to the Incident [incident] table.

```
import { Record } from '@servicenow/sdk/core';

export const incident1 = Record({
  $id: Now.ID['incident-1'],
  table: 'incident',
  data: {
    active: 'true',
    approval: 'not requested',
    description: 'Unable to send or receive emails.',
    incidentState: '1',
    shortDescription: 'Email server is down.',
    subcategory: 'email',
```

```
    callerId: '77ad8176731313005754660c4cf6a7de',
  }
})
```

In this example, a record defining a server is added to the Server [cmdb_ci_server] table.

```
import { Record } from '@servicenow/sdk/core';

export const ciserver1 = Record({
  $id: Now.ID['cmdb-ci-server-1'],
  table: 'cmdb_ci_server',
  data: {
    assetTag: 'P1000199',
    attested: 'false',
    canPrint: 'false',
    company: 'e7c1f3d53790200044e0bfc8bcbe5deb',
    cost: '2160',
    costCc: 'USD',
    cpuSpeed: '633',
    cpuType: 'GenuineIntel',
    diskSpace: '100',
    manufacturer: 'b7e7d7d8c0a8016900a5d7f291acce5c',
    name: 'DatabaseServer1',
    os: 'Linux Red Hat',
    shortDescription: 'DB Server',
    subcategory: 'Computer',
  }
})
```

Role API - ServiceNow Fluent

The Role API defines roles [sys_user_role] that grant specific permissions to users of an application.

For general information about user roles, see Managing roles.

Related topics

ServiceNow Fluent

Role object

Create a role [sys_user_role] to control access to applications and their features.

Properties

Name	Type	Description
name	String	A name for the role beginning with the application scope in the following format: `<scope>.<name>`.
assignableBy	String	Other roles that can assign this role to users.
canDelegate	Boolean	Flag that indicates if the role can be delegated to other users. For more information, see Delegating roles. Valid values: • true: The role can be delegated to other users. • false: The role can't be delegated to other users.

Properties (continued)

Name	Type	Description
		Default: true
description	String	A description of what the role can access.
elevatedPrivilege	Boolean	Flag that indicates whether manually accepting the responsibility of using the role before you can access the features of the role is required. For more information about elevated privileges, see Elevated privilege roles. Valid values: • true: You must manually accept the responsibility of using the role before you can access its features. • false: You don't need to manually accept the responsibility of using the role to access its features. Default: false
grantable	Boolean	Flag that indicates whether the role can be granted independently. Valid values: • true: The role can be granted independently. • false: The role can't be granted independently. Default: true
containsRoles	Array	The variable identifiers of other *Role* objects that this role contains.
scopedAdmin	Boolean	Flag that indicates whether the role is an Application Administrator role. For general information about application administration roles, see Application administration. Valid values: • true: The role is an Application Administrator. • false: The role isn't an Application Administrator. Default: false
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*:

Properties (continued)

Name	Type	Description
		• demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { Role } from "@servicenow/sdk/core";

const managerRole = Role({
   $id: Now.ID['manager_role'],
   name: 'x_snc_example.manager'
})

const adminRole = Role({
   $id: Now.ID['admin_role'],
   name: 'x_snc_example.admin',
   containsRoles: [managerRole]
})
```

Script Action API - ServiceNow Fluent

The Script Action API defines script actions [sysevent_script_action] that run when an event occurs.

For general information about scheduled script executions, see Script actions.

Related topics

ServiceNow Fluent

ScriptAction object

Create a script action [sysevent_script_action] that performs a task when triggered by an event.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A unique name for the script action.
script	Script	Required. A server-side script that runs when triggered by an event. This property supports a function from a JavaScript module, a reference to another file in the application that contains a script, or inline JavaScript. Format:

Properties (continued)

Name	Type	Description
		• For functions, use the name of a function, function expression, or default function exported from a JavaScript module and import it into the `.now.ts` file. For information about JavaScript modules, see JavaScript modules and third-party libraries. • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script'` or `` `Script` ``.
eventName	String	Required. The event that triggers the script action to run. For information about creating events, see Create an event.
active	Boolean	Flag that indicates whether the script action is enabled. Valid values: • true: The script action executes when triggered by the event. • false: The script action doesn't execute. Default: false
description	String	A description of the functionality and purpose of the script action.
order	Number	A number indicating the sequence in which the script action should run. If there are multiple script actions on a particular event, the script actions run in the order specified, from lowest to highest. Default: 100
conditionScript	String	A JavaScript conditional statement that specifies the fields and values that must be true for the script to run. **Note:** Don't use this property if you include the condition statement with the *`script`* property. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script'` or `` `Script` ``.
$meta	Object	Metadata for the application metadata.

Properties (continued)

Name	Type	Description
		With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { ScriptAction } from '@servicenow/sdk/core'
import { insertIncident } from '../server/scripts.js'

ScriptAction({
    $id: Now.ID['sample-script-action'],
    name: 'SampleScriptAction',
    active: true,
    description: 'Insert an incident',
    script: insertIncident,
    eventName: 'sample.event',
    order: 100,
    conditionScript: "gs.hasRole('my_role')"
})
```

The *script* property refers to a function from the `scripts.js` module. For example:

```
import { GlideRecord } from '@servicenow/glide'

export const insertIncident = () => {
    var gr = new GlideRecord('incident')
    gr.initialize()
    gr.setValue('short_description', 'New incident from event')
    gr.insert()
}
```

Script Include API - ServiceNow Fluent

The Script Include API defines script includes [sys_script_include] that store JavaScript functions and classes for use by server-side scripts.

Note:
For new scripts, use JavaScript modules instead of script includes when possible to support code reuse and using third-party libraries within a scoped application. For more information about JavaScript module support and limitations, see JavaScript modules and third-party libraries.

For general information about script includes, see Script includes.

Related topics

ServiceNow Fluent

ScriptInclude object

Create a script include [sys_script_include] to define a server-side script that runs when called from other scripts.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. The name of the script include. If you define a class, the name must match the name of the class, prototype, and type. If you use a classless (on demand) script include, the name must match the function name.
script	Script	Required. A server-side script to call from other scripts. The script must define a single JavaScript class or a global function. The class or function name must match the *`name`* property. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``.
apiName	String	An internal name for the script include, which is used to call the script include from out-of-scope applications. Default: <scope>.<name>
description	String	A description of the purpose and function of the script include.
clientCallable	Boolean	Flag that indicates whether client-side scripts can call the script include using GlideAjax. The script include is available to client scripts, list/report filters, reference qualifiers, or if specified as part of the URL. Client callable script includes are invoked from *`GlideAjax`* and require users to satisfy an ACL associated with the script include. Valid values:

Properties (continued)

Name	Type	Description
		• true: The script include is available to client-side scripts. • false: The script include isn't available to client-side scripts. Default: false
mobileCallable	Boolean	Flag that indicates whether the script include is available to client scripts called from mobile devices. Valid values: • true: The script include is available to client scripts called from mobile devices. • false: The script include isn't available to client scripts called from mobile devices. Default: false
sandboxCallable	Boolean	Flag that indicates whether the script include is available to scripts invoked from the script sandbox, such as a query condition. **Important:** Script includes should only be made available to the script sandbox if necessary. Valid values: • true: The script include is available to scripts invoked from the script sandbox. • false: The script include isn't available to scripts invoked from the script sandbox. Default: false
callerAccess	String	An option for how cross-scope access to the script include is permitted. For more information, see Restricted caller access privilege settings. Valid values: • restriction: Calls to the script include must be manually approved. Access requests are tracked in the Restricted Caller Access table with a status of Requested. • tracking: Calls to the script include are automatically approved. Calls are tracked in the Restricted Caller Access table with a status of Allowed.
accessibleFrom	String	Specifies which applications can access the script include. Valid values:

Properties (continued)

Name	Type	Description
		• public: All application scopes can call the script include. • package_private: The script include can only be called from the application scope that it's within. Default: package_private
active	Boolean	Flag that indicates whether the script include is enabled. Valid values: • true: The script include is callable. • false: The script include isn't callable. Default: true
protectionPolicy	String	A policy that determines whether someone can view or edit the script include after the application is installed on their instance. If undefined, other application developers can customize the script include. Valid values: • read: Allows anyone to read values from this downloaded or installed script include. No one can change script values on the instance on which they download or install the script include. • protected: Provides intellectual property protection for application developers. Customers who download the script include cannot see the contents of the script field. The script is encrypted in memory to prevent unauthorized users from seeing it in plain text.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

In the following example, the script include uses a script from the `SampleClass.server.js` file.

```
import { ScriptInclude } from '@servicenow/sdk/core';

ScriptInclude({
    $id: Now.ID['sample-script-include'],
    name: 'SampleScriptInclude',
    script: Now.include("./SampleClass.server.js"),
    description: 'some description',
    apiName: 'x_scope.SampleScriptInclude',
    callerAccess: 'tracking',
    clientCallable: true,
    mobileCallable: true,
    sandboxCallable: true,
    accessibleFrom: 'public',
    active: true,
})
```

Scripted REST API - ServiceNow Fluent

The Scripted REST API defines the endpoints, query parameters, and headers for a scripted REST service [sys_ws_definition].

For general information about scripted REST services, see Scripted REST APIs.

Related topics

ServiceNow Fluent

RestApi object

Create a scripted REST API [sys_ws_definition] to define web service endpoints.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. The name of the API, which is used in the API documentation.
serviceId	String	Required. The API identifier used to distinguish this API in URI paths. It must be unique within the API namespace.
active	Boolean	Flag that indicates whether the API can serve requests. Valid values: • true: The API can serve requests. • false: The API can't serve requests. Default: true
shortDescription	String	A brief description of the API, which is used in the API documentation.

Properties (continued)

Name	Type	Description
consumes	String	A list of media types that resources of the API can consume. Default: application/json,application/xml,text/xml
docLink	String	A URL that links to static documentation about the API.
enforceAcl	Array	A list of variable identifiers of *ACL* objects or sys_ids of ACLs to enforce when accessing resources [sys_security_acl]. For more information, see Access Control List API - ServiceNow Fluent. To not enforce ACLs, set this property to an empty array ([ ]). Default: Scripted REST External Default
produces	String	A list of media types that resources of the API can produce. Default: application/json,application/xml,text/xml
routes	Array	The resources [sys_ws_operation] for the API. For more information, see routes object.
policy	String	The policy for how application files are protected when downloaded or installed. Valid values: • read: Files are viewable only. • protected: Users with password permissions can edit the files.
versions	Array	A list of versions [sys_ws_version] for the API. For more information, see versions object. Specifying versions allows you to manage different versions of an API and their statuses, such as whether they are active, the default version, or deprecated.
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*:

Properties (continued)

Name	Type	Description
		• demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { RestApi } from '@servicenow/sdk/core'
import { process } from '../server/handler.js'

RestApi({
    $id: Now.ID['rest1'],
    name: 'customAPI',
    serviceId: 'custom_api',
    consumes: 'application/json',
    routes: [
        {
            $id: Now.ID['route1'],
            path: '/home/{id}',
            script: process,
            parameters: [{ $id: Now.ID['param1'],
 name: 'n_param' }],
            headers: [{ $id: Now.ID['header1'],  name: 'n_token' }],
            enforceAcl: [acl],
            version: 1,
        },
    ],
    enforceAcl: [acl],
    versions: [
        {
            $id: Now.ID['v1'],
            version: 1,
        },
    ],
})
```

The ACL referenced is defined using the *ACL* object:

```
import { Acl } from "@servicenow/sdk/core";

const acl = Acl({
    name: 'My random ACL',
    type: 'rest_endpoint',
    script: `answer = (Math.random() > 0.5)`,
    active: true,
    adminOverrides: false,
    operations: ['execute'],
})
```

routes object

Create a scripted REST resource [sys_ws_operation] to define the HTTP method, the processing script, and to override settings from the parent service.

Use the `routes` object within the `RestApi` object.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	The name of the API resource, which is used in the API documentation. Default: the value of the `path` property
script	Script	Required. The custom script defines how the operation parses and responds to requests. This property supports a function from a JavaScript module, a reference to another file in the application that contains a script, or inline JavaScript. Format: • For functions, use the name of a function, function expression, or default function exported from a JavaScript module and import it into the `.now.ts` file. For information about JavaScript modules, see JavaScript modules and third-party libraries. • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or `Script``.
parameters	Array	A list of query parameters [sys_ws_query_parameter] for the route. For more information, see parameters and headers objects.
headers	Array	A list of headers [sys_ws_header] for the route. For more information, see parameters and headers objects.
active	Boolean	Flag that indicates whether the resource is used. Valid values: • true: The resource is used. • false: The resource isn't used.

Properties (continued)

Name	Type	Description
		Default: true
path	String	The path of the resource relative to the base API path. The relative URI can contain path parameters such as `'/abc/{id}'`. Default: /
shortDescription	String	A brief description of the resource, which is used in the API documentation.
consumes	String	A list of media types that the resource can consume. This property can be overridden with the PUT, PATCH, or POST methods. Default: The value of the *`consumes`* property in the *`RestApi`* object
enforceAcl	Array	A list of variable identifiers of *`ACL`* objects or sys_ids of ACLs to enforce when accessing resources [sys_security_acl]. For more information, see Access Control List API - ServiceNow Fluent. To not enforce ACLs, set this property to an empty array ([ ]). Default: Scripted REST External Default
produces	String	A list of media types that the resource can produce. Default: The value of the *`produces`* property in the *`RestApi`* object
requestExample	String	A valid sample request body payload for the resource, which is used in the API documentation.
method	String	The HTTP method that the resource implements. Valid values: GET, POST, PUT, PATCH, DELETE Default: GET
authorization	Boolean	Flag that indicates whether users must be authenticated to access the resource. Valid values: • true: Users must be authenticated to access the resource. • false: Authentication isn't required to access the resource. Default: true

Properties (continued)

Name	Type	Description
authentication	Boolean	Flag that indicates whether ACLs are enforced when accessing the resource. Valid values: • true: ACLs are enforced when accessing the resource. • false: ACLs aren't enforced when accessing the resource. Default: true
internalRole	Boolean	Flag that indicates whether the route requires the snc_internal role. This property is supported only if the Explicit Roles plugin (com.glide.explicit_roles) is enabled. Valid values: • true: The route requires the snc_internal role. • false: The route doesn't require the snc_internal role. Default: true
policy	String	The policy for how application files are protected when downloaded or installed. Valid values: • read: Files are viewable only. • protected: Users with password permissions can edit the files.
version	Number	The version of the API. This property is required if the *versions* property is used in the *RestApi* object. The version specified with this property is used to automatically generate a URI with a version, such as `/api/management/v1/table/{tableName}`. Version numbers identify the endpoint version that a URI accesses. By specifying a version number, you can test and deploy changes without impacting existing integrations.
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }`

Properties (continued)

Name	Type	Description
		Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
routes: [
   {
      $id: Now.ID['route1'],
      path: '/home/{id}',
      script: process,
      parameters: [{ $id: Now.ID['param1'],  name: 'n_param' }],
      headers: [{ $id: Now.ID['header1'],  name: 'n_token' }],
      enforceAcl: [acl],
      version: 1,
   },
],
```

parameters and headers objects

Create query parameters [sys_ws_query_parameter] and headers [sys_ws_header] for routes in a scripted REST API. Query parameters control what values a requesting user can pass in the request URI. Headers specify what the API accepts and can respond with.

Use the *parameters* and *headers* objects within the *routes* object.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. The name of the parameter or header, which is used in the API documentation.
required	Boolean	Flag that indicates whether the parameter or header is required. Valid values: • true: The parameter or header is required. • false: The parameter or header isn't required. Default: false

Properties (continued)

Name	Type	Description
exampleValue	String	An example of a valid value for the parameter or header, which is used in the API documentation.
shortDescription	String	A brief description of the parameter or header, which is used in the API documentation.
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
parameters: [{ $id: Now.ID['param1'],  name: 'n_param' }],
headers: [{ $id: Now.ID['header1'],  name: 'n_token' }],
```

versions object

Create versions for a scripted REST API [sys_ws_version] to define web service endpoints.

Use the *versions* object within the *RestApi* object.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
version	Number	Required. A version of the REST API.
active	Boolean	Flag that indicates whether the version of the REST API can serve requests. Valid values: • true: The version of the API can serve requests. • false: The version of the API can't serve requests.

Properties (continued)

Name	Type	Description
		Default: true
deprecated	Boolean	Flag that indicates whether the version of the REST API is deprecated. Resources belonging to deprecated versions can serve requests, but are identified as deprecated in documentation. Valid values: • true: The version of the API is identified as deprecated. • false: The version of the API isn't identified as deprecated. Default: false
shortDescription	String	A brief description of the version of the REST API, which appears in the API documentation.
isDefault	Boolean	Flag that indicates whether the version of the REST API is the default version. Clients can access the default version using either the versioned or non-versioned URI path. Valid values: • true: The version of the API is the default version. • false: The version of the API isn't the default version. Default: false
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
versions: [
 {
   $id: Now.ID['v1'],
```

```
    version: 1,
  },
],
```

Service Catalog API - ServiceNow Fluent

The Service Catalog API defines catalog items [sc_cat_item], record producers [sc_cat_item_producer], and related aspects of service catalogs.

For general information about service catalogs, see Service Catalog.

Related topics

ServiceNow Fluent

CatalogItem object

Create a catalog item [sc_cat_item] that users can request from a service catalog.

A catalog item must reference a flow, workflow, or execution plan that defines how the item request is fulfilled.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A name for the item that appears in the catalog.
active	Boolean	Flag that indicates whether the item is active and available to be ordered. Default: true
availability	String	The type of device on which the item displays. Valid values: • desktopOnly: The item displays only on desktop devices. • mobileOnly: The item displays only on mobile devices. • both: The item displays on both desktop and mobile devices. Default: desktopOnly
checkedOut	Boolean	Flag that indicates whether the item is checked out for editing. Default: false
description	String	A detailed description of the item that displays in the catalog when a user selects the item or the associated **Preview** link. You can embed videos, images, links to internal knowledge base (KB) articles, and links to external sources of information and instruction documentation.

Properties (continued)

Name	Type	Description
meta	Array	A list of metadata tags used to aid in searches related to the item. **Note:** The *meta* property is used only for Zing text indexing and search engine and not for AI Search.
model	Reference or String	The variable identifier or sys_id of a product model [cmdb_model] associated with the item. To define a product model, use the Record API - ServiceNow Fluent.
order	Number	The order in which the item appears within its category. Default: 0
owner	Reference or String	The variable identifier or sys_id of a user [sys_user] who owns the item and has edit access to it. To define a user, use the Record API - ServiceNow Fluent.
roles	Array	A list of variable identifiers of *Role* objects or sys_ids of roles [sys_user_role] that can access the item. For more information, see Role API - ServiceNow Fluent.
shortDescription	String	A short description of the item that appears on the catalog home page, search results, and the title bar of the order form.
showVariableHelpOnLoad	Boolean	Flag that indicates whether the item displays variable help text by default. Default: false
startClosed	Boolean	Flag that indicates whether to start the item in a collapsed state. Default: false
state	String	The publication state of the item, such as draft or published.
variables	Object	The variable definitions for the item that provide options for requesting it. Each variable type has a specific function. For example: `variables: {` `  laptopType: SelectBoxVariable({` `  question: "Laptop Type",` `  choices: {` `  standard: { label: "Standard Laptop", sequence: 1 },` `  developer: { label: "Developer Workstation", sequence: 2 }` `  },` `  mandatory: true,` `  order: 100` `  }),`

Properties (continued)

Name	Type	Description
		justification: MultiLineTextVariable({ question: "Business Justification", mandatory: true, order: 200 }) } For general information about catalog variables, see Service catalog variables.
version	Number	A version of the item. Default: 1
image	String	Deprecated. An image for the item.
icon	String	An image file that displays as an icon beside the item name in the catalog. Use a 27x27 pixel image. If no image is uploaded, the default icon appears.
picture	String	An image file to display as a picture of the item.
mobilePicture	String	An image to display as a picture of the item on mobile devices. This property applies only if the value of the *mobilePictureType* is mobilePicture.
mobilePictureType	String	The type of picture to display for the item on mobile devices. Valid values: • desktopPicture: Use the image from the *picture* property. • mobilePicture: Use the image from the *mobilePicture* property. • noPicture: The item doesn't display a picture on mobile devices. Default: desktopPicture
hideAddToCart	Boolean	Flag that indicates whether to hide the **Add to Cart** button. Default: false
hideAddToWishlist	Boolean	Flag that indicates whether to hide the **Add to Wishlist** button. Default: false
hideDeliveryTime	Boolean	Flag that indicates whether to hide the delivery time. Default: false
hideQuantitySelector	Boolean	Flag that indicates whether to hide the **Quantity** field.

Properties (continued)

Name	Type	Description
		Default: false
hideSaveAsDraf	Boolean	Flag that indicates whether to hide the **Save as Draft** button. Default: false
hideSP	Boolean	Flag that indicates whether to hide an item on Service Portal. Default: false
mandatoryAttac	Boolean	Flag that indicates whether to require adding an attachment to submit a request. Default: false
hideAttachment	Boolean	Flag that indicates whether to hide the attachment section and not support adding attachments. Default: false
assignedTopics	Array	A list of sys_ids of existing taxonomy topics that control the visibility of the item in the Employee Center portal. For more information, see Associate a catalog item with a taxonomy topic in Employee Center. **Note:** This property is available only when the Employee Experience Taxonomy plugin (sn_ect) is active.
availableFor	Array	A list of sys_ids of user criteria [user_criteria] that define who can access the item.
notAvailableFor	Array	A list of sys_ids of user criteria [user_criteria] that define who can't access the item. This property overrides the *availableFor* property.
variableSets	Array	A list of variable identifiers of *VariableSet* objects or sys_ids of variable sets [item_option_new_set] to attach to the item. For more information, see VariableSet object. Format: `{ variableSet, order }`
flow	Reference or String	The variable identifier of a *Flow* object or sys_id of a flow [sys_hub_flow] that defines how the request is fulfilled. To define a flow, use the Flow API - ServiceNow Fluent. **Note:** You should use flows as the fulfillment method for catalog items. When *flow*, *workflow*, and *executionPlan* are all specified, the system uses the flow.
executionPlan	Reference or String	The variable identifier or sys_id of an execution plan [sc_cat_item_delivery_plan] that defines how the request is

Properties (continued)

Name	Type	Description
		fulfilled. To define an execution plan, use the Record API - ServiceNow Fluent.
workflow	String	The sys_id of a legacy workflow [wf_workflow] that defines how the item request is fulfilled. Use the `flow` property for new implementations.
accessType	String	The user access required to request the item. Valid values: • restricted: Only users who have access to the item can request the item. • delegated: Users who don't have access to the item can request the item on behalf of someone else using the delegated request experience. For more information, see Delegated request experience. **Note:** This functionality is only applicable when the item has a Requested For variable. Default: restricted
location	Reference or String	The variable identifier or sys_id of a location [cmn_location] where the item is provided. To define a location, use the Record API - ServiceNow Fluent.
vendor	Reference or String	The variable identifier or sys_id of a vendor associated with the item. To define a vendor, use the Record API - ServiceNow Fluent.
deliveryPlanScript	Script	Deprecated. Use the `executionPlan` property instead.
deliveryTime	Object	The estimated time to deliver the item. Format: `deliveryTime: { days: Number, hours: Number })`
entitlementScript	Script	A script that defines entitlement for the item.
makeItemNonConversational	Boolean	Flag that indicates whether to prevent the ability to request the item from a conversational experience such as Virtual Agent. If true, the item can be requested from a conversational experience. Default: false
visibleBundle	Boolean	Flag that indicates whether item is visible in saved bundles. Default: true
visibleGuide	Boolean	Flag that indicates whether item is visible in order guides. Default: true
visibleStandalone	Boolean	Flag that indicates whether the standalone view is visible.

Properties (continued)

Name	Type	Description
		Default: true
fulfillmentAutomationLevel	String	The level of fulfillment automation for requests. Valid values: • unspecified: Automation level of item isn't specified. • manual: Catalog item isn't automated. • semiAutomated: Catalog item is semi-automated. • fullyAutomated: Catalog item is automated completely.
fulfillmentGroup	Reference or String	The variable identifier or sys_id of the group [sys_user_group] responsible for delivering the item. To define a user group, use the Record API - ServiceNow Fluent.
catalogs	Array	A list of sys_ids of the catalogs [sc_catalog] in which the item appears.
categories	Array	A list of sys_ids of the categories [sc_category] to which the item belongs. A catalog must be specified in the `catalogs` property before categories can be assigned. In the Service Portal, catalog searches find only items that are assigned to a category. In Employee Center, catalog searches find only items that are associated with a taxonomy topic. For more information, see Associate a catalog item with a taxonomy topic in Employee Center.
cost	Number	A number indicating the one-time cost of the item. Default: 0
displayPriceProperty	String	The system property that controls how the item price is displayed.
ignorePrice	Boolean	Flag that indicates whether to hide the item price in the cart and in the catalog listing. Default: true
mobileHidePrice	Boolean	Flag that indicates whether to hide the item price on mobile devices. Default: false
omitPrice	Boolean	Flag that indicates whether to omit the price entirely from all views of the item. Default: false
billable	Boolean	Flag that indicates whether the item is billable. Default: false

Properties (continued)

Name	Type	Description
pricingDetails	Array	A list of pricing entries for the item. Valid values: • price: A one-time price for the item. • recurring_price: A price that recurs at a regular interval. When using this value, the *`recurringFrequency`* property is required. Format: `{ amount, currencyType, field }`
recurringFrequency	String	The time interval at which the recurring price repeats, such as monthly or yearly. This property is required when *`pricingDetails`* contains a `recurring_price` field entry.
requestMethod	String	The label for the submission button and the order submission experience. Valid values: • order: Displays an **Order Now** button. A confirmation dialog is shown and delivery information is editable. • request: Displays a **Request** button. A confirmation dialog is shown but delivery information isn't displayed. Use for scenarios where delivery information is pre-determined, such as a code access request. • submit: Displays a **Submit** button. No confirmation dialog or delivery information is displayed. Use for scenarios where no further request information is required, such as a password reset. Default: order
customCart	Reference or String	The variable identifier or sys_id of a custom UI macro to use for cart rendering. To define a custom cart, use the Record API - ServiceNow Fluent.
useScLayout	Boolean	Flag that indicates whether the item uses the Service Catalog layout for display. Default: true
noCart	Boolean	Deprecated. Flag that indicates whether to hide the shopping cart. Default: false
noOrder	Boolean	Deprecated. Flag that indicates whether to hide the **Order** option. Default: false

Properties (continued)

Name	Type	Description
noOrderNow	Boolean	Deprecated. Flag that indicates whether to hide the **Order Now** option. Default: false
noProceedCheckout	Boolean	Deprecated. Flag that indicates whether to hide the checkout process. Default: false
noQuantity	Boolean	Deprecated. Flag that indicates whether to hide the **Quantity** field. Default: false
noSearch	Boolean	Deprecated. Flag that indicates whether to hide search. Default: false
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { CatalogItem } from "@servicenow/sdk/core";

export const softwareLicenseRequest = CatalogItem({
  $id: Now.ID["software_license_request"],
  name: "Software License Request",
  shortDescription: "Request a software license",

  catalogs: [serviceCatalog],
  categories: [softwareCategory],

  // Attach reusable variable sets
  variableSets: [
    { variableSet: contactInfoSet, order: 100 },
    { variableSet: approvalInfoSet, order: 200 }
  ],
```

```
  // Item-specific variables
  variables: {
    software_name: SingleLineTextVariable({
      question: "Software Name",
      mandatory: true,
      order: 100
    }),
    license_type: SelectBoxVariable({
      question: "License Type",
      choices: {
        individual: { label: "Individual", sequence: 1 },
        team: { label: "Team (5 seats)", sequence: 2 },
        enterprise: { label: "Enterprise (unlimited)", sequence: 3 }
      },
      mandatory: true,
      order: 200
    }),
    number_of_licenses: SingleLineTextVariable({
      question: "Number of Licenses",
      defaultValue: "1",
      order: 300
    }),
    justification: MultiLineTextVariable({
      question: "Business Justification",
      mandatory: true,
      order: 400
    })
  },

  // Pricing with recurring charges
  pricingDetails: [
    { amount: 0, currencyType: "USD", field: "price" },
    { amount: 99, currencyType: "USD", field: "recurring_price" }
  ],
  recurringFrequency: "monthly",

  flow: "523da512c611228900811a37c97c2014",
  deliveryTime: { days: 3, hours: 0 }
})
```

CatalogItemRecordProducer object

Create a record producer [sc_cat_item_producer] for users to create task-based records, such as incident or change request records, from the service catalog.

You can create a record producer for tables and database views that are in the same scope as the record producer. You can also create a record producer for tables that allow create access from applications in other scopes.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs.

Properties (continued)

Name	Type	Description
		Format: `Now.ID['String' or Number]`
table	Reference or String	Required. The variable identifier of a *`Table`* object or name of a table in which the record producer creates records, such as incident or change_request.
name	String	Required. A name for the item that appears in the catalog.
active	Boolean	Flag that indicates whether the item is active and available to be ordered. Default: true
availability	String	The type of device on which the item displays. Valid values: • desktopOnly: The item displays only on desktop devices. • mobileOnly: The item displays only on mobile devices. • both: The item displays on both desktop and mobile devices. Default: desktopOnly
checkedOut	Boolean	Flag that indicates whether the item is checked out for editing. Default: false
description	String	A detailed description of the item that displays in the catalog when a user selects the item or the associated **Preview** link. You can embed videos, images, links to internal knowledge base (KB) articles, and links to external sources of information and instruction documentation.
meta	Array	A list of metadata tags used to aid in searches related to the item. **Note:** The *`meta`* property is used only for Zing text indexing and search engine and not for AI Search.
model	Reference or String	The variable identifier or sys_id of a product model [cmdb_model] associated with the item. To define a product model, use the Record API - ServiceNow Fluent.
order	Number	The order in which the item appears within its category. Default: 0
owner	Reference or String	The variable identifier or sys_id of a user [sys_user] who owns the item and has edit access to it. To define a user, use the Record API - ServiceNow Fluent.

Properties (continued)

Name	Type	Description
roles	Array	A list of variable identifiers of *Role* objects or sys_ids of roles [sys_user_role] that can access the item. For more information, see Role API - ServiceNow Fluent.
shortDescription	String	A short description of the item that appears on the catalog home page, search results, and the title bar of the order form.
showVariableHelpOnLoad	Boolean	Flag that indicates whether the item displays variable help text by default. Default: false
startClosed	Boolean	Flag that indicates whether to start the item in a collapsed state. Default: false
state	String	The publication state of the item, such as draft or published.
variables	Object	The variable definitions for the item that provide options for requesting it. Each variable type has a specific function. For example: `variables: { laptopType: SelectBoxVariable({ question: "Laptop Type", choices: { standard: { label: "Standard Laptop", sequence: 1 }, developer: { label: "Developer Workstation", sequence: 2 } }, mandatory: true, order: 100 }), justification: MultiLineTextVariable({ question: "Business Justification", mandatory: true, order: 200 }) }` For general information about catalog variables, see Service catalog variables.
version	Number	A version of the item. Default: 1
view	Reference or String	Required. The variable identifier or name of the UI view [sys_ui_view] which applies, or the default view. To define a UI view, use the Record API - ServiceNow Fluent. To use the default view (`default_view`), you must import it:

Properties (continued)

Name	Type	Description
		`import { default_view } from '@servicenow/sdk/core'`
icon	String	An image file that displays as an icon beside the item name in the catalog. Use a 27x27 pixel image. If no image is uploaded, the default icon appears.
picture	String	An image file to display as a picture of the item.
mobilePicture	String	An image to display as a picture of the item on mobile devices. This property applies only if the value of the *`mobilePictureType`* is `mobilePicture`.
mobilePictureType	String	The type of picture to display for the item on mobile devices. Valid values: • desktopPicture: Use the image from the *`picture`* property. • mobilePicture: Use the image from the *`mobilePicture`* property. • noPicture: The item doesn't display a picture on mobile devices. Default: desktopPicture
assignedTopics	Array	A list of sys_ids of existing taxonomy topics that control the visibility of the item in the Employee Center portal. For more information, see Associate a catalog item with a taxonomy topic in Employee Center. **Note:** This property is available only when the Employee Experience Taxonomy plugin (sn_ect) is active.
availableFor	Array	A list of sys_ids of user criteria [user_criteria] that define who can access the item.
notAvailableFor	Array	A list of sys_ids of user criteria [user_criteria] that define who can't access the item. This property overrides the *`availableFor`* property.
variableSets	Array	A list of variable identifiers of *`VariableSet`* objects or sys_ids of variable sets [item_option_new_set] to attach to the item. For more information, see VariableSet object. Format: `{ variableSet, order }`
hideAddToCart	Boolean	Flag that indicates whether to hide the **Add to Cart** button. Default: false

Properties (continued)

Name	Type	Description
hideAddToWishlist	Boolean	Flag that indicates whether to hide the **Add to Wishlist** button. Default: false
hideDeliveryTime	Boolean	Flag that indicates whether to hide the delivery time. Default: false
hideQuantitySelector	Boolean	Flag that indicates whether to hide the **Quantity** field. Default: false
hideSaveAsDraft	Boolean	Flag that indicates whether to hide the **Save as Draft** button. Default: false
hideSP	Boolean	Flag that indicates whether to hide an item on Service Portal. Default: false
mandatoryAttachment	Boolean	Flag that indicates whether to require adding an attachment to submit a request. Default: false
hideAttachment	Boolean	Flag that indicates whether to hide the attachment section and not support adding attachments. Default: false
allowEdit	Boolean	Flag that indicates whether users can edit the created record after submission. Default: false
canCancel	Boolean	Flag that indicates whether to display a Cancel button on the record producer form. Users can select Cancel to cancel the record producer and return to the last-viewed screen. Default: false
postInsertScript	Script	A server-side script that runs after the record is inserted in the associated table. You can call `current.update()` in this script. **Note:** This script overrides the target record values and record producer template values. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format:

Properties (continued)

Name	Type	Description
		• To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``. Default: `/**` `* This script is executed after the record is generated.` `* `current` Is the GlideRecord produced by Record Producer. Use `current.update()` to update the record` `* To access the variables, use `producer.var1` where var1 is the name of the variable` `* To access the Record Producer use `cat_item`` `*/`
redirectUrl	String	The redirect destination after the record is generated. Valid values: • generatedRecord: Redirects to the task record created by the record producer. • catalogHomePage: Redirects to the service catalog. Default: generatedRecord
saveOptions	String	Advanced configuration options for saving the record producer.
saveScript	Script	A script that runs at every step save in Catalog Builder. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``. Default: `/**` `* This script is executed at every step save in Catalog Builder.` `* This script is executed before `Script` is executed.` `* `current` Is the GlideRecord produced by Record Producer.`

Properties (continued)

Name	Type	Description
		* To access the variables, use `producer.var1` where var1 is the name of the variable * To access the Record Producer use `cat_item` */
script	Script	A server-side script that runs before the record is created. Use this script to assign values to fields on the record dynamically. Don't call `current.update()` or `current.insert()` in this script. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``. Default: /** This script is executed before the Record is generated * `current`- GlideRecord produced by Record Producer * Don't use `current.update()` or `current.insert()` as the record is generated by Record Producer * Don't use `current.setValue('sys_class_name', 'xxx')` as this will trigger reparent flow and can cause data loss * Avoid `current.setAbortAction()` and generate a separate record * Use `producer.var1` to access variables */
image	String	Deprecated. An image for the item.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*:

Properties (continued)

Name	Type	Description
		• demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { CatalogItemRecordProducer } from "@servicenow/sdk/core";

const serviceCatalog = "e0d08b13c3330100c8b837659bba8fb4";
const itServicesCategory = "d258b953c611227a0146101fb1be7c31";

export const comprehensiveIncidentProducer =
 CatalogItemRecordProducer({
  $id: Now.ID["comprehensive_incident_producer"],
  name: "Report Incident with Full Configuration",
  shortDescription: "Complete incident producer with variables and scripts",
  table: "incident",

  catalogs: [serviceCatalog],
  categories: [itServicesCategory],

  variables: {
    short_description: SingleLineTextVariable({
      question: "Brief Summary",
      mandatory: true,
      mapToField: true,
      field: "short_description",
      order: 100
    }),
    urgency: SelectBoxVariable({
      question: "Urgency",
      mandatory: true,
      mapToField: true,
      field: "urgency",
      choices: {
        "1": { label: "High", sequence: 1 },
        "2": { label: "Medium", sequence: 2 },
        "3": { label: "Low", sequence: 3 }
      },
      order: 200
    }),
    assignment_group: ReferenceVariable({
      question: "Assignment Group",
      mapToField: true,
      field: "assignment_group",
      referenceTable: "sys_user_group",
      order: 300
    })
  },

  script: Now.include("../../scripts/rp-pre-insert.js"),
```

```
  postInsertScript: Now.include("../../scripts/rp-post-insert.js"),

  redirectUrl: "generatedRecord",
  view: "ess",
  allowEdit: true
})
```

CatalogUiPolicy object

Configure a catalog UI policy [catalog_ui_policy] to control variable behavior on catalog item forms based on conditions.

Catalog UI policies can make variables mandatory, read-only, visible, or hidden when specified conditions are met. For validation, calculations, or asynchronous calls, use catalog client scripts instead.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
shortDescription	String	Required. A short description of what the catalog UI policy does.
catalogItem	Reference or String	Required only if the *`variableSet`* property isn't used. The variable identifier of a *`CatalogItem`* or *`CatalogItemRecordProducer`* object or sys_id of the catalog item [sc_cat_item] or record producer [sc_cat_item_producer] to which the UI policy applies. For more information, see CatalogItem object or CatalogItemRecordProducer object.
variableSet	Reference or String	Required only if the *`catalogItem`* property isn't used. The variable identifier of a *`VariableSet`* object or sys_id of the variable set [item_option_new_set] to which the UI policy applies. For more information, see VariableSet object.
appliesTo	String	The scope to which the UI policy applies. Valid values: • item: The policy applies to a catalog item. • set: The policy applies to a variable set. Default: item
active	Boolean	Flag that indicates whether the catalog UI policy is active. Default: true
global	Boolean	Flag that indicates on which views of the table the Catalog UI Policy runs. Default: true

Properties (continued)

Name	Type	Description
onLoad	Boolean	Flag that indicates whether the catalog UI policy runs when the form loads. If false, the policy applies only when variable values change. Default: true
reverseIfFalse	Boolean	Flag that indicates whether to reverse the UI policy actions when the catalog condition evaluates to false. Default: true
inherit	Boolean	Flag that indicates whether the Catalog UI Policy is inherited. Default: false
isolateScript	Boolean	Flag that indicates whether the policy scripts run in an isolated scope. Default: true
catalogCondition	String	The encoded query conditions based on catalog item variable values that trigger the UI policy. For example: `catalogCondition: ` `${catalogItem.variables.singleLineText}=catalogitem^${catalogItem.variables.reference}ISNOTEMPTY^EQ`
appliesOnCatalogItemView	Boolean	Flag that indicates whether the UI policy applies to catalog items displayed in the order screen. This view is available to requesters. Default: true
appliesOnTargetRecord	Boolean	Flag that indicates whether the UI policy applies to records created for task-extended tables via record producers. Default: false
appliesOnCatalogTasks	Boolean	Flag that indicates whether the UI policy applies to catalog task forms. This view is available to fulfillers. Default: false
appliesOnRequestedItems	Boolean	Flag that indicates whether the UI policy applies to requested item forms. This view is available to fulfillers. Default: false
runScripts	Boolean	Flag that indicates whether to run the *executeIfTrue* and *executeIfFalse* scripts for this UI policy. Use scripts to apply behaviors beyond read-only, mandatory, or visible, such as targeting a specific role.

Properties (continued)

Name	Type	Description
		Default: false
executeIfTrue	String	A client-side script that runs when the catalog condition evaluates to true. The script must be wrapped in a `function onCondition() {}` function.
executeIfFalse	String	A client-side script that runs when the catalog condition evaluates to false. The script must be wrapped in a `function onCondition() {}` function.
runScriptsInUiType	String	The UI type on which the policy scripts run. Valid values: • desktop: Scripts run on the desktop interface. • mobileOrServicePortal: Scripts run on the mobile and Service Portal interfaces. • all: Scripts run on all interfaces. Default: desktop
vaSupported	Boolean	Flag that indicates whether the UI policy is supported in Virtual Agent conversations. Default: false
actions	Array	A list of variable actions to perform when the catalog condition is met. For more information, see actions array.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { CatalogUiPolicy } from "@servicenow/sdk/core";
import { hardwareRequestItem } from "./catalog-items/HardwareRequest.now";

export const managerApprovalPolicy = CatalogUiPolicy({
  $id: Now.ID["manager_approval_policy"],
  shortDescription: "Show manager approval when high priority selected",
  catalogItem: hardwareRequestItem,
```

```
  catalogCondition:
 `${hardwareRequestItem.variables.priority}=high^EQ`,
  actions: [
    {
      variableName:
 hardwareRequestItem.variables.manager_approval,
      visible: true,
      mandatory: true
    }
  ]
})
```

actions array

Configure the variable actions [catalog_ui_policy_action] that a catalog UI policy performs on variables when its conditions are met.

Each action in the array specifies a variable and the property changes to apply to that variable when the UI policy condition evaluates to true.

Properties

Name	Type	Description
variableName	String	Required. The variable to which the action applies.
visible	Boolean	Flag that indicates whether to make the variable visible. Default: false
disabled	Boolean	Flag that indicates whether to turn off the variable. Default: false
mandatory	Boolean	Flag that indicates whether to make the variable required. Default: false
cleared	Boolean	Flag that indicates whether to clear the variable value when the condition is met. Default: false
variableMessage	String	A message to display on the variable when the condition is met. This property applies only if the *variableMessageType* property has a value.
variableMessageType	String	The type of field message. Valid values: • info • warning • error
value	String	The value to set on the variable when the condition is met.

Properties (continued)

Name	Type	Description
		This property applies only if the *valueAction* property is set to setValue.
valueAction	String	The action to take on the variable value when the condition is met. Valid values: • setValue: Sets the variable to the value specified in the *value* property. • clearValue: Clears the variable value.
order	Number	The order in which the action is evaluated relative to other actions. Default: 100
readOnly	Boolean	Flag that indicates whether to make the variable read only. Default: false

```
actions: [
    {
      variableName: laptopRequest.variables.justification,
      mandatory: true,
      variableMessage: "Justification required for urgent requests",
      variableMessageType: "info",
      order: 100
    },
    {
      variableName: laptopRequest.variables.manager_approval,
      visible: true,
      mandatory: true,
      order: 200
    },
    {
      variableName: laptopRequest.variables.delivery_date,
      visible: true,
      order: 300
    }
]
```

CatalogClientScript object

Configure a catalog client script [catalog_script_client] that runs on the client side to control the behavior of a catalog item form.

Use catalog client scripts to validate user input, auto-populate fields, or display alerts on catalog item forms. For simple show/hide, mandatory, and read-only logic, use catalog UI policies instead.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A unique name for the catalog client script.
script	Script	The client-side script to run on the catalog item form. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``.
type	String	The event that triggers the client script to run. Valid values: • onLoad: Runs when the form loads. Use for initial setup such as setting field states, default values, and visibility. • onChange: Runs when a specific variable changes. Include an `if (isLoading) return;` guard. • onSubmit: Runs when the form is submitted. Return `false` to block submission. Avoid using *`GlideAjax`* in this type because of asynchronous issues.
uiType	String	The UI type on which the client script runs. Valid values: • desktop: Applies to the desktop interface. • mobileOrServicePortal: Applies to the mobile and Service Portal interfaces. • all: Applies to all interfaces. Default: desktop
active	Boolean	Flag that indicates whether the client script is enabled. Default: true
appliesTo	String	Required if using the *`variableSet`* property. The scope to which the catalog client script applies. Valid values:

Properties (continued)

Name	Type	Description
		• item: The script applies to a catalog item. • set: The script applies to a variable set. Default: item
catalogItem	Reference or String	Required only if the *variableSet* property isn't used. The variable identifier of a *CatalogItem* or *CatalogItemRecordProducer* object or sys_id of the catalog item [sc_cat_item] or record producer [sc_cat_item_producer] to which the client script applies. For more information, see CatalogItem object or CatalogItemRecordProducer object.
variableSet	Reference or String	Required only if the *catalogItem* property isn't used. The variable identifier of a *VariableSet* object or sys_id of the variable set [item_option_new_set] to which the UI policy applies. For more information, see VariableSet object.
variableName	String	Required if the value of the *type* property is `onChange`. The name of the catalog variable that triggers the script when its value changes.
appliesOnCatalogItemView	Boolean	Flag that indicates whether the client script applies to catalog items displayed in the order screen of the service catalog. This view is available to requesters. Default: true
appliesOnRequestedItems	Boolean	Flag that indicates whether the client script applies to requested item forms after the item is requested. This view is available to fulfillers. Default: false
appliesOnCatalogTasks	Boolean	Flag that indicates whether the client script applies to catalog task forms for the item. This view is available to fulfillers. Default: false
appliesOnTargetTask	Boolean	Flag that indicates whether the client script applies to records created for task-extended tables via record producers. Default: false
global	Boolean	Flag that indicates whether the client script runs in the global scope. Default: true
vaSupported	Boolean	Flag that indicates whether the client script is supported in Virtual Agent conversations. Default: false

Properties (continued)

Name	Type	Description
publishedRef	String	The sys_id of a published catalog item [sc_cat_item] or record producer [sc_cat_item_producer] that this client script references.
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { CatalogClientScript } from "@servicenow/sdk/core";
import { laptopRequest } from "../catalog-items/laptop-request.now";

CatalogClientScript({
  $id: Now.ID["laptop_onload"],
  name: "Laptop Request - OnLoad",
  script: Now.include("../../client/laptop-onload.js"),
  type: "onLoad",
  catalogItem: laptopRequest,
  active: true,
  appliesOnCatalogItemView: true
});
```

The client script is defined in the `laptop-onload.js` file referenced from the *script* property. For example:

```
function onLoad() {
   // Set initial field states
   g_form.setReadOnly("estimated_cost", true);
   g_form.setValue("estimated_cost", "$0");
   g_form.setMandatory("justification", true);
}
```

VariableSet object

Create a variable set [item_option_new_set] that groups reusable variables for use across multiple catalog items and record producers.

Variable sets are reusable collections of variables that can be attached to catalog items and record producers. Catalog UI policies and client scripts can be scoped to a variable set by setting the `appliesTo` property to `set`.

Note:

- Variable sets within a catalog item can't have the same internal name.
- Within a catalog item, the name of a variable can't be the same as the title or internal name of a variable set.
- Catalog client scripts and catalog UI policy scripts must refer to the internal name of a variable set, not the title or display name.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
title	String	Required. The display title of the variable set. This title appears only if the the *`displayTitle`* property is set to true.
internalName	String	The internal name of the variable set used for programmatic access through the *`g_form`* API and server-side scripts. If not provided, the internal name is automatically generated from the *`title`* property.
description	String	A description of the variable set and its intended use.
type	String	The type of the variable set. Valid values: • singleRow: Variables are arranged in a single row. • multiRow: Variables are arranged in multiple rows that users can add or remove. Default: singleRow
layout	String	The column layout for the variable set. Valid values: • normal: Variables are displayed in a single column. • 2down: Variables are displayed in two columns, one side then the other. • 2across: Variables are displayed in two columns with alternating sides. Default: normal
order	Number	The order in which a variable set displays relative to other variable sets on the form. Default: 100
displayTitle	Boolean	Flag that indicates whether to display a collapsible section header for the variable set. If true, the value of the *`title`* property is displayed as a collapsible header.

Properties (continued)

Name	Type	Description
		Note: Check box variables are grouped under a default title of Options. To use a custom title, add a Label variable with an `order` value that places it directly above the check box variables. Default: false
setAttributes	String	Additional configuration attributes for the variable set as a comma-separated string, such as `max_rows=10,collapsible=true`. Use the `max_rows` attribute to set the maximum number of rows for a multi-row variable set. **Note:** `AttachmentVariable`, `ContainerVariable`, `HtmlVariable`, and `CustomVariable` types aren't supported in multi-row variable sets.
readRoles	Array	A list of variable identifiers of `Role` objects or sys_ids of roles [sys_user_role] that can view the variable set. For more information, see Role API - ServiceNow Fluent.
writeRoles	Array	A list of variable identifiers of `Role` objects or sys_ids of roles [sys_user_role] that can modify variable values in the set. For more information, see Role API - ServiceNow Fluent.
createRoles	Array	A list of variable identifiers of `Role` objects or sys_ids of roles [sys_user_role] that can create row instances. For more information, see Role API - ServiceNow Fluent. This property applies only if the value of the `type` property is `multiRow`.
variables	Object	The variable definitions for the item that provide options for requesting it. Each variable type has a specific function. For example: `variables: { laptopType: SelectBoxVariable({ question: "Laptop Type", choices: { standard: { label: "Standard Laptop", sequence: 1 }, developer: { label: "Developer Workstation", sequence: 2 } }, mandatory: true, order: 100 }), justification: MultiLineTextVariable({ question: "Business Justification", mandatory: true,`

Properties (continued)

Name	Type	Description
		<pre>order: 200 }) }</pre> For general information about catalog variables, see Service catalog variables.
name	String	An optional name for additional identification of the variable set.
version	Number	The version of the variable set. Default: 0
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. <pre>$meta: { installMethod: 'String' }</pre> Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import {
  VariableSet,
  EmailVariable,
  SingleLineTextVariable,
  ReferenceVariable
} from "@servicenow/sdk/core";

export const contactInfoSet = VariableSet({
  $id: Now.ID["contact_info_set"],
  title: "Contact Information",
  description: "Standard contact information fields",
  type: "singleRow",
  layout: "2across",
  order: 100,
  displayTitle: true,
  variables: {
    email: EmailVariable({
      question: "Email Address",
      mandatory: true,
      order: 100
    }),
```

```
    phone: SingleLineTextVariable({
      question: "Phone Number",
      mandatory: true,
      order: 200
    }),
    department: ReferenceVariable({
      question: "Department",
      referenceTable: "cmn_department",
      referenceQualCondition: "active=true",
      order: 300
    })
  }
})
```

Service Level Agreement API - ServiceNow Fluent

The Service Level Agreement API defines service level agreements [contract_sla] that set the amount of time for a task to reach a specified condition, ensuring that tasks are resolved according to the service levels agreed between a service provider and customer.

For general information about SLAs, see Service Level Management.

Related topics

ServiceNow Fluent

Sla object

Create an SLA definition [contract_sla] that controls the timing, conditions, workflows, and other information required to create and progress task SLAs.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A name for the SLA definition.
table	String	The name of the table to which the SLA applies. Default: incident
type	String	The type of service level agreement. The type is used for reporting purposes only. Valid values: • SLA: A service level agreement between a service provider and an external customer. • OLA: An operational level agreement between internal teams within the same organization. • Underpinning contract: A contract with an external vendor that underpins the service delivered to the customer.

Properties (continued)

Name	Type	Description
		Default: SLA
active	Boolean	Flag that indicates whether the SLA definition is active and can be matched against task records. Valid values: • true: The SLA definition is active. • false: The SLA definition is inactive and isn't matched against task records. Default: true
target	String	The stage of the task that the SLA measures. The target is used for filtering, searching, and reporting purposes only. Valid values: • response: The SLA measures the time to first respond to a task. • resolution: The SLA measures the time to resolve a task.
duration	Object	The time duration within which the task must reach the target condition. Use the `Duration()` function to specify the duration. This property is required if the value of the *`durationType`* property is empty. Format: `duration: Duration({ days: Number, hours: Number, minutes: Number, seconds: Number })`
durationType	Reference or String	The sys_id of a relative duration [cmn_relative_duration], such as Breach on Due Date or End of next business day, to use instead of a user-specified duration. To define a relative duration, use the Record API - ServiceNow Fluent.
relativeDurationWorks	String	The record type from which the relative duration is calculated. This property only applies if the value of the *`durationType`* property is set. Valid values: • Task record: The relative duration is calculated from the task record. • SLA record: The relative duration is calculated from the SLA record. Default: Task record

Properties (continued)

Name	Type	Description
schedule	Reference or String	The sys_id of a schedule [cmn_schedule] for the time periods during which the SLAs accumulate business time. To define a schedule, use the Record API - ServiceNow Fluent. This property is required if the value of the *`scheduleSource`* property is `sla_definition`.
scheduleSource	String	The source from which the schedule is obtained. Valid values: • sla_definition: The schedule is specified in the SLA definition using the *`schedule`* property. • task_field: The schedule is obtained from a field on the task record specified with the *`scheduleSourceField`* property. • no_schedule: No schedule is used, and the SLA runs continuously (24x7). Default: sla_definition
scheduleSourceField	String	The field from the task that provides the schedule. This property is required if the value of the *`scheduleSource`* property is `task_field`.
conditions	Object	Encoded query conditions that control the timing of an SLA. For example, `'priority=1^state!=6'` matches tasks with a priority of 1 that aren't in a closed state. For more information about SLA conditions, see Create an SLA definition. For information about filter queries, see Operators available for filters and queries. Format: `conditions: { start: 'String', stop: 'String', pause: 'String', resume: 'String', reset: 'String', cancel: 'String', }`
advancedConditionType	String	The type of advanced condition logic to apply. Valid values: • none: No advanced condition logic is applied. • advanced: A custom advanced condition script is used. • advanced_journal: Advanced condition logic based on journal entries is used.

Properties (continued)

Name	Type	Description
		• advanced_system: Advanced condition logic based on system events is used. • advanced_journal_and_system: Advanced condition logic based on both journal entries and system events is used. Default: none
conditionType	String	The sys_id of an SLA condition [sla_condition_class] that determines when transitions between different stages of each task SLA.
resetAction	String	The action to take on the current task SLA record when the reset condition is met. Valid values: • cancel: The current task SLA record is canceled and a new one is created. • complete: The current task SLA record is marked as complete and a new one is created. Default: cancel
whenTo	Object	Settings that control when a paused SLA resumes and when an SLA cancels. • `resume`: The behavior that controls when a paused SLA resumes timing. Valid values: ○ on_condition: The SLA resumes when the task record matches the condition defined with the `resume` property of the `conditions` object. ○ no_match: The SLA resumes when the task record no longer matches the pause condition defined with the `pause` property of the `conditions` object. Default: on_condition • `cancel`: The behavior that controls when an active SLA is canceled. Valid values:

Properties (continued)

Name	Type	Description
		○ on_condition: The SLA is canceled when the task record matches the condition defined with the *cancel* property of the *conditions* object. ○ no_match: The SLA is canceled when the task record no longer matches the start condition defined with the *start* property of the *conditions* object. ○ never: The SLA is never canceled regardless of task record state. Default: on_condition Format: `whenTo: { resume: 'String', cancel: 'String', }`
retroactive	Object	Settings that control whether the SLA start time is set to a point in the past based on a specified field value. • *start*: Flag that indicates whether retroactive start is enabled. When enabled, the SLA start time is set to the value of the field specified in *setStartTo* rather than the time when the start condition was first met. Default: false • *setStartTo*: The field on the task record whose value is used as the SLA start time when retroactive start is enabled. This property is required if the value of the *start* property is true. • *pause*: Flag that indicates whether retroactive pause is enabled when retroactive start is active. When enabled, any time that the task was in a paused state before the SLA was attached is subtracted from the elapsed time. Default: true Format: `retroactive: { start: Boolean, setStartTo: 'String', pause: Boolean, }`
timezoneSource	String	The source from which the time zone for SLA calculations is obtained. Valid values:

Properties (continued)

Name	Type	Description
		• task.caller_id.time_zone: The time zone of the caller on the task record. • task.caller_id.location.time_zone: The time zone of the caller's location on the task record. • task.cmdb_ci.location.time_zone: The time zone of the configuration item's location on the task record. • task.location.time_zone: The time zone of the location on the task record. • sla.timezone: The time zone specified with the *`timezone`* property. Default: task.caller_id.time_zone
timezone	String	The time zone to use for SLA calculations, such as `US/Pacific`. This property is required if the value of the *`timezoneSource`* property is `sla.timezone`.
overrides	Reference or String	The variable identifier or sys_id of another SLA definition [contract_sla] that this SLA definition overrides.
workflow	Reference or String	The variable identifier or sys_id of a workflow [wf_workflow] to run when the SLA reaches a milestone or breaches. To define a workflow, use the Record API - ServiceNow Fluent.
flow	Reference or String	The variable identifier or sys_id of a flow [sys_hub_flow] to run when the SLA reaches a milestone or breaches. To define a flow, use the Flow API - ServiceNow Fluent. Default: Default SLA flow (828f267973333300e289235f04f6a7a3)
vendor	Reference or String	The variable identifier or sys_id of a company [core_company] that is the vendor for an underpinning contract SLA. To define a company, use the Record API - ServiceNow Fluent.
domainPath	String	The domain path that determines which domain owns the SLA in a multi-domain environment. Default: The global domain (/)
enableLogging	Boolean	Flag that indicates whether debug logging is enabled for the SLA definition. Valid values: • true: Debug logging is enabled. • false: Debug logging is turned off. Default: false
$meta	Object	Metadata for the application metadata.

Properties (continued)

Name	Type	Description
		With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { Sla, Duration } from '@servicenow/sdk/core'

Sla({
    $id: Now.ID['incident-p1-resolution'],
    name: 'P1 Incident Resolution',
    table: 'incident',
    target: 'resolution',
    duration: Duration({ hours: 4 }),
    schedule: 'b1992362eb601100fcfb858ad106fe16',
    conditions: {
        start: 'priority=1',
        stop: 'state=6',
        pause: 'state=3',
        resume: 'state!=3',
    },
    whenTo: {
        resume: 'on_condition',
    },
    resetAction: 'cancel',
})
```

Service Portal API - ServiceNow Fluent

The Service Portal API defines custom widgets [sp_widget] for portal pages.

For general information about portals, see Service Portal.

Related topics

ServiceNow Fluent

SPWidget object

Create a custom widget [sp_widget] to include on a portal page.

For general information about creating custom widgets, see Developing custom widgets.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A name for the widget.
category	String	The type of widget. Valid values: • standard • otherApplications • custom • sample • knowledgeBase • servicePortal • serviceCatalog Default: custom
clientScript	Script	A client-side script that defines the AngularJS controller. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``. Default: `api.controller=function() { /* widget controller */ var c = this;};`
serverScript	Script	A server-side script that sets the initial widget state, sends data to the widget's client script using the `data` object, or runs server-side queries. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format:

Properties (continued)

Name	Type	Description
		• To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``. Default: `(function () { /* populate the 'data' object */ /* e.g., data.table = $sp.getValue('table'); */ })();`
controllerAs	String	A variable for a reference to the controller in the directive's scope. The client script accesses the server data object using the `c.data` variable by default. Default: c
htmlTemplate	String	The body HTML code that defines what is rendered when the page is shown. It can contain either static XHTML, dynamically generated content defined as Jelly, or call script includes and UI Macros. This property supports a reference to another file in the application that contains HTML or inline HTML. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide inline HTML, use string literals or template literals for multiple lines of code: `'HTML' or` `` `HTML` ``. Default: `<div><!-- your widget template --></div>`
customCss	String	The CSS or SCSS that defines the widget style. This property supports a reference to another file in the application that contains CSS or inline CSS. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide inline CSS, use string literals or template literals for multiple lines of code: `'CSS' or` `` `CSS` ``.
dataTable	String	The table in which to store widget instance options. To define a custom option schema, you can add fields to a table that extends the Widget Instance [sp_instance] table and set your

Properties (continued)

Name	Type	Description
		widget to use the extension table as a data source. For more information, see Store instance options in a table. Default: sp_instance
demoData	String or Object	Data that demonstrates the widget functionality. This property supports inline strings, a reference to another file in the application that contains JSON, or inline JSON serializable objects. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide inline JSON, use an object with key and value pairs: `{ key: value }`.
description	String	A description of the widget and its purpose.
docs	Reference	The variable identifier of the Service Portal documentation [sp_documentation] that provides additional information about the widget and its purpose. To define Service Portal documentation, see Record API - ServiceNow Fluent.
fields	Array	A list of column names from the data table to use in the widget option schema.
hasPreview	Boolean	Flag that indicates whether you can preview the widget from the Widget Editor. Valid values: • true: You can preview the widget from the Widget Editor. • false: You can't preview the widget from the Widget Editor. Default: false
id	String	A unique ID for the widget. The ID can't contain spaces.
linkScript	Script	A link function that uses AngularJS to directly manipulate the DOM. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``.

Properties (continued)

Name	Type	Description
		Default: `function link(scope, element, attrs, controller) { }`
roles	Array	A list of variable identifiers of *Role* objects or names of roles that can access the widget. For more information, see Role API - ServiceNow Fluent.
optionSchema	Array	A list of parameters that a Service Portal administrator (sp_admin) can configure for a widget instance. The widget option schema supports reusing a widget and uniquely configuring instances of the widget on different pages. For more information, see Widget option schema. `optionSchema: [ { name: 'String', label: 'String', section: 'String', type: 'String', defaultValue: 'String', hint: 'String' }, ... ]` • name: Required. The name of the parameter. • label: Required. A label for the parameter that appears in the widget instance options. • section: Required. The section of the widget instance options in which the parameter appears. Valid values: ○ data ○ behavior ○ documentation ○ presentation ○ other • type: Required. The data type of the parameter. Valid values: ○ string ○ boolean ○ integer ○ reference ○ choice ○ fieldList

Properties (continued)

Name	Type	Description
		○ fieldName ○ glideList ○ glyphIcon • defaultValue: The default value of the parameter. • hint: A short description of the parameter that displays as tooltip when hovering over it.
public	Boolean	Flag that indicates whether the widget is available to unauthenticated users. Valid values: • true: Unauthenticated users can access the widget. • false: Only authenticated users can access the widget. Default: false
dependencies	Array	A list of variable identifiers of *SPWidgetDependency* objects or names or sys_ids of dependencies for the widget. For more information, see SPWidgetDependency object.
angularProviders	Array	A list of variable identifiers of *SPAngularProvider* objects or names or sys_ids of Angular providers for the widget. For more information, see SPAngularProvider object.
templates	Array	A list of Angular ng-templates [sp_ng_template] to associate with the widget. Angular ng-templates contain content that is rendered only when you instruct it to render. `templates: [ { $id: Now.ID['String' or Number] id: 'String' htmlTemplate: 'HTML' or `HTML` }, ... ]` • $id: Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. Format: `Now.ID['String' or Number]` • id: Required. The ID of the Angular ng-template. • htmlTemplate: Required. The HTML template content. This property supports a reference to another file in the application that contains HTML or inline HTML.
$meta	Object	Metadata for the application metadata.

Properties (continued)

Name	Type	Description
		With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { SPWidget } from '@servicenow/sdk/core'

SPWidget({
    $id: 'my_simple_widget',
    name: 'My Simple Widget',
    category: 'knowledgeBase',
    clientScript: Now.include('./client.js'),
    serverScript: Now.include('./server.js'),
    controllerAs: '$ctrl',
    customCss: Now.include('./custom_css.css'),
    dataTable: 'sp_instance',
    demoData: { message: 'Hello, World!' },
    description: 'This is a test widget',
    docs: widgetDoc,
    htmlTemplate: Now.include('./template.html'),
    fields: ['color', 'class_name'],
    hasPreview: true,
    id: 'my-simple-widget',
    linkScript: Now.include('./link.client.js'),
    optionSchema: [
        {
            name: 'my_option',
            label: 'My Option',
            type: 'string',
            section: 'behavior'
        }
    ],
    roles: [manager, 'admin'],
})
```

SPAngularProvider object

Create an Angular Provider [sp_angular_provider] to reuse components in multiple widgets and improve portal performance.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A name for the Angular provider.
clientScript	Script	A client-side script to reuse in widgets.This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or `Script``.
type	String	The type of Angular provider. Valid values: • directive • factory • service Default: directive
requires	Array	A list of variable identifiers of other *`SPAngularProvider`* objects or names or sys_ids of Angular providers.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*:

Properties (continued)

Name	Type	Description
		• demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { SPAngularProvider } from '@servicenow/sdk/core'
const OTHER_ANGULAR_PROVIDER = 'd11f285fe069e1f119b44bd05c0770aa'

SPAngularProvider({
    $id: 'my_angular_provider',
    name: 'my_angular_provider',
    clientScript: Now.include('my_angular_provider.client.js'),
    type: 'directive',
    requires: [OTHER_ANGULAR_PROVIDER]
})
```

SPWidgetDependency object

Create a widget dependency [sp_dependency] to link JavaScript and CSS files to widgets and use third-party libraries, external style sheets, or Angular modules.

Dependencies are loaded asynchronously from the server when needed. Widgets can have as many or as few dependencies as needed. However, the more you add, the more content a widget must download to render on the page. Keep dependencies as small as possible for more efficient load times. For more information, see Create a widget dependency.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. The name of the widget dependency.
angularModuleName	String	The name of the Angular module to load if the JS include is an Angular module.
includeOnPageLoad	Boolean	Flag that indicates when the dependency loads on a page. Valid values: • true: The dependency loads with the initial page load. • false: The dependency loads only when the linked widget is loaded on a page.

Properties (continued)

Name	Type	Description
		Default: false
cssIncludes	Array	A list of variable identifiers of *`CssInclude`* objects or sys_ids of CSS includes and their order. For more information, see CssInclude object. `cssIncludes: [ { order: Number, include: 'String' or Reference, }, ... ]`
jsIncludes	Array	A list variable identifiers of *`JsInclude`* objects or sys_ids of JS includes and their order. For more information, see JsInclude object. `jsIncludes: [ { order: Number, include: 'String' or Reference, }, ... ]`
portalsForPageLoad	Array	A list of sys_ids of portals [sp_portal] that load the widget dependency. If empty, the dependency is included on the page load for all portals.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { SPWidgetDependency } from '@servicenow/sdk/core'

SPWidgetDependency({
    $id: 'samplejs',
```

```
    name: 'Sample',
    angularModuleName: 'samplejs',
    includeOnPageLoad: true,
    portalsForPageLoad:
['b4572a48262a16df3032b48cef75a853', 'fe12dbbed14bd3f712f0787141c2f656'],
    cssIncludes: [
        {
            order: 100,
            include: localCss,
        },
        {
            order: 200,
            include: '94112ccb0fb3c2ed072b01d3cb401196',
        },
    ],
    jsIncludes: [
        {
            order: 100,
            include: localJs,
        },
        {
            order: 200,
            include: 'f8af18a5e6c71a3702c4f2038b43cf62',
        },
    ],
})
```

CssInclude object

Create a CSS include [sp_css_include] to reference a style sheet or external CSS in a widget dependency.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. The name of the CSS include.
url	String	The URL to an external CSS file. This property is required if a style sheet isn't provided with the *`spCss`* property.
spCss	String	The sys_id of a style sheet [sp_css]. This property is required if an external CSS file isn't provided with the *`url`* property.
rtlCssUrl	String	The URL to an external right-to-left (RTL) CSS file for mirroring the direction of a widget when the session language is a right-to-left language, such as Hebrew. For more information, see Styling for right-to-left languages in portals.

Properties (continued)

Name	Type	Description
lazyLoad	Boolean	Flag that indicates how to load the CSS Include. This property applies only if you use the *spCSS* property to specify a style sheet. Valid values: • true: The CSS include loads asynchronously to improve page load time. • false: The CSS include doesn't load asynchronously. Default: false
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { CssInclude } from '@servicenow/sdk/core'

const localCss = CssInclude({
    $id: '22bcf16da81e2bc0340c53d50d531adf',
    name: 'Sample Styles',
    spCss: '50e3e32aa321b1c7d1945c5f423228bd',
})
```

JsInclude object

Create a JS include [sp_js_include] to reference a UI script or external JavaScript code in a widget dependency.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs.

Properties (continued)

Name	Type	Description
		Format: `Now.ID['String' or Number]`
name	String	Required. The name of the JS include.
url	String	The URL to an external JavaScript file. The URL should be an absolute path. This property is required if a UI script isn't provided with the *`sysUiScript`* property.
sysUiScript	String	The sys_id of a UI script [sys_ui_script]. This property is required if an external JavaScript file isn't provided with the *`url`* property.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { JsInclude } from '@servicenow/sdk/core'

const localJs = JsInclude({
    $id: '98239e4eadfac88b01cce7daa23b6fc3',
    name: 'Sample Framework',
    sysUiScript: 'b67af05645f738df1f286bb3e9ecd55f',
})
```

Table API - ServiceNow Fluent

The Table API defines tables [sys_db_object] to store data in an application.

Create a table using the *`Table`* object. From the *`schema`* property, add Column objects, such as *`StringColumn`* or *`IntegerColumn`*, to define the columns.

For general information about tables, see Table administration.

Related topics

ServiceNow Fluent

Table object

Create a table [sys_db_object] in an application.

Properties

Name	Type	Description
name	String	Required. A name for the table beginning with the application scope and in all lowercase letters in the following format: `<scope>_<name>`. The name should match the variable identifier of the *`Table`* object. **Note:** To add columns to an existing table in a different application scope, you can provide the name of the table without the application scope followed by `as any`. The column names must begin with the application scope instead. Maximum length: 80
schema	Array	Required. A list of *`Column`* objects. For more information, see Column object.
extends	String	The name of any other table on which the table is based. Extending a base table incorporates all the fields of the original table and creates system fields for the new table. If they are in the same scope or if they can be configured from other scopes, you can extend tables that are marked as extensible.
label	String or Array	A unique label for the table in list and form views. Field labels can be provided as a string or an array of *`label`* objects. For more information, see label object. Maximum length: 80 Default: the value of the *`name`* property
licensingConfig	Object	The licensing configuration [ua_table_licensing_config] for a table. For more information, see licensingConfig object.
display	String	The default display column. Use a column name from the *`schema`* property.
extensible	Boolean	Flag that indicates whether other tables can extend the table. Valid values: • true: Other tables can extend the table. • false: Other tables can't extend the table. Changing this property from true to false prevents the creation of additional child tables but existing child tables remain unchanged. Default: false

Properties (continued)

Name	Type	Description
liveFeed	Boolean	Flag that indicates if live feeds are available for records in the table. Valid values: • true: Live feeds are provided for records in the table. This option adds the **Show Live Feed** option () in the form header. • false: Live feeds aren't provided for records in the table. Default: false
accessibleFrom	String	The application scopes that can access the table. Valid values: • public: The table is accessible from all application scopes. • package_private: The table is accessible from only the application scope it's in. Default: public
callerAccess	String	The access level for cross-scope requests. Valid values: • restricted: Calls to the resource must be manually approved. Access requests are tracked in the Restricted Caller Access table with a status of Requested. • tracking: Calls to the resource are automatically approved. Calls are tracked in the Restricted Caller Access table with a status of Allowed. • none: Cross-scope calls to the resource are approved or denied based on the value of the `accessibleFrom` property. For more information, see Restricted caller access privilege settings. Default: none
actions	Array	A list of access options. Valid values: • read: Allow script objects from other application scopes to read records stored in this table. For example, a script in another application can query data on this table. Read access is required to grant any other API record operations.

Properties (continued)

Name	Type	Description
		• create: Allow script objects from other application scopes to create records in this table. For example, a script in another application can insert a new record in this table. • update: Allow script objects from other application scopes to modify records stored in this table. For example, a script in another application can modify a field value on this table. • delete: Allow script objects from other application scopes to delete records from this table. For example, a script in another application can remove a record from this table. Default: read
allowWebServiceAccess	Boolean	Flag that indicates whether web services can make calls to the table. Valid values: • true: Web services can make calls to the table. • false: Web services can't make calls to the table. Default: false
allowNewFields	Boolean	Flag that indicates whether to allow design time configuration of new fields on the table from other application scopes. Valid values: • true: Allow design time configuration of new fields on the table from other application scopes. • false: Don't allow design time configuration of new fields on the table from other application scopes. Default: false
allowUiActions	Boolean	Flag that indicates whether to allow design time configuration of UI actions on the table from other application scopes. Valid values: • true: Allow design time configuration of UI actions on the table from other application scopes. • false: Don't allow design time configuration of UI actions on the table from other application scopes. Default: false

Properties (continued)

Name	Type	Description
allowClientScripts	Boolean	Flag that indicates whether to allow design time configuration of client scripts on the table from other application scopes. Valid values: • true: Allow design time configuration of client scripts on the table from other application scopes. • false: Don't allow design time configuration of client scripts on the table from other application scopes. Default: false
audit	Boolean	Flag that indicates whether to track the creation, update, and deletion of all records in the table. Valid values: • true: Track the creation, update, and deletion of all records in the table • false: Don't track the creation, update, and deletion of all records in the table. Default: false
readOnly	Boolean	Flag that indicates whether users can edit fields in the table. Valid values: • true: Users can't edit fields in the table. • false: Users can edit fields in the table. Default: false
textIndex	Boolean	Flag that indicates whether search engines index the text in a table. Valid values: • true: The table's text is indexed. • false: The table's text isn't indexed. Default: false
attributes	Object	Key and value pairs of any supported dictionary attributes [sys_schema_attribute]. For example: `attributes:` `  {` `  updateSyncCustom: Boolean,` `  nativeRecordLock: Boolean` `  }`

Properties (continued)

Name	Type	Description
		For more information, see Dictionary Attributes.
index	Array	A list of column references to generate indexes in the metadata XML of the table. The value of the *element* property should match the object key used with the *Column* object. A database index increases the speed of accessing data from the table with the expense of using additional storage. `index: [ { name: 'String', element: 'String', unique: Boolean }, ... ]`
autoNumber	Object	The auto-numbering configuration [sys_number] for a table. For more information, see autoNumber object.
scriptableTable	Boolean	Flag that indicates whether the table is a remote table that uses data retrieved from an external source. For more information, see Remote tables. Valid values: • true: The table is a remote table. • false: The table isn't a remote table. Default: false

For typeahead support for columns, assign the *Table* object to an exported variable with the same name as the *name* property.

```
import { Table, StringColumn } from "@servicenow/sdk/core";
import { myFunction } from "../server/myFunction.js"

export const x_snc_example_to_do = Table({
    name: 'x_snc_example_to_do',
    label: 'My To Do Table',
    extends: 'task',
    schema: {
        status: StringColumn({ label: 'Status' }),
        deadline: StringColumn({
            label: 'Deadline',
            active: true,
            mandatory: false,
            readOnly: false,
            maxLength: 40,
            dropdown: 'none',
            attributes: {
```

```
            updateSync: false,
        },
        default: 'today',
        dynamicValueDefinitions: {
            type: 'calculated_value',
            calculatedValue: '',
        },
        choices: {
            choice1: {
                label: 'Choice1 Label',
                sequence: 0,
                inactiveOnUpdate: false,
                dependentValue: '5',
                hint: 'hint',
                inactive: false,
                language: 'en',
            },
            choice2: { label: 'Choice2 Label', sequence: 1 },
        },
    }),
    dynamic1: StringColumn({
        dynamicValueDefinitions: {
            type: 'calculated_value',
            calculatedValue: myFunction,
        },
    }),
    dynamic2: StringColumn({
        dynamicValueDefinitions: {
            type: 'dynamic_default',
            dynamicDefault: `gs.info()`,
        },
    }),
    dynamic3: StringColumn({
        dynamicValueDefinitions: {
            type: 'dependent_field',
            columnName: 'status',
        },
    }),
    dynamic4: StringColumn({
        dynamicValueDefinitions: {
            type: 'choices_from_other_table',
            table: 'sc_cat_item',
            field: 'display',
        },
    }),
},
actions: ['create', 'read'],
display: 'deadline',
accessibleFrom: 'package_private',
allowClientScripts: true,
allowNewFields: true,
allowUiActions: true,
allowWebServiceAccess: true,
extensible: true,
liveFeed: true,
callerAccess: 'none',
autoNumber: {
```

```
        number: 10,
        numberOfDigits: 2,
        prefix: 'abc',
    },
    audit: true,
    readOnly: true,
    textIndex: true,
    attributes: {
        updateSync: true,
    },
    index: [
        {
            name: 'idx',
            element: 'status',
            unique: true,
        },
    ],
})
```

Column object

Add a column [sys_dictionary] to a table.

Add *Column* objects in the *schema* property of the *Table* object.

There are many types of columns based on the field type. Column objects use the format *<Type>Column* where *<Type>* is the field type. For information about field types, see Field types.

The following types of columns are supported: *ListColumn*, *RadioColumn*, *StringColumn*, *ChoiceColumn*, *ScriptColumn*, *BooleanColumn*, *ConditionsColumn*, *DecimalColumn*, *IntegerColumn*, *VersionColumn*, *DomainIdColumn*, *FieldNameColumn*, *ReferenceColumn*, *TableNameColumn*, *UserRolesColumn*, *BasicImageColumn*, *DocumentIdColumn*, *DomainPathColumn*, *TranslatedTextColumn*, *SystemClassNameColumn*, *TranslatedFieldColumn*, *GenericColumn*, *DateColumn*, *DateTimeColumn*, *CalendarDateTime*, *BasicDateTimeColumn*, *DueDateColumn*, *CalendarDateTime*, *IntegerDateColumn*, *ScheduleDateTimeColumn*, *OtherDateColumn*, *Password2Column*, *GuidColumn*, *JsonColumn*, *NameValuePairsColumn*, *UrlColumn*, *EmailColumn*, *HTMLColumn*, *FloatColumn*, *MultiLineTextColumn*, *DurationColumn*, *TimeColumn*, *FieldListColumn*, *SlushBucketColumn*, *TemplateValueColumn*, and *ApprovalRulesColumn*.

Properties

Name	Type	Description
label	String or Array	A unique label for the column that appears on list headers and form fields. Field labels can be provided as a string or an array of *label* objects. For more information, see label object. Default: the key used for the column object
maxLength	Number	The maximum length of values in the column.

Properties (continued)

Name	Type	Description
		A length of under 254 appears as a single-line text field. Anything 255 characters or over appears as a multi-line text box. **Note:** To avoid data loss, only decrease the length of a string field when you're developing a new application and not when a field contains data. Default: varies depending on the column type
active	Boolean	Flag that indicates whether to display the field in lists and forms. Valid values: • true: Displays the field. • false: Hides the field. Default: true
mandatory	Boolean	Flag that indicates whether the field must contain a value to save a record. Valid values: • true: The field must contain a value. • false: The field isn't required. Default: false
readOnly	Boolean	Flag that indicates whether you can edit the field value. Valid values: • true: You can't change the value, and the system calculates and displays the data for the field. • false: You can change the field value. Default: false
default	Any	The default value of the field when creating a record. The value must use the correct type based on the column type.
choices	Object	A list of choices [sys_choice] for a column. For more information, see choices object. This property only applies to `ChoiceColumn` objects and column types that extend choice columns. It can include either an array of primitive values or a series of choice objects.

Properties (continued)

Name	Type	Description
attributes	Object	Key and value pairs of any supported dictionary attributes [sys_schema_attribute]. For example: `attributes: { updateSyncCustom: Boolean, nativeRecordLock: Boolean }` For more information, see Dictionary Attributes.
functionDefinition	String	The definition of a function that the field performs, such as a mathematical operation, field length computation, or day of the week calculation. Each definition begins with `glidefunction:`, followed by the operation to be performed (such as, `concat`), followed by function parameters. Constants must be enclosed in single quotes. For example, the following function definition creates a field that shows the short description, followed by a space, followed by the caller name: `functionDefinition: 'glidefunction:concat(short_description, ' ', caller_id.name)'` For more information about function definitions, see Function field.
dynamicValueDefinitions	Object	Default values that are generated dynamically based on dynamic filters. Provide a combination of a type and a related behavior key to specify dynamic defaults. The following types are supported: • dynamic_default: Provide a function from the Dynamic Filter Options [sys_filter_option_dynamic] table. For more information, see Create a dynamic filter option. For example: `dynamicValueDefinitions: { type: 'dynamic_default', dynamicDefault: `gs.info()`, },` • dependent_field: Provide another column name from the same table. For example: `dynamicValueDefinitions: { type: 'dependent_field', columnName: 'status', },` • calculated_value: Provide a function for calculating the value. The function can be imported from a JavaScript module or be defined inline. For example:

Properties (continued)

Name	Type	Description
		`` dynamicValueDefinitions: { type: 'calculated_value', calculatedValue: function, }, ``` • choices_from_other_table: Provide choices from a column on another table. For example: ``` dynamicValueDefinitions: { type: 'choices_from_other_table', table: 'sc_cat_item', field: 'display', }, ```
dropdown	String	An option for how a list of choices displays for list and form views of the table. This property only applies to *`ChoiceColumn`* objects and column types that extend choice columns. Valid values: • none: The choices aren't enforced. • dropdown_without_none: A menu without the **-- None --** option. If you select this option, you must configure the *`default`* property for the column. • dropdown_with_none: A menu with the **-- None --** option. The default value is **-- None --**. • suggestion: Choices are displayed in a list of suggested values. Default: none

Column names are provided as object keys paired with the column definitions.

```
schema: {
   deadline: DateColumn({ label: 'Deadline' }),
   state: StringColumn({
      label: 'State',
      choices: {
         ready: { label: 'Ready' },
         completed: { label: 'Completed' },
         inProgress: { label: 'In Progress' },
      }
   }),
   task: StringColumn({ label: 'Task', maxLength: 120 }),
}
```

If the table name doesn't include the application scope, column names must be prefixed with the application scope instead.

```
schema: {
   x_scope_myColumn: StringColumn({...})
}
```

choices object

Configure choices [sys_choice] for a column in a table.

The `choices` object is a property within the `Column` object. Use the `choices` object with supported column types in the `schema` property of a `Table` object. Only certain column types extend the choice column type (`ChoiceColumn`) and can include choices.

Properties

Name	Type	Description
label	String	Required. The text to display for the choice in the list.
dependentValue	String	A value that you map to the `dependentField` in the `dynamicValueDefinitions` property of the `Column` object.
hint	String	A short description of the choice that displays as tooltip when hovering over it.
language	String	The BCP 47 code of the language for the translated choice. Default: en
sequence	Integer	The order in the list of choices that a choice occurs.
inactive	Boolean	Flag that indicates whether to show the choice in the list. Valid values: • true: The choice is hidden from the list. • false: The choice appears in the list. Default: false

The `choices` object includes a series of choice objects, where the names of the choices are provided as object keys paired with the choices definitions.

```
choices: {
   choice1: {
      label: 'Choice1 Label',
      sequence: 0,
      inactiveOnUpdate: false,
      dependentValue: '5',
      hint: 'hint',
      inactive: false,
      language: 'en',
   },
   choice2: { label: 'Choice2 Label', sequence: 1 },
}
```

label object

Configure a field label [sys_documentation] for a table or column.

The *label* object is a property within the *Table* and *Column* objects.

Properties

Name	Type	Description
language	String	The BCP 47 code of the language for the field label. A language can have only one label, so each language must be unique within an array of *label* objects.
label	String	The text of the field label in the specified language.
hint	String	A short description that displays as a tooltip when hovering over the field label.
help	String	Additional information about the field. Help text isn't displayed in form or list views of the table.
plural	String	The plural form of the field label.
url	String	A URL for a web page that provides information about the field. When a URL is provided, the label displays as a hyperlink.
urlTarget	String	Not used (deprecated).

```
label: [
   {
      label: 'English description',
      language: 'en',
      hint: 'Provide a short description'
   },
   {
      label: 'Description de español',
      language: 'es'
   },
]
```

licensingConfig object

Create a licensing configuration [ua_table_licensing_config] to track subscription counts for a table.

The *licensingConfig* object is a property within the *Table* object. If this property isn't specified, a default licensing configuration with *licenseModel* set to none is generated for the table on the instance.

Note:
Specifying a licensing model is not applicable for ServiceNow customers who build custom applications for their own use. Licensing models are used only by partners who sell and monitor the usage of resellable applications on the ServiceNow Store.

Properties

Name	Type	Description
licenseModel	String	The model for tracking subscription usage.

Properties (continued)

Name	Type	Description
		Valid values: • none: Licensing isn't used for the table. • fulfiller: Fulfiller/requester operations are tracked. This model applies to applications in which users open requests and fulfillers address them. Fulfillment is determined by insert, update, and delete operations on records in one or more key tables in the application under a set of specified conditions. For more information, see Fulfillment tables. • producer: Producer operations are tracked. This model applies to applications in which users can perform insert, update, and delete operations on a table without identifying requesters and fulfillers. Default: none
licenseRoles	Array	A list of roles for which any operations on records in the table count toward the subscription.
opDelete	Boolean	Flag that indicates whether a subscription is required to delete records for tables with the producer model. Valid values: • true: A subscription is required to delete records in the table. • false: A subscription isn't required to delete records in the table. Default: true
opInsert	Boolean	Flag that indicates whether a subscription is required to insert records for tables with the producer model. Valid values: • true: A subscription is required to insert records in the table. • false: A subscription isn't required to insert records in the table. Default: true
opUpdate	Boolean	Flag that indicates whether a subscription is required to update records for tables with the producer model. • true: A subscription is required to update records in the table. • false: A subscription isn't required to update records in the table.

Properties (continued)

Name	Type	Description
		Default: true
licenseCondition	String	A filter query that determines conditions for counting operations toward a subscription. For the fulfiller model, specify the set of conditions that determine whether the logged-in user is the fulfiller of the record. For the producer model, specify the set of conditions that determine whether records count toward the subscription.
ownerCondition	String	A filter query that determines whether a user owns a record for the fulfiller model.
isFulfillment	Boolean	Not used (deprecated). Flag that indicates whether to disallow updates by users who aren't subscribed to the application. Valid values: • true: Users who aren't subscribed to the application can't make updates to the table. • false: Users who aren't subscribed to the application can make updates to the table. Default: false

```
licensingConfig: {
  licenseModel: 'fulfiller',
  opInsert: false,
  licenseRoles: ['admin'],
}
```

autoNumber object

Configure auto-numbering [sys_number] for a table.

The *autoNumber* object is a property within the *Table* object.

Properties

Name	Type	Description
prefix	String	A prefix for every record number in the table. For example, INC for Incident. Default: pre
number	Integer	The base record number for this table. Record numbers are automatically incremented, and the next number is maintained in the Counter [sys_number_counter] table.

Properties (continued)

Name	Type	Description
		If you set the base number to a value higher than the current counter, the next record number uses the new base number. Otherwise the next record number uses the current counter. The counter doesn't reset to a base number lower than itself. Default: 1000
numberOfDigits	Integer	The minimum number of digits to use after the prefix. Leading zeros are added to auto-numbers, if necessary. For example, INC0001001 contains three leading zeros. The number of digits can exceed the minimum length. For example, if *numberOfDigits* is 2 and more than 99 records are created on the table, the numbers continue past 100 (such as INC101). **Warning:** Changing this field can update all number values for existing records on a table. Take care when changing this field on a production instance. Default: 7

```
autoNumber: {
   prefix: 'TODO',
   number: 2000,
   numberOfDigits: 9,
}
```

To use the number in a table, you need to create a number column that uses the number as the default value. For example:

```
number: IntegerColumn({
   default: 'javascript:getNextObjNumberPadded();'
})
```

UI Action API - ServiceNow Fluent

The UI Action API defines custom user interface (UI) actions [sys_ui_action], such as buttons, links, and context menu items on forms and lists.

For general information about UI actions, see Create a UI action.

Related topics

ServiceNow Fluent

UiAction object

Create a UI action [sys_ui_action] to display on a form.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
table	String	Required. The name of the table on which the UI action is available. By default, the UI action also appears on tables that extend the selected table. For example, Task actions appear on the Incident table. Set the value to `global` to make the action available on all tables.
name	String	Required. The text that appears on the button, link, or context menu item. The name must be unique within the table specified.
actionName	String	A unique name to use when referencing the UI action in scripts.
active	Boolean	Flag that indicates whether the UI action is enabled. Valid values: • true: The UI action appears and executes. • false: The UI action is hidden. Default: true
form	Object	Options for how UI actions appear on forms. For more information, see form object.
list	Object	Options for how UI actions appear on the list view. For more information, see list object.
client	Object	Options to execute the script in the browser. For more information, see client object.
workspace	Object	Options for how UI actions function and appear in workspaces. For more information, see workspace object.
overrides	Reference or String	The name or variable identifier of another UI action that the UI action overrides.
showInsert	Boolean	Flag that indicates whether to show a button on new records before they're inserted. Valid values: • true: The button appears on records before they're inserted. • false: The button doesn't appear on records that haven't been inserted.

Properties (continued)

Name	Type	Description
		Default: false
showUpdate	Boolean	Flag that indicates whether to show a button on existing records. Valid values: • true: The button appears on existing records. • false: The button doesn't appear on existing records. Default: true
showQuery	Boolean	Flag that indicates whether the UI action is visible on a list when a filter query is applied. Valid values: • true: The UI action appears when a filter query is applied. • false: The UI action doesn't appear a filter query is applied. Default: false
showMultipleUpdate	Boolean	Flag that indicates whether to show a button when multiple records are selected. Valid values: • true: The button appears when multiple records are selected. • false: The button doesn't appear when multiple records are selected. Default: false
condition	String	A JavaScript conditional statement that specifies the fields and values that must be true for the script to run.

Properties (continued)

Name	Type	Description
		Note: • Don't use this property if you include the condition statement with the *`script`* property. • The current object isn't available for conditions on a list context menu. If the list's *`showContextMenu`* property is true, any use of `current` on these actions is ignored. • You can reference the parent record for the UI action conditions on a related list button. For example, to disable the **New** and **Edit** buttons on the Affected CIs related list for closed changes, copy the global m2m UI actions to the task_ci table and add a condition of *`parent.active`*. • If you leave one of the fields that you specify in your condition statement empty, that condition defaults to true. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or `Script``.
script	Script	A client-side or server-side script that runs when the UI action is executed. Function names must be unique. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or `Script``.
comments	String	Internal notes about the UI action.
messages	String	Text strings that the UI action can use as a key to look up a localized message alternative from the Message [sys_ui_message] table. Each message key is on a separate line in the Messages field. The instance looks for a localized message string anytime the UI action makes a

Properties (continued)

Name	Type	Description
		`getmessage('[message]')` call where the message string matches a key in the Messages field. For more information, see Translate a client script message.
hint	String	A short description of the UI action that displays as tooltip when hovering over it.
order	Number	The order in which the UI action appears. The order applies to buttons from left to right and to menu actions from top to bottom. Default: 100
isolateScript	Boolean	Flag that indicates whether the script runs in strict mode, with access to direct DOM, jQuery, prototype, and the window object turned off. Valid values: • true: Isolate the script and don't run it in strict mode. • false: Run the script in strict mode. Default: false
roles	Array	A list of variable identifiers of *`Role`* objects or names of roles required for the UI action to apply. For more information, see Role API - ServiceNow Fluent.
includeInViews	Array	A list of names of views in which the UI action is included.
excludeFromViews	Array	A list of names of views from which the UI action is excluded.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
import { UiAction } from '@servicenow/sdk/core'

UiAction({
    $id: Now.ID['car_info'],
    table: 'x_snc_ts_custom_cars',
    actionName: 'Car Information',
    name: 'View car info',
    active: true,
    showInsert: true,
    showUpdate: true,
    hint: 'View car info',
    condition: "current.type == 'SUV'",
    form: {
        showButton: true,
        showLink: true,
        showContextMenu: false,
        style: 'destructive',
    },
    list: {
        showLink: true,
        style: 'primary',
        showButton: true,
        showContextMenu: false,
        showListChoice: false,
        showBannerButton: true,
        showSaveWithFormButton: true,
    },
    workspace: {
        isConfigurableWorkspace: true,
        showFormButtonV2: true,
        showFormMenuButtonV2: true,
        clientScriptV2: `function onClick(g_form) {
                        }`,
    },
    script: `current.name =  "updated by script";
                current.update();`,
    roles: ['u_requestor'],
    client: {
        isClient: true,
        isUi11Compatible: true,
        isUi16Compatible: true,
        },
    order: 100,
    showQuery: false,
    showMultipleUpdate: false,
    isolateScript: false,
    includeInViews: ['specialView'],
    excludeFromViews: [],
})
```

form object

Configure how a UI action appears on a form.

The *form* object is a property within the *UiAction* object.

Properties

Name	Type	Description
showButton	Boolean	Flag that indicates whether to include a button on a form. Valid values: • true: A button appears on forms. • false: A button doesn't appear on forms. Default: false
showLink	Boolean	Flag that indicates whether to include a link in the Related Links section of a form. Valid values: • true: A link appears in the Related Links section. • false: A link doesn't appear in the Related Links section. Default: false
showContextMenu	Boolean	Flag that indicates whether to include an item in the context menu of a form. Valid values: • true: An item appears in the context menu. • false: An item doesn't appear in the context menu. Default: false
style	String	A style that defines how UI action buttons appear on a form. Valid values: • primary: Colors the UI action blue. • destructive: Colors the UI action red. • unstyled: The UI action isn’t styled.
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*:

Properties (continued)

Name	Type	Description
		• demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
form: {
  showButton: true,
  showLink: true,
  showContextMenu: false,
  style: 'destructive',
}
```

list object

Configure how a UI action appears on the list view.

The *`list`* object is a property within the *`UiAction`* object.

Properties

Name	Type	Description
showButton	Boolean	Flag that indicates whether to include a button at the bottom of a list. **Note:** Buttons at the bottom of a list appear regardless of condition and are evaluated per record on execution. Valid values: • true: A button appears at the bottom of lists. • false: A button doesn't appear at the bottom of lists. Default: false
showLink	Boolean	Flag that indicates whether to include a link in the Related Links section of a list. Valid values: • true: A link appears in the Related Links section. • false: A link doesn't appear in the Related Links section. Default: false
showContextMenu	Boolean	Flag that indicates whether to include an item in the context menu of a list.

Properties (continued)

Name	Type	Description
		Valid values: • true: An item appears in the context menu. • false: An item doesn't appear in the context menu. Default: false
style	String	A style that defines how UI action buttons appear on the list view. Valid values: • primary: Colors the UI action blue. • destructive: Colors the UI action red. • unstyled: The UI action isn't styled.
showListChoice	Boolean	Flag that indicates whether to include a choice in the Actions list of a list. **Note:** Choices in the Actions list appear regardless of condition and are evaluated per record on execution. Valid values: • true: A choice appears in the Actions list. • false: A choice doesn't appear in the Actions list. Default: false
showBannerButton	Boolean	Flag that indicates whether to include a button on the banner of a list. **Note:** Buttons on the banner of a list aren't intended to support record-specific conditions. Only the first row is considered when the condition is evaluated to determine whether the button appears for the list. Don't use record-specific conditions, such as `current.getValue('state') === 'closed'`. Valid values: • true: A button appears on the banner of lists. • false: A button doesn't appear on the banner of lists. Default: false
showSaveWithFormButton	Boolean	Flag that indicates whether the form is saved when accessed from a list before executing the UI action button.

Properties (continued)

Name	Type	Description
		Valid values: • true: The form must be saved before the UI action executes. • false: The form doesn't have to be saved before the UI action executes. Default: false
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
list: {
  showButton: true,
  showLink: true,
  showContextMenu: false,
  style: 'primary',
  showListChoice: false,
  showBannerButton: true,
  showSaveWithFormButton: true,
}
```

client object

Configure options to execute the UI action script in the browser.

The *client* object is a property within the *UiAction* object.

Properties

Name	Type	Description
isClient	Boolean	Flag that indicates where the UI action script runs. Valid values:

Properties (continued)

Name	Type	Description
		• true: The script runs on the client (the user's browser). • false: The script runs on the server. Default: false
isUi11Compatible	Boolean	Flag that indicates whether the UI action is supported in the legacy UI 11. Valid values: • true: The UI action is supported in the legacy UI 11. • false: The UI action isn't supported in the legacy UI 11. Default: false
isUi16Compatible	Boolean	Flag that indicates whether the UI action is supported in the Core UI. Valid values: • true: The UI action is supported in the Core UI. • false: The UI action isn't supported in the Core UI. Default: false
onClick	String	The name of the JavaScript function to run when the UI action is executed. The function is defined with the *`script`* property.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
client: {
  isClient: true,
  isUi11Compatible: true,
  isUi16Compatible: true,
```

```
  onClick: 'reopenIncident()'
}
```

workspace object

Configure how a UI action functions and appears in workspaces.

The *workspace* object is a property within the *UiAction* object.

Properties

Name	Type	Description
isConfigurableWorkspace	Boolean	Flag that indicates the type of workspace in which a UI action applies. Valid values: • true: The UI action applies to Configurable Workspaces. • false: The UI action applies to Legacy Workspaces. Default: false
showFormButtonV2	Boolean	Flag that indicates whether to include a button on forms in a workspace. Valid values: • true: A button appears on forms. • false: A button doesn't appear on forms. Default: false
showFormMenuButtonV2	Boolean	Flag that indicates whether to include an item in the More Actions menu in a workspace. Valid values: • true: An item appears in the More Actions menu. • false: An item doesn't appear in the More Actions menu. Default: false
clientScriptV2	String	A script that runs when the UI action is executed in workspaces. This property supports inline JavaScript or a reference to another file in the application that contains a script. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more

Properties (continued)

Name	Type	Description
		information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or `Script``.
$meta	Object	Metadata for the application metadata. With the *installMethod* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *installMethod*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

```
workspace: {
  isConfigurableWorkspace: true,
  showFormButtonV2: true,
  showFormMenuButtonV2: true,
  clientScriptV2: `function onClick(g_form) {
  }`,
}
```

UI Page API - ServiceNow Fluent

The UI Page API defines custom user interface (UI) pages [sys_ui_page] that display forms, dialogs, lists, and other UI components.

A UI page displays as a web page and can be added to a widget for use in dashboards. For general information about UI pages, see UI pages.

You can develop a simple React application with the UI Page API. In the `src/client` directory, add static content files that define the HTML, client script, and styling of the page. From the *UiPage* object, refer to the page's HTML entry point (`index.html`). For more information, see User interface development with React.

Related topics

ServiceNow Fluent

UiPage object

Create a UI page [sys_ui_page] for a custom user interface.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
endpoint	String	Required. The endpoint to access the web page. The endpoint value can't contain spaces. Format: `<scope>_<ui_page_name>.do`
description	String	A description of the user interface and its purpose.
direct	Boolean	Flag that indicates whether to omit the standard HTML, CSS, and JavaScript for a UI page. For React UI pages, set this property to `true`. Valid values: • true: Omit the standard HTML, CSS, and JavaScript and provide custom CSS and JavaScript for the page. • false: Include the standard HTML, CSS, and JavaScript. Default: false
html	String	The body HTML code that defines what is rendered when the page is shown. It can contain either static XHTML, dynamically generated content defined as Jelly, or call script includes and UI Macros. This property supports an alias to import an `index.html` file for React development, a reference to another file in the application that contains HTML, or inline JavaScript. **Note:** With React development and imported HTML, the *`html`* property of the *`UiPage`* object supports only one-way synchronization. After defining the HTML of a UI page in source code, if the HTML is modified outside of the source code, those changes aren't synchronized and reflected in the source code. Format: • An alias to import the `index.html` file from the previous build for UI development with React. The alias can be any value and must also be imported in the `now.ts` file that contains the *`UiPage`* object: `import <alias> from 'path/to/index.html'`. • To use text content from another file, refer to a file in the application using the following format:

Properties (continued)

Name	Type	Description
		`Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide inline HTML, use string literals or template literals for multiple lines of code: `'HTML' or` `` `HTML` ``.
category	String	The type of UI page. Valid values: • general: The page is general purpose. • homepages: The page is used as a home page. • htmleditor: The page is used to insert HTML content. • kb: The page is used with a Knowledge Base. • cms: The page is used with the Content Management System (CMS). • catalog: The page is used with Service Catalog.
clientScript	Script	A script that runs in the browser, such as functions called by buttons. This script handles any required client-side processing needed, like setting focus to a field or other interactive DHTML features after a page is loaded. This property supports inline JavaScript or a reference to another file in the application that contains a script. Client scripts for the UI page are deployed to the browser within a `<script>` tag, so the content can similarly be defined within the HTML field. The *`clientScript`* property can be used instead to define these scripts concisely to maintain the Jelly and HTML manageability. Format: • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script' or` `` `Script` ``.
processingScript	Script	A script that runs on the server when the page is submitted, which is useful if your page has a form defined with the `<g:ui_form/>` or `<g:form/>` tags. This property supports a function from a JavaScript module, a reference to another file in the application that contains a script, or inline JavaScript. Format:

Properties (continued)

Name	Type	Description
		• For functions, use the name of a function, function expression, or default function exported from a JavaScript module and import it into the `.now.ts` file. For information about JavaScript modules, see JavaScript modules and third-party libraries. • To use text content from another file, refer to a file in the application using the following format: `Now.include('path/to/file')`. For more information, see ServiceNow Fluent language constructs. • To provide an inline script, use string literals or template literals for multiple lines of code: `'Script'` or `` `Script` ``.
$meta	Object	Metadata for the application metadata. With the *`installMethod`* property, you can map the application metadata to an output directory that loads only in specific circumstances. `$meta: { installMethod: 'String' }` Valid values for *`installMethod`*: • demo: Outputs the application metadata to the `metadata/unload.demo` directory to be installed with the application when the **Load demo data** option is selected. • first install: Outputs the application metadata to the `metadata/unload` directory to be installed only the first time an application is installed on an instance.

In this example of UI development with React, the HTML of the page is imported from the `index.html` file in the `src/client` directory.

```
//incident-manager.now.ts
import '@servicenow/sdk/global'
import { UiPage } from '@servicenow/sdk/core'
import incidentPage from '../../client/index.html'

UiPage({
    $id: Now.ID['incident-manager-page'],
    endpoint: '<scope>_incident_manager.do',
    description: 'Incident Response Manager UI Page',
    category: 'general',
    html: incidentPage,
    direct: true,
})
```

In the `index.html` file, the `<script>` tag refers to the `main.jsx` file in the `src/client` directory, which contains the component code.

```
//index.html
<html>
<head>
  <title>Incident Response Manager</title>

  <!-- Initialize globals and Include ServiceNow's required scripts -->
  <sdk:now-ux-globals></sdk:now-ux-globals>

  <!-- Include your React entry point -->
  <script src="./main.jsx" type="module"></script>
</head>
<body>
  <div id="root"></div>
</body>
</html>
```

UI Policy API - ServiceNow Fluent

The UI Policy API defines user interface (UI) policies [sys_ui_policy] that dynamically change the behavior of information on a form and control custom process flows for tasks.

UI Policies can make fields mandatory, read-only, visible, hidden, or cleared when certain conditions are met. You can also use client scripts to perform all of these actions, but for faster load times use UI policies when possible.

For general information about UI policies, see UI policies.

Related topics

ServiceNow Fluent

UiPolicy object

Create a UI policy sys_ui_policy to configure form behavior.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
table	String	Required. The table of the form to modify.
shortDescription	String	Required. A description of the policy and its purpose.
active	Boolean	Flag that indicates whether the policy is applied to the form. Default: true
global	Boolean	Flag that indicates whether to which form views the policy applies. If true, the policy applies to all views of the table. If false, the policy is specific to the form view specified with the *`view`* property. Default: true

Properties (continued)

Name	Type	Description
onLoad	Boolean	Flag that indicates when the policy executes. If true, the policy runs every time a form is loaded if the conditions are satisfied. Default: true
reverseIfFalse	Boolean	Flag that indicates whether to invert the policy behavior when the condition evaluates to false. If true, the policy action is undone when the conditions of its policy evaluate to false. Default: true
inherit	Boolean	Flag that indicates whether tables that extend the current table inherit the policy. If true, extended tables inherit the policy. When a child table has an inherited policy from its parent table, the policy on the child table runs first. This event is true regardless of the order of the policies. Default: false
isolateScript	Boolean	Flag that indicates whether to run scripts in isolated scope. If true, the script runs in isolated scope. This property only applies if the *runScripts* is set to true. Default: false
conditions	String	A filter query that specifies the fields and values that must be true for users to access the object. For more information, see Operators available for filters and queries. To set conditions using a script, use a client script instead. Conditions are only rechecked if a user manually changes a field on a form. If the change is made by a UI action, context menu action, or through the list editor, it isn't evaluated.
runScripts	Boolean	Flag that indicates whether advanced behavior can be scripted for both true and false conditions. If true, scripts defined with the *scriptTrue*, *scriptFalse*, *uiType*, and *isolateScript* properties run when applicable. Default: false
scriptTrue	String	Client-side script that runs if the conditions of the policy are met. This property is required if the *runScripts* property is set to true. Format: `function onCondition() {}` Default: `function onCondition() {\n\n}`
scriptFalse	String	Client-side script that runs if the conditions of the policy aren't met and the *reverseIfFalse* property is set to true. This property is required if the *runScripts* property is set to true. Format: `function onCondition() {}`

Properties (continued)

Name	Type	Description
		Default: `function onCondition() {\n\n}`
uiType	String	The type of user interface to which the policy applies. This property is required if the *`runScripts`* property is set to true. Valid values: • desktop: The policy applies to desktop interfaces. • mobile-or-service-portal: The policy applies to mobile and Service Portal interfaces. • all: The policy applies to all interfaces. Default: desktop
actions	Array	A list of field actions to apply if the conditions are met. For more information, see actions array.
relatedListActions	Array	A list of visibility controls for related lists. For more information, see relatedListActions array.
description	String	Additional information about the policy.
modelId	String	The sys_id of the parent UI policy to which the policy applies when this policy is inherited. This property works in conjunction with *`model_table`* property.
modelTable	String	The name of the parent table to which the policy applies when a UI policy is inherited from a parent table. This property works in conjunction with the *`model_id`* property.
order	Number	The execution order in which to apply policies if more than one policy fits the conditions. For inherited UI policies, the extended table's policies are executed first. Then the base table policies are executed. Default: 100
setValues	String	(Deprecated) The field values to set using an encoded string format. Use UI policy actions [sys_ui_policy_action] instead.
view	Reference or String	Required. The variable identifier or name of the UI view [sys_ui_view] which applies, or the default view. To define a UI view, use the Record API - ServiceNow Fluent. To use the default view (`default_view`), you must import it: `import { default_view } from '@servicenow/sdk/core'`

```
import { UiPolicy } from '@servicenow/sdk/core';

export const securityIncidentPolicy = UiPolicy({
  $id: Now.ID['security_incident_policy'],
  table: 'incident',
  shortDescription: 'Lock critical fields for security incidents',
```

```
  active: true,
  onLoad: true,
  conditions: 'category="security"',
  actions: [
    {
      field: 'security_notes',
      mandatory: true,
      visible: true
    },
    {
      field: 'caller_id',
      readOnly: true   // Lock caller field when category is security
    },
    {
      field: 'assignment_group',
      readOnly: true   // Lock assignment group when category is security
    },
    {
      field: 'priority',
      readOnly: true   // Lock priority when category is security
    }
  ],
  relatedListActions: [
    {
      // Using plain GUID for system relationships
      list: 'b9edf0ca0a0a0b010035de2d6b579a03',  // Attachments
      visible: false
    },
    {
      // Using table.field format for reference fields
      list: 'x_snc_17sepapp1_expenseitem.expensereport',  // Example reference field
      visible: true
    }
  ]
})
```

actions array

Configure the actions [sys_ui_policy_action] that the UI policy performs on fields.

Use the *`actions`* array within the *`UiPolicy`* object. Actions are processed in the order that they appear in the array. At least one of the `visible`, `readOnly`, `mandatory`, or `cleared` properties must be specified for each action in the array.

Properties

Name	Type	Description
field	String	Required. The name of the field to which the action applies. **Note:** If the specified field isn't found on the form, the UI policy performs the action on the variable with the same name.
visible	Boolean or String	An option to control the visibility of the field. Valid values:

Properties (continued)

Name	Type	Description
		• true: The field is visible. • false: The field is hidden. • 'ignore': The field isn't changed. Default: ignore
readOnly	Boolean or String	An option to control access to edit the field. Valid values: • true: The field is read-only. • false: The field is editable. • 'ignore': The field isn't changed. Default: ignore
mandatory	Boolean or String	An option to control whether the field is required. Valid values: • true: The field is required. • false: The field is optional. • 'ignore': The field isn't changed. Default: ignore
cleared	Boolean	Flag that indicates whether the field should be cleared if the conditions of the policy are met. Default: false
table	String	The table to which the action applies, which overrides the table specified by the policy. If empty, the table specified by the policy applies.
value	String	The value to set the field to if the policy conditions are met.
fieldMessage	String	A message to display about the field if the policy conditions are met.
fieldMessageType	String	A message type that determines how the field message is presented. Valid values: • error • info • warning • none Default: none

Properties (continued)

Name	Type	Description
valueAction	String	An action to perform on the field value. Valid values: • set_value • clear_value • ignore Default: ignore

```
actions: [
        {
            field: 'assignment_group',
            mandatory: true,
            value: 'Critical Response Team',  // Set default value
            fieldMessage: 'This incident requires immediate attention from the
Critical Response Team',
            fieldMessageType: 'error'  // Show as error message
        },
        {
            field: 'urgency',
            value: '1',  // Set to High urgency
            fieldMessage: 'Urgency has been automatically set to High',
            fieldMessageType: 'info'
        },
        {
            field: 'impact',
            value: '1',  // Set to High impact
            fieldMessage: 'Impact has been automatically set to High',
            fieldMessageType: 'warning'
        }
    ]
```

relatedListActions array

Configure the visibility of related lists [sys_ui_policy_rl_action] on a form for a UI policy.

Use the *relatedListActions* array within the *UiPolicy* object. Related list actions are processed in the order that they appear in the array. Either the *list* or *visible* property must be specified for each related list action in the array.

Properties

Name	Type	Description
list	String	A reference to a related list on the form. If empty, the action applies to all related lists. This property is required if the *visible* property is not set. Format:

Properties (continued)

Name	Type	Description
		• sys_ID: The sys_ID for the list. Use this format for custom queries, attachments, and system-defined related lists. • table.field: The table and field name associated for the list, such as `'incident.caller_id'`. Use this format for reference field-based relationships.
visible	Boolean or String	An option to control the visibility of the related list. This property is required if the *`list`* property is not set. Valid values: • true: The related list is visible. • false: The related list is hidden. • 'ignore': The related list isn't changed. Default: ignore

```
relatedListActions: [
    {
      // Using plain GUID for system relationships
      list: 'b9edf0ca0a0a0b010035de2d6b579a03',  // Attachments
      visible: false
    },
    {
      // Using table.field format for reference fields
      list: 'x_snc_17sepapp1_expenseitem.expensereport',  // Example reference field
      visible: true
    }
  ]
```

Workspace API - ServiceNow Fluent

The Workspace API defines configurable workspace experiences for organizing and sharing data visually.

The Workspace API creates application metadata in the following tables depending on the workspace definition: UX Application [sys_ux_page_registry], UX App Configuration [sys_ux_app_config], UX Application Category M2M [sys_ux_registry_m2m_category], UX Page Property [sys_ux_page_property], UX Screen Collection [sys_ux_screen_type], UX App Route [sys_ux_app_route], UX Screen [sys_ux_screen], and UX Macroponent Definition [sys_ux_macroponent].

Dashboards can be used as the home page of a workspace by referring to one or more workspaces from the *`visibilities`* array of the *`Dashboard`* object. To create a dashboard, see Dashboard API - ServiceNow Fluent.

For general information about workspaces, see Configurable Workspace UI.

Related topics

ServiceNow Fluent

Workspace object

Create a workspace for managing business entities in a single focused working area that enables users to complete an entire job.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
title	String	Required. A name for the workspace that appears in navigation and headers.
path	String	Required. The URL path segment of the workspace. Workspace URLs follow the pattern `/now/<path>/<landingPath>` and use kebab case. Workspaces require access control lists (ACLs) to secure the workspace routes. The *`field`* property of an *`Acl`* object must match the value of this property with a wildcard pattern: `{workspace.path}.*`.
tables	Array	Required. A list of table names to manage in the workspace.
listConfig	Reference	Required. The variable identifier of a *`UxListMenuConfig`* object that defines the navigation structure of the workspace. For more information, see UxListMenuConfig object.
landingPath	String	The URL path segment of the landing page. Workspace URLs follow the pattern `/now/<path>/<landingPath>` and use kebab case. Default: home
active	Boolean	Flag that indicates whether the workspace is accessible to users. Default: true

```
import { Workspace } from '@servicenow/sdk/core';

const itsmWorkspace = Workspace({
    $id: Now.ID['itsm_workspace'],
    title: 'IT Service Management',
    path: 'itsm',
    tables: ['incident', 'problem', 'change_request', 'user', 'sys_user_group'],
    listConfig: incidentListConfig
})
```

The UX List Menu Configuration referenced is defined using the *`UxListMenuConfig`* object.

UxListMenuConfig object

Define a UX list menu configuration [sys_ux_list_menu_config] for the navigation structure and list views of a workspace.

A UX list menu configuration organizes data into categories and lists, enabling users to access different views of business data with filtering, column selection, and role-based visibility. This structure appears in the workspace's navigation panel, providing organized access to different data views.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A name for the list configuration.
description	String	A description of the list configuration.
active	Boolean	Flag that indicates whether the list configuration is active. Default: true
categories	Array	A list of top-level groupings in the list configuration. For more information, see categories array.

```
import { UxListMenuConfig } from '@servicenow/sdk/core';

const incidentListConfig = UxListMenuConfig({
    $id: Now.ID['incident_list_config'],
    name: 'Incident List Configuration',
    description: 'Navigation for Incident Workspace',
    categories: [
        {
            $id: Now.ID['incidents_category'],
            title: 'Incidents',
            order: 10,
            lists: [
                {
                    $id: Now.ID['incidents_open'],
                    title: 'Open',
                    order: 10,
                    condition: 'active=true^EQ',
                    table: 'incident',
                    columns: 'number,short_description,priority,state',
                    applicabilities: [
                        {
                            $id: Now.ID['incidents_open_applicability'],
                            applicability: applicability
                        }
                    ]
                }
            ]
        }
    ]
})
```

categories array

Define categories of related lists [sys_ux_list_category] for a UX list menu configuration.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
title	String	Required. A title for the category to display in the navigation menu.
lists	Array	Required. A list of list views in the category. For more information, see lists array.
order	Number	A number indicating the position of the category in the navigation menu. Categories with lower numbers appear first.
active	Boolean	Flag that indicates whether the category is visible in the navigation menu. Default: true
description	String	A description of the category.

```
categories: [
    {
      $id: Now.ID["incidents_category"],
      title: "Incidents",
      order: 10,
      lists: [
        {
          $id: Now.ID["incidents_open"],
          title: "Open",
          order: 10,
          condition: "active=true^EQ",
          table: "incident",
          columns: "number,short_description,priority,state",
          applicabilities: [
            {
              $id: Now.ID["incidents_open_applicability"],
              applicability: applicability
            }
          ]
        },
        {
          $id: Now.ID["incidents_all"],
          title: "All",
          order: 20,
          condition: "",
          table: "incident",
          columns: "number,short_description,priority,state",
          applicabilities: [
            {
```

```
            $id: Now.ID["incidents_all_applicability"],
            applicability: applicability
          }
        ]
      }
]
```

lists array

Define list views of table data [sys_ux_list] with filtering and column configurations for a UX list menu configuration.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
title	String	Required. A title for the list to display in the navigation menu.
table	String	Required. The name of a table to use for the list.
columns	String	A comma-separated list of column names to display in the list.
condition	String	An encoded query string to filter the records displayed in the list.
order	Number	A number indicating the position of the list within its category. Lists with lower numbers appear first.
active	Boolean	Flag that indicates whether the list is visible to users. Default: true
applicabilities	Array	A list of variable identifiers of *`Applicability`* objects that control which roles can view the list. For more information, see Applicability object.

```
lists: [
        {
          $id: Now.ID["assets_active"],
          title: "Active",
          order: 10,
          condition: "install_status=1",
          table: "alm_asset",
          columns: "asset_tag,display_name,model_category,assigned_to",
          applicabilities: [
            {
              $id: Now.ID["assets_active_applicability"],
              applicability: assetApplicability
            }
          ]
        },
        {
          $id: Now.ID["assets_all"],
          title: "All",
```

```
            order: 20,
            condition: "",
            table: "alm_asset",
            columns: "asset_tag,display_name,model_category,assigned_to",
            applicabilities: [
              {
                $id: Now.ID["assets_all_applicability"],
                applicability: assetApplicability
              }
            ]
  }
]
```

Applicability object

Define the audience [sys_ux_applicability] that can view a list in the UX list menu configuration.

Properties

Name	Type	Description
$id	String or Number	Required. A unique ID for the metadata object. When you build the application, this ID is hashed into a unique sys_id. For more information, see ServiceNow Fluent language constructs. Format: `Now.ID['String' or Number]`
name	String	Required. A name for the applicability rule.
description	String	A description of the audience.
active	Boolean	Flag that indicates whether the applicability rule is applied. Default: true
roles	Array	A list of variable identifiers of *`Role`* objects or sys_ids of roles that a user must have to view the list. For more information, see Role API - ServiceNow Fluent.
roleNames	String	A comma-separated list of role names that a user must have to view the list. This property is an alternative to the *`roles`* property.

```
import { Applicability } from '@servicenow/sdk/core';

const managerApplicability = Applicability({
    $id: Now.ID['manager_applicability'],
    name: 'Managers Only',
    roles: [managerRole]
})
```

The role referenced is defined using the *`Role`* object:

```
import { Role } from '@servicenow/sdk/core';

const managerRole = Role({
    $id: Now.ID['manager_user_role'],
    name: 'x_snc_manager.user',
```

```
    containsRoles: ['canvas_user']
})
```

Guided Application Creator

Guided Application Creator is an intuitive development interface for building applications on the ServiceNow AI Platform®. It provides a step-by-step process to guide you through your initial application construction.

Important:
You've landed at a legacy app creation tool that will be supported until the Australia release in Q1 2026. Try building and editing apps in the new version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Explains how to use Guided Application Creator to create a new application for workspace, mobile, and classic with an imported spreadsheet.

Watch this four-minute video to learn about getting started with Guided Application Creator.

Note:
Although the video describes creating a workspace in Guided Application Creator, this feature is no longer supported. Instead, you can create a workspace using App Engine Studio. For more information, see Building applications in App Engine Studio.

Using Guided Application Creator

Setting up an application in Guided Application Creator involves:

- Creating an application record
- Defining user roles
- Designating data tables
- Designing the application for different user experiences

You can also use Guided Application Creator to create an application record and then configure it in Studio later. For more information, see Create an application record in Guided Application Creator.

User experiences

When you set up an application in Guided Application Creator, you can make your application available in the following user experiences.

User experience	Description
Mobile	ServiceNow Agent mobile app. Select this user experience to support users who work from a mobile device.
Classic	Classic ServiceNow AI Platform experience. Select this user experience to let your users work on your application via lists and forms.

Guided Application Creator and the application creation process

Setting up an application in Guided Application Creator does not capture the entire process of creating an application. It is just one step of the process. Before you set up an application

in Guided Application Creator, plan what kind of application you need for your business requirements before you build it.

After you set up an application in Guided Application Creator, you can further develop the application using ServiceNow Studio, Flows in Workflow Studio, and Team Development. After the application is fully built, you can test the application and then share the application with your users.

For more information on the application creation process, see Building applications.

Activation information

Guided Application Creator is enabled via the Guided Application Creator (com.glide.sn-guided-app-creator) plugin, which is active by default in the ServiceNow AI Platform.

To work in Guided Application Creator, use a different browser. For more detail on Internet Explorer 11 support, see KB0683275.

Setting up an application in Guided Application Creator

Set up an application in Guided Application Creator to store information and manage business processes.

Shows how to create a no-code application on the ServiceNow platform quickly and easily through Guided Application Creator.

Note:
Although the video describes creating a workspace in Guided Application Creator, this feature is no longer supported. Instead, you can create a workspace using App Engine Studio. For more information, see Building applications in App Engine Studio.

Requirements

Plan your application before you build it. Define the business requirements of your organization, and then identify what information you want the application to track. For more information on the application development process, see Building applications.

To work in Guided Application Creator, use a different browser. For more detail on Internet Explorer 11 support, see KB0683275.

You must have the sn_g_app_creator.app_creator or admin role to access Guided Application Creator.

What to do

1. Create an application record in Guided Application Creator.

   Create an application record in Guided Application Creator to identify a custom application.

2. Define roles in Guided Application Creator.

   Create or select roles in Guided Application Creator for the members of your organization who use your custom application.

3. Select user experiences in Guided Application Creator.

   Let users access your application via the ServiceNow Agent mobile app or lists and forms.

4. Designate data tables in Guided Application Creator.

Select an existing table or create a custom table in Guided Application Creator to store data for your custom business application.

5. Customize user experiences in Guided Application Creator.

   Customize how the application appears in each user experience that you select.

Tip:
Complete the entire process of setting up an application in Guided Application Creator. If you exit, you have to continue setting up your application in Studio and across various forms. You can't pick up where you left off in Guided Application Creator.

Next steps

Although your application is ready to use, you may need to customize your application to fit the needs of your organization.

In ServiceNow Studio, configure:

- System notifications
- Form layouts
- UI policy
- Reports

You can also learn how to automate your processes and create business logic for your application in Flows in Workflow Studio.

Create an application record in Guided Application Creator

Create an application record in Guided Application Creator to identify a custom application.

Before you begin

By default, developers can't create applications in the global scope in Guided Application Creator. You can limit global application development to certain developers by assigning them an additional role. For more information, see Allow global application development in Guided Application Creator.

Role required: sn_g_app_creator.app_creator or admin

Procedure

1. Navigate to **All** > **System Applications** > **Studio**, and then select **Create Application**.
   If you are launching Guided Application Creator for the first time, select **Let's get started** on the welcome screen.
2. On the screen, fill in the fields.

Field	Description
Name	Name of your application.
Description	Description of your application.
Scope	Scope of your application. The application scope is set automatically when you name your application.

Field	Description
	If available, you can also select to create your application in the global scope. For more information on application scopes, see Application scope.

3. **Optional:** Give your application a logo image.
   You can drag and drop an image on the logo field, or you can select **Drag and drop or browse to upload logo**.

4. Select **Create** to assign roles for your application.

   To continue building your application in Guided Application Creator, follow the steps in Define roles in Guided Application Creator.

   You can optionally exit Guided Application Creator on the **Let's create some roles for this app** screen to save the application and add more functions to it later. To exit, select **X** to close the screen and then select **Yes, close**.

   Tip:
   Complete the entire process of setting up an application in Guided Application Creator. If you exit, you have to continue setting up your application in Studio and across various forms. You can't pick up where you left off in Guided Application Creator.

Result

Your application record is saved in the Custom Application [sys_app] table. You are added as a delegated developer for the application. To view your application later, navigate to **System Applications** > **My Company Applications**.

Define roles in Guided Application Creator

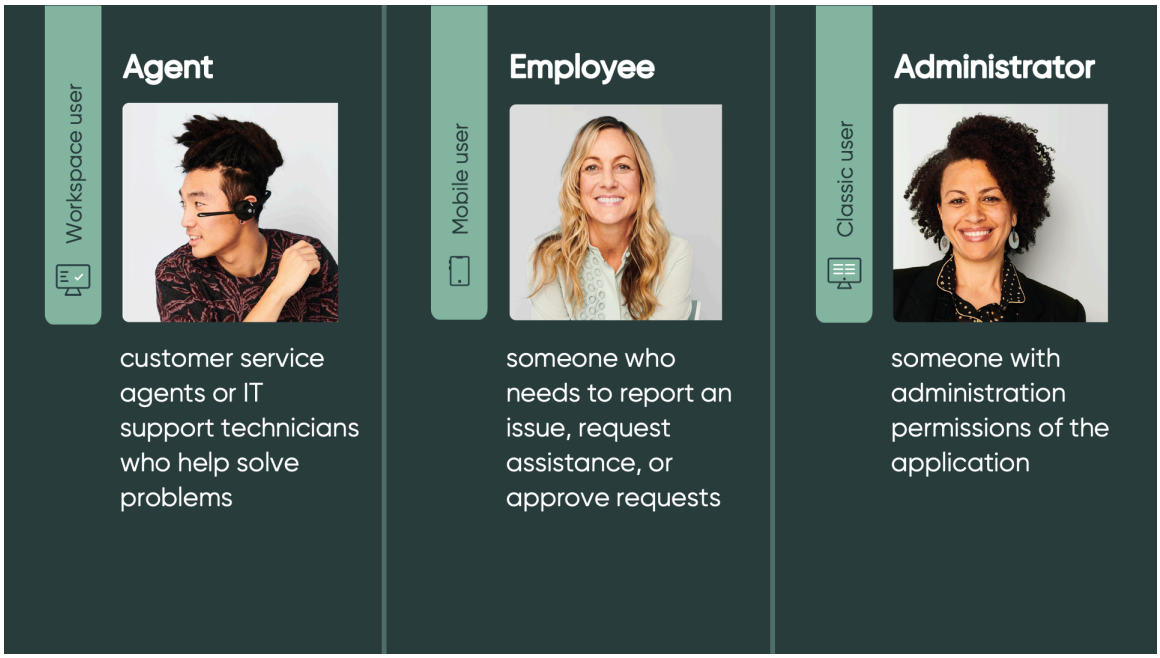
Create or select roles in Guided Application Creator for the members of your organization who use your custom application.

Before you begin

Complete the steps in Create an application record in Guided Application Creator.

Role required: sn_g_app_creator.app_creator or admin

About this task



Procedure

1. Define roles for your application.

   You can select existing roles or create roles.

2. Select **Continue**.

   You are taken to the screen in which you select user experiences for your application.

   Note:
   If you created any roles, you must select **Continue** to save the roles to your instance. If you exit Guided Application Creator before selecting **Continue**, the roles that you created are not saved.

What to do next

Continue building your application by following the steps in Select user experiences in Guided Application Creator.

Select user experiences in Guided Application Creator

Let users access your application via the ServiceNow Agent mobile app or lists and forms.

Before you begin

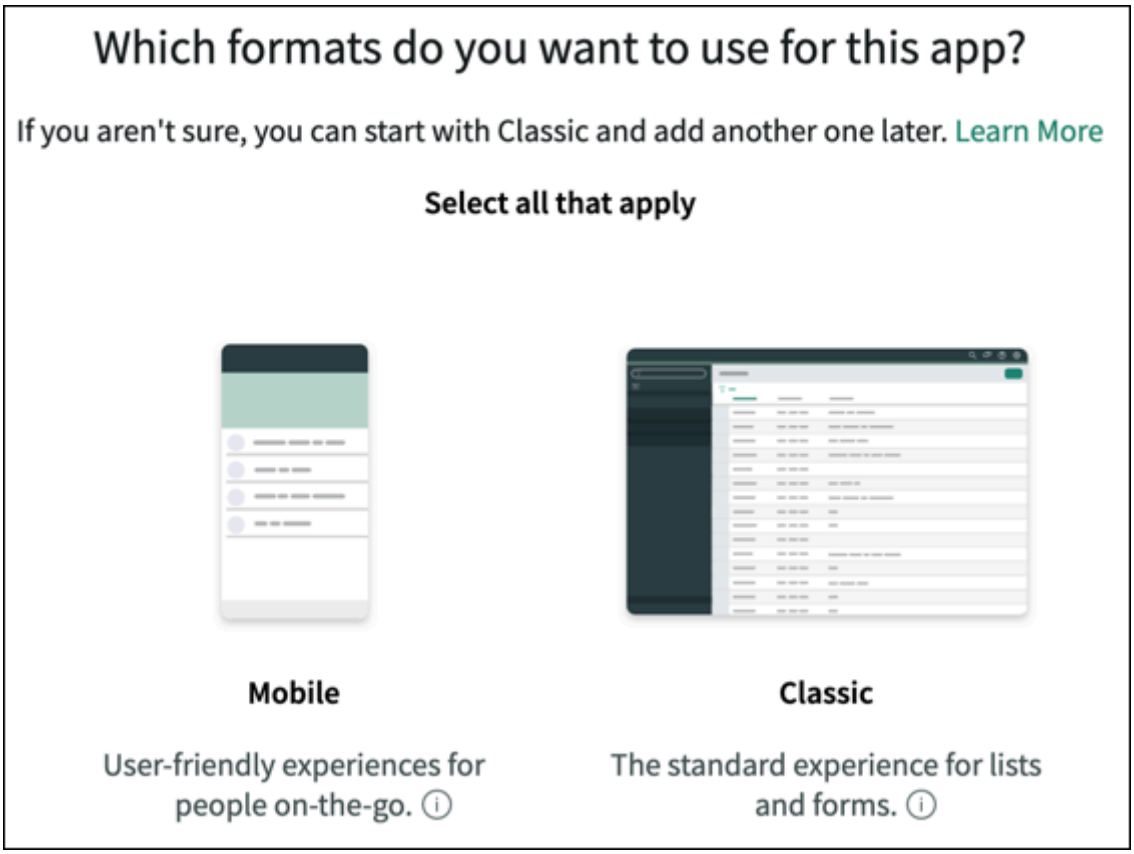
Complete:

1. Create an application record in Guided Application Creator
2. Define roles in Guided Application Creator

Role required: sn_g_app_creator.app_creator or admin

Procedure

1. Select user experiences in which to make your application available.



User experience	Description
Mobile	ServiceNow Agent mobile app. Select this user experience to support users who work from a mobile device.
Classic	Classic ServiceNow AI Platform experience. Select this user experience to let your users work on your application via lists and forms.

There is no option to create a workspace in Guided Application Creator. Instead, you can create a workspace using App Engine Studio. For more information, see Building applications in App Engine Studio.

2. Select **Continue** to designate a data table for your application.

What to do next

Continue building your application by following the steps in Designate data tables in Guided Application Creator. If you exit Guided Application Creator, the user experiences that you selected are not saved to your application.

Designate data tables in Guided Application Creator

Select an existing table or create a custom table in Guided Application Creator to store data for your custom business application.

Before you begin

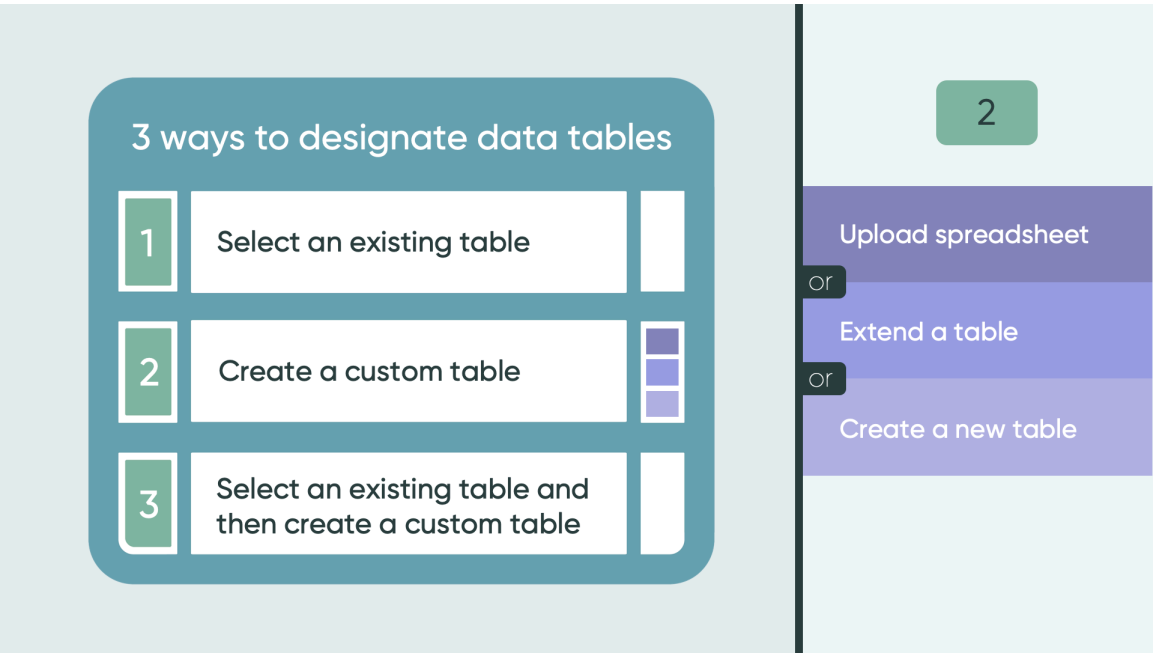
Complete:

1. Create an application record in Guided Application Creator
2. Define roles in Guided Application Creator
3. Select user experiences in Guided Application Creator

If you plan to create a custom data table, review the Data table guidelines for Guided Application Creator to ensure that your system performs as expected after you create a table.

Role required: sn_g_app_creator.app_creator or admin

About this task



Procedure

Designate data tables for your application.
You can select existing tables or create custom tables.

What to do next

Continue building your application by following the steps in Customize user experiences in Guided Application Creator. If you exit Guided Application Creator, the data tables that you designated are not saved to your application.

Upload a spreadsheet in Guided Application Creator

Turn your spreadsheet into a custom table in Guided Application Creator to store data for your custom application.

Before you begin

Complete:

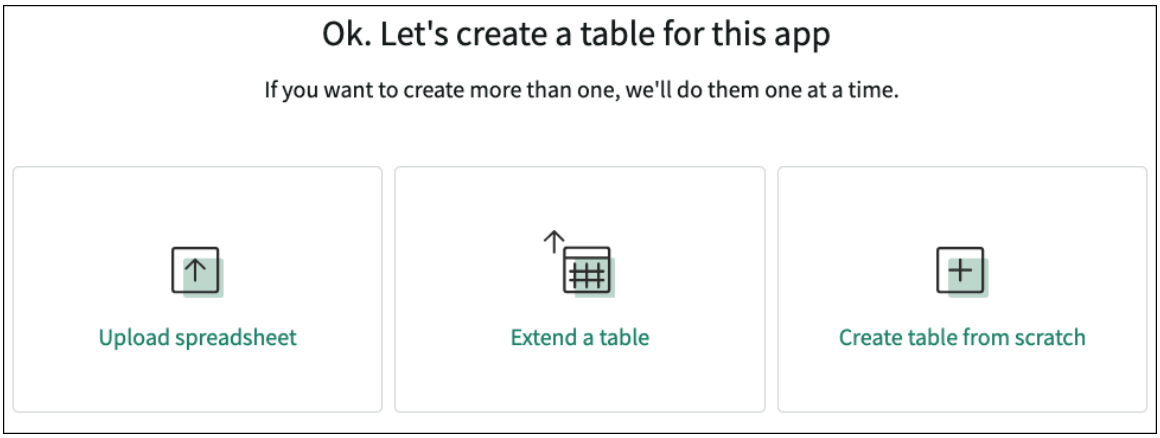
1. Create an application record in Guided Application Creator
2. Define roles in Guided Application Creator
3. Select user experiences in Guided Application Creator
4. Designate data tables in Guided Application Creator

Review the spreadsheet guidelines to ensure that your data uploads as expected.

Role required: sn_g_app_creator.app_creator or admin

Procedure

1. To select a table creation method, on the screen, select **Upload spreadsheet** and then select **Continue**.



2. Upload your spreadsheet.

   You can drag and drop the spreadsheet file or browse your system to select the spreadsheet file.

3. In **Enter a row for the table header**, enter the number of the header row on your spreadsheet.

   For example, if your spreadsheet has two columns in which the headers are in the second row (such as in the following image), you would enter 2.

	A	B
1		
2	**Client name**	**Phone number**
3	Beth Anglin	555-555-5551
4	Fred Luddy	555-555-5552
5	David Loo	555-555-5553
6	John Jason	555-555-5554
7	David Miller	555-555-5555
8		

4. Select the **Import spreadsheet data** option.
5. Select **Continue**.

   Your spreadsheet content is parsed into fields for your custom table.

6. On the form, fill in the fields.

Field	Description
Field Label	Unique label for the field.
Field Name	Name of the field in the database.
Field Type	Type of field. For more information on the different field types, see Field types. By default, there are only 18 field types to choose from. You can add a property to include more field types in Guided Application Creator. For more information, see Add field types in Guided Application Creator.
Display	When used as a reference field, the field is used as the display value for the table. Only one field can be defined as the display value for a table.
Mandatory	Requires that the field must contain a value before the record can be saved.

7. Select **Continue** to define properties and permissions for your custom table.
8. On the form, fill in the fields.

Field	Description
Table label	Unique label for the table (such as **Laptops** or **Thin clients**). The label appears on list and form views for the table. See Field Labels in Data dictionary tables.
Table name	Name of the table in the database. The table name is automatically populated based on the table label and the application scope that you defined earlier.
Make extensible	Option to enable other tables to extend this table. For more information on table extension, see Table extension and classes.
Auto-number	Option to add a number field to the table and automatically increment the ID numbers as they get added.
Manage access	User permissions for your application. For each role that you selected earlier, you can select to give your different levels of access. **Create** Enables users to insert new records (rows) into a table. **Read** Enables users to display records from a table. **Write** Enables users to update records in a table. **Delete** Enables users to remove records from a table or drop a table.

9. Select **Continue**.
10. On the confirmation screen, select **Continue**.
11. To add more tables to your application, follow the steps in Designate data tables in Guided Application Creator.
12. To finish designating tables, select **Done with tables**.

What to do next

Continue building your application by following the steps in Customize user experiences in Guided Application Creator. If you exit Guided Application Creator, the tables that you configured are not saved to the system.

Extend a table in Guided Application Creator

Extend a table in Guided Application Creator to create a custom table that copies an existing table. You can add more fields to your child table.

Before you begin

Complete:

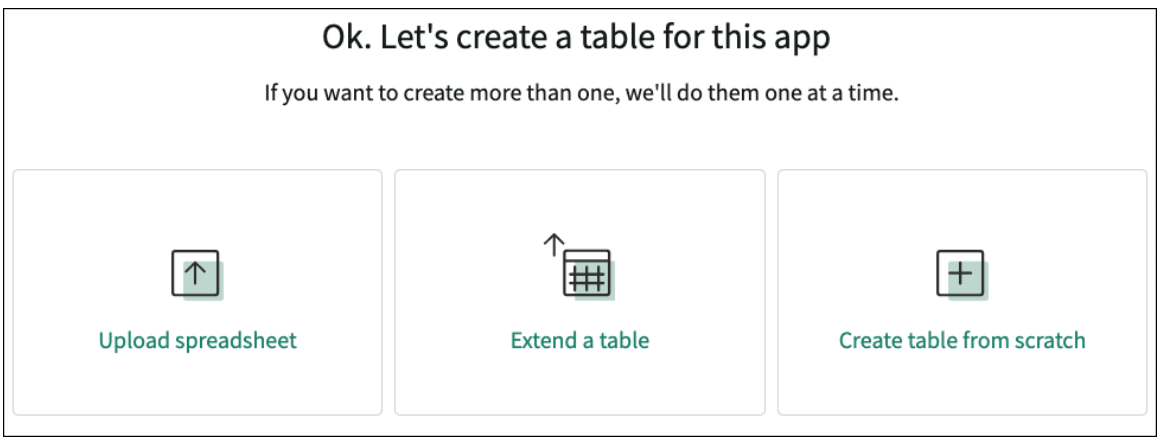
1. Create an application record in Guided Application Creator
2. Define roles in Guided Application Creator
3. Select user experiences in Guided Application Creator
4. Designate data tables in Guided Application Creator

Plan which table to extend. Review the extension model so that you can track which database tables are created after you extend your table.

Role required: sn_g_app_creator.app_creator or admin

Procedure

1. To select a table creation method, on the screen, select **Extend a table**.



2. In the **Table** field, select a table to extend.
3. Select **Continue**.

   The fields from the parent table are copied over to your custom table. You can't change the copied fields.

4. To add a field to your child table, select **+ Add a new field** and then fill in the fields.

Field	Description
Field Label	Unique label for the field.
Field Name	Name of the field in the database.
Field Type	Type of field. For more information on the different field types, see Field types. By default, there are only 18 field types to choose from. You can add a property to include more field types in Guided Application Creator. For more information, see Add field types in Guided Application Creator.
Display	When used as a reference field, the field is used as the display value for the table. Only one field can be defined as the display value for a table. **Note:** Extended tables inherit the display value of the parent table. Setting a separate display value for the extended table overrides the parent table's display value.
Mandatory	Requires that the field must contain a value before the record can be saved.

5. Select **Continue** to define properties and permissions for your custom table.
6. On the form, fill in the fields.

Field	Description
Table label	Unique label for the table (such as **Laptops** or **Thin clients**). The label appears on list and form views for the table. See Field Labels in Data dictionary tables.
Table name	Name of the table in the database. The table name is automatically populated based on the table label and the application scope that you defined earlier.
Make extensible	Option to enable other tables to extend this table. For more information on table extension, see Table extension and classes.
Auto-number	Option to add a number field to the table and automatically increment the ID numbers as they get added.
Manage access	User permissions for your application. For each role that you selected earlier, you can give different levels of access. **Create** Enables users to insert new records (rows) into a table. **Read** Enables users to display records from a table.

Field	Description
	Write Enables users to update records in a table. **Delete** Enables users to remove records from a table or drop a table.

7. Select **Continue**.
8. On the confirmation screen, select **Continue**.
9. To add more tables to your application, follow the steps in Designate data tables in Guided Application Creator.
10. To finish designating tables, select **Done with tables**.

What to do next

Continue building your application by following the steps in Customize user experiences in Guided Application Creator. If you exit Guided Application Creator, the tables that you configured are not saved to the system.

Create a table in Guided Application Creator

Create a table in Guided Application Creator to customize your application to fit your business needs.

Before you begin

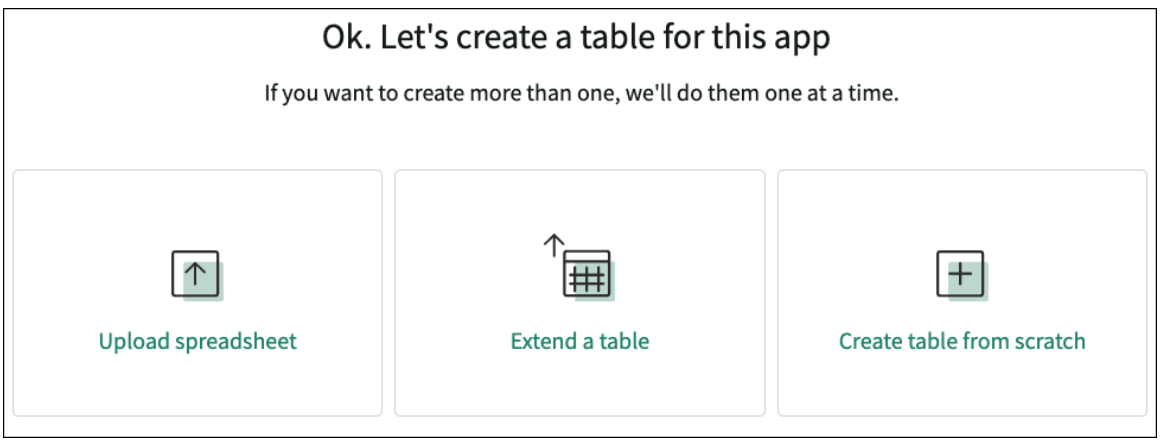
Complete:

1. Create an application record in Guided Application Creator
2. Define roles in Guided Application Creator
3. Select user experiences in Guided Application Creator
4. Designate data tables in Guided Application Creator

Role required: sn_g_app_creator.app_creator or admin

Procedure

1. To select a table creation method, on the screen, select **Create table from scratch** and then select **Continue**.



2. Add fields to your custom table.

a. Select **+ Add a new field**.

b. On the form, fill in the fields.

Field	Description
Field Label	Unique label for the field.
Field Name	Name of the field in the database.
Field Type	Type of field. For more information on the different field types, see Field types. By default, there are only 18 field types to choose from. You can add a property to include more field types in Guided Application Creator. For more information, see Add field types in Guided Application Creator.
Display	When used as a reference field, the field is used as the display value for the table. Only one field can be defined as the display value for a table.
Mandatory	Option to require that the field must contain a value before the record can be saved.

3. Select **Continue** to define properties and permissions for your custom table.

4. On the form, fill in the fields.

Field	Description
Table label	Unique label for the table (such as **Laptops** or **Thin clients**). The label appears on list and form views for the table. See Field Labels in Data dictionary tables.
Table name	Name of the table in the database. The table name is automatically populated based on the table label and the application scope that you defined earlier.
Make extensible	Option to enable other tables to extend this table. For more information on table extension, see Table extension and classes.
Auto-number	Option to add a number field to the table and automatically increment the ID numbers as they get added.
Manage access	User permissions for your application. For each role that you selected earlier, you can give different levels of access. **Create** Enables users to insert new records (rows) into a table. **Read** Enables users to display records from a table.

Field	Description
	Write Enables users to update records in a table. **Delete** Enables users to remove records from a table or drop a table.

5. Select **Continue**.
6. On the confirmation screen, select **Continue**.
7. To add more tables to your application, follow the steps in Designate data tables in Guided Application Creator.
8. To finish designating tables, select **Done with tables**.

What to do next

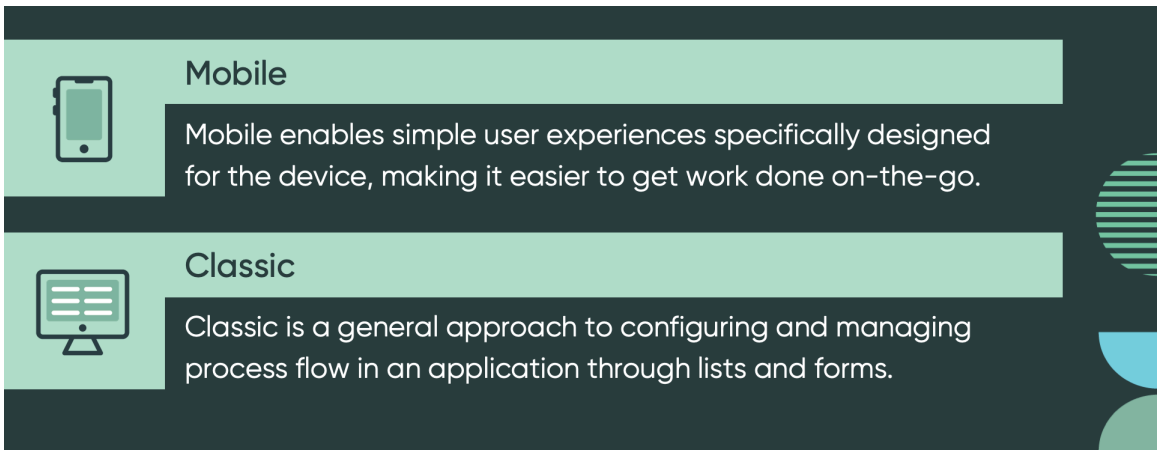
Continue building your application by following the steps in Customize user experiences in Guided Application Creator. If you exit Guided Application Creator, the tables that you configured are not saved to the system.

Customize user experiences in Guided Application Creator

Customize how the application appears in each user experience that you select.

Before you begin:

1. Create an application record in Guided Application Creator
2. Define roles in Guided Application Creator
3. Select user experiences in Guided Application Creator
4. Designate data tables in Guided Application Creator



Customize your application for the Mobile Agent mobile app

If you selected the Mobile Agent mobile app as a user experience for your application, you can customize how the application appears in the mobile app.

Before you begin

Role required: sn_g_app_creator.app_creator or admin

Note:
To edit application properties, a user must have at least one of the following:

- admin role
- sn_g_app_creator.app_creator role (this role is automatically assigned when the sn_app_eng_studio.user role is granted)
- delegated developer with delete permissions
- delegated admin

Procedure

1. Next to **Mobile**, select **Start** to configure application details.
2. On the form, fill in the fields.

Field	Description
App icon	Icon that appears on the application home page in the mobile app.
Available offline	Option to enable users to update application records while in offline mode. This option is available only if the SG Offline Support (com.glide.sg.offline) plugin is activated. For information on offline mode, see Updating records without an internet connection.
Name	Name of your application as it appears on the application home page. By default, this value is the application name that you created earlier.
Description	Description of your application. By default, this value is the application description that you created earlier.
Tables	Tables that appear as applets in the mobile app. Applets are subsections under your application that categorize the information and business processes in your application.
Roles	User roles that can access your application in the mobile app. By default, these values are the roles that you defined earlier.

3. Select **Create**.

What to do next

You've finished customizing your application for the Mobile Agent mobile app. If you selected another user experience, customize that user experience next. If you have no more user experiences left to customize, select **Done with apps**.

Customize your application for the classic ServiceNow AI Platform experience

If you selected the classic ServiceNow AI Platform experience for your application, you can customize how the application appears in the application navigator.

Before you begin

Role required: sn_g_app_creator.app_creator or admin

Note:
To edit application properties, a user must have at least one of the following:

- admin role
- sn_g_app_creator.app_creator role (this role is automatically assigned when the sn_app_eng_studio.user role is granted)
- delegated developer with delete permissions
- delegated admin

Procedure

1. Next to **Classic**, select **Start** to configure application details.
2. On the form, fill in the fields.

Field	Description
Name	Name of your application as it appears on the application navigator. By default, this value is the application name that you created earlier.
Description	Description of your application. By default, this value is the application description that you created earlier.
Tables	Tables in your application. Application modules are created for each table. By default, these values are the data tables that you designated earlier.
Roles	User roles that can access your application. By default, these values are the roles that you defined earlier.

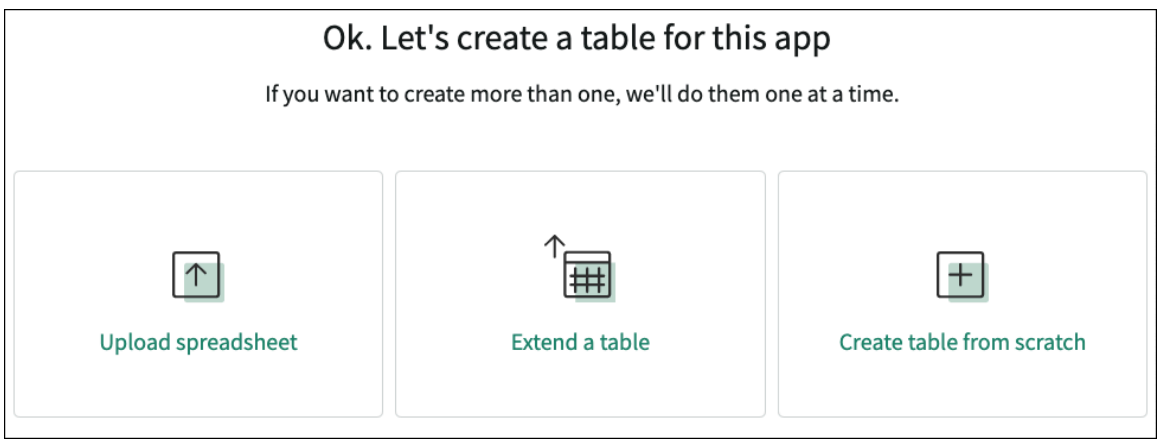
3. Select **Create**.

What to do next

You've finished customizing your application for the classic ServiceNow AI Platform experience. If you selected another user experience, customize that user experience next. If you have no more user experiences left to customize, select **Done with apps**.

Data table guidelines for Guided Application Creator

When you create an application in Guided Application Creator, you can optionally create a data table for your application. To ensure that you are within the limits of your subscription and that your application performs as expected, consult the data table guidelines before you create a data table.



Note:

ServiceNow AI Platform application subscriptions include custom table entitlements. You can create custom tables for any purpose, up to the entitlement limit in the subscription. To learn more about how your usage administrator maps the custom tables that you create to subscriptions, see Map custom tables to a product subscription in Subscription Management.

Spreadsheet guidelines

One way to create a new data table in Guided Application Creator is to upload an external spreadsheet. Prepare your spreadsheet beforehand so that you can transfer your data seamlessly and build custom applications faster.

Ensure your spreadsheet is:

- Formatted with horizontal columns and a header label for each
- Saved as an XLSX file type

For example, format your spreadsheet as follows.

	A	B
1		
2	**Client name**	**Phone number**
3	Beth Anglin	555-555-5551
4	Fred Luddy	555-555-5552
5	David Loo	555-555-5553
6	John Jason	555-555-5554
7	David Miller	555-555-5555
8		

In Guided Application Creator, you would designate the second row as the header row so that **Client name** and **Phone number** are uploaded as table fields.

Table extension guidelines

One option for creating a data table in Guided Application Creator is to copy an existing data table and then add more fields to the newly created table. Before you extend a data table, review the extension model for the table so that you can track which database tables are created after extension.

Extending a base table incorporates all the fields of the original table and creates system fields for the new table. If they are in the same scope or if they can be configured from other scopes, you can extend tables that are marked as extensible.

For more information on extension models, see Table extension and classes.

Table creation guidelines

You can optionally create a new table in Guided Application Creator, but it's possible to create more custom tables than your subscription permits. To ensure that you remain within the limits of your subscription, review the custom table entitlements for your organization.

Allow global application development in Guided Application Creator

By default, developers can't create applications in the global scope in Guided Application Creator. You can limit global application development to certain developers by assigning them an additional role.

Before you begin

Important:
Applications that you create in the global scope bypass scope protections and may cause licensing issues. For information on what to expect when you create an application in the global scope, see Application scope.

Role required: admin

About this task

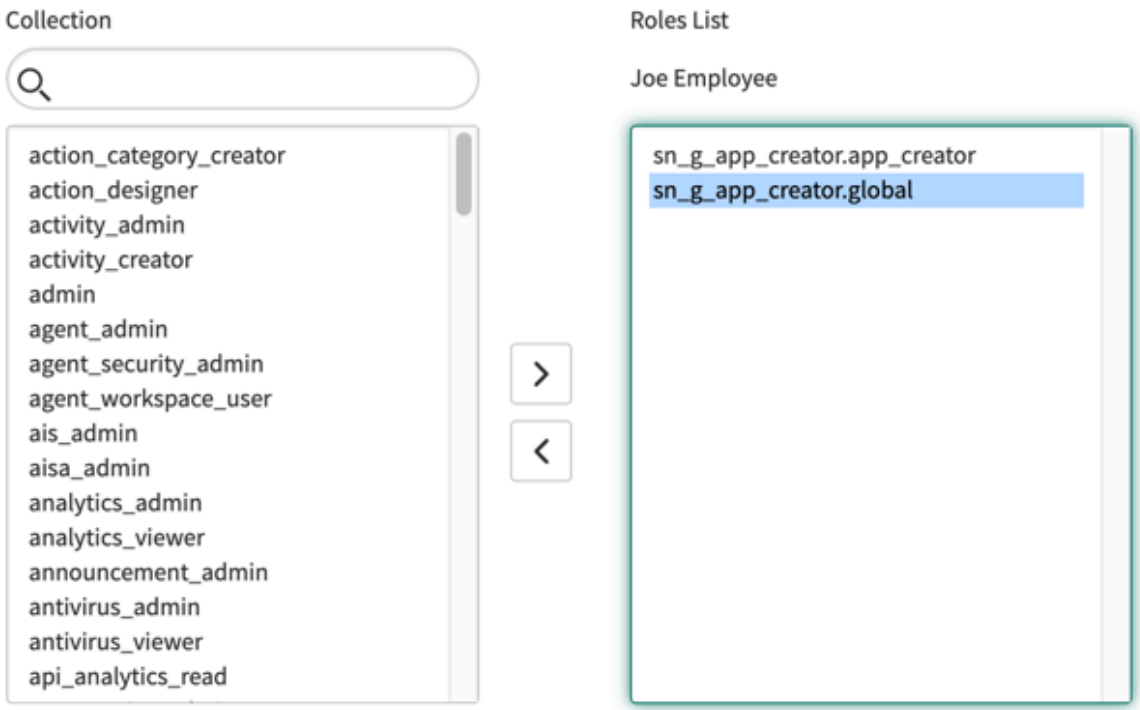
You designate someone to use Guided Application Creator by assigning them the sn_g_app_creator.app_creator role. You can allow this user to create applications in the global scope by also assigning the sn_g_app_creator.global role.

Alternatively, to allow all users with the sn_g_app_creator.app_creator role to create applications in the global scope, add the *sn_g_app_creator.allow_global* system property and set it to **true**. For more information on adding a property, see Add a system property.

Note:
Users with the admin role can create applications in the global scope, regardless of whether or not they have the sn_g_app_creator.global role.

Procedure

1. Navigate to **All** > **User Administration** > **Users** and open the user record of the developer.
2. On the Roles related list, select **Edit...**.
3. On the Edit Members page, move the sn_g_app_creator.global role from **Collection** to **Roles List**.



Ensure that the sn_g_app_creator.app_creator role is also selected. The developer needs both the sn_g_app_creator.app_creator and sn_g_app_creator.global roles to create applications in the global scope in Guided Application Creator.

4. Select **Save**.
5. On the user record, select **Update**.

Result

The option to create an application in the global scope is available to developers with the sn_g_app_creator.global role. Developers can use, for example, ServiceNow Studio, to create an app.

Add field types in Guided Application Creator

When you add fields to a custom table in Guided Application Creator, there are only 18 field types available by default. You can add a property to include more field types.

Before you begin

Role required: admin

About this task

For a list of field types that are available in the ServiceNow AI Platform, see Field types.

Procedure

1. Add a system property with the following settings:
   - Name: *sn_g_app_creator.field_types*
   - Type: string
2. In the **Value** field, enter a comma-separated list of field type names.
   For example, to add the Image and Document ID field types, enter `user_image,document_id`.
3. Select **Submit**.

Scripting

Use scripts to extend your instance beyond standard configurations. With scripts, you may automate processes, add functionality, integrate your instance with an outside application and more.

APIs (Application Programming Interfaces) provide classes and methods that you can use in scripts to define functionality. ServiceNow provides APIs as JavaScript classes, web services, and other points of connection for integrations. Note that you cannot access commonly used JavaScript objects (such as DOM or Window). Jelly scripts are also used in some modules. Jelly is used to turn XML into HTML and may include both client-side and server-side scripts.

Scripts may be server-side (run on the server or database), client-side (run in the user's browser), or run on the MID server.

Note:
When you are writing scripts, you cannot use reserved words.

Understand JavaScript before you begin customizing your instance, and with Jelly if you intend to deploy Jelly scripts.

Server-side scripts

Perform database operations. For example, use a server-side script to update a record. Create a script in a scoped application or in the global scope. Each execution context includes a set of available APIs.

Scoped environment

Use scoped APIs when scripting in a scoped application. Scoped `Glide` APIs do not include all the methods included in the global `Glide` APIs, and you cannot call a global `Glide` API in a scoped application.

Global environment

The global scope is a special application scope that identifies applications developed prior to application scoping, or applications intended to be accessible to all other global applications. Use global APIs when scripting in the global scope.

To learn more about server-side scripting, see Server-side scripting. To learn more about application scope, see Application scope.

Client-side scripts

Make changes to the appearance of forms, display different fields based on values that are entered, or change other custom display options.

- onLoad client scripts run when the form or page is loaded
- onChange client scripts run when something specific gets changed AND also when the form or page loads
- onSubmit client scripts run when the form is submitted

Client Scripts can also be called by other scripts or modules, including UI policies. To learn more about client-side scripting, see Client-side scripting.

Available script types

Scripts can be used in many places. The most important detail is whether the script runs on the client or the server.

Script types and where they run

Script	Description	Runs on
Access Control	Determines whether access will be granted for a specified operation to a specific entity. • type of entity being secured • operation being secured • unique identifier describing the object Can be defined by roles, conditional expressions or scripts.	server - script and any condition run on the server

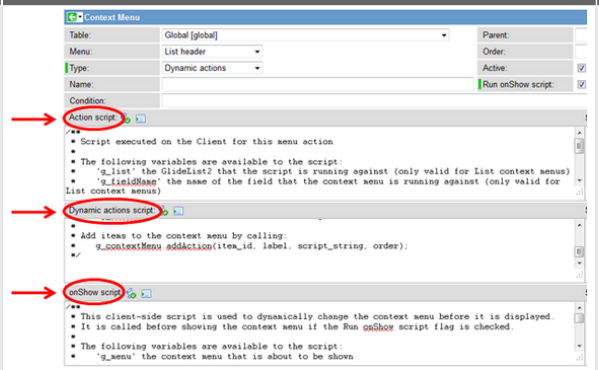
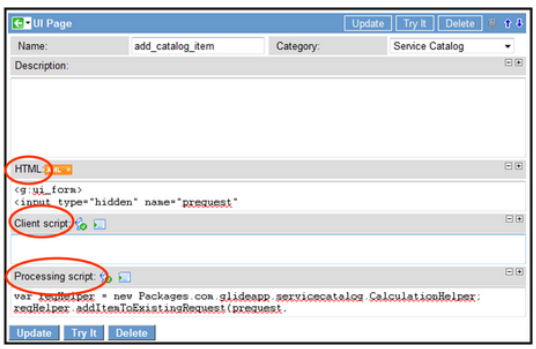
Script types and where they run (continued)

Script	Description	Runs on
Ajax Scripts	Enables the client to get data from the server to dynamically incorporate into a page without reloading the whole page. • Ajax Client Scripts request that information be returned, or that action be taken, or sometimes both • Ajax Server Scripts fulfill Ajax Client Script requests	• client - Ajax Client Scripts run on the client • server - Ajax Server Scripts run on the server
Business Rules	Customizes system behavior • runs when a database action occurs (query, insert, update or delete) • the script may run ○ before or after the database action is performed (runs as part of the database operation) ○ asynchronously (at some point after the database operation) ○ on display (when displaying the data in a form)	server - script and any condition run on the server
Service Catalog UI policies	Defines the display of a variable set or a catalog item (from the service catalog).	• client - scripts in the "execute if true" field or "execute if false" field run on the client • server - all conditions run on the server
Client Scripts	Used for making changes to the appearance of forms, displaying different fields based on values that are entered or other custom display options. • onLoad means the Client Script runs when the form or page is loaded • onChange means the Client Script runs when something specific gets changed AND also when the form or page loads • onSubmit means the Client Script runs when the form is submitted Client Scripts can also be called by other scripts or modules, including UI policies.	client
Script actions	Contains scripts which run when an event occurs, for example	server - script and any condition run on the server

Script types and where they run (continued)

Script	Description	Runs on
	• approval is cancelled • change is approved • problem is assigned Can have a condition which must be true for the script to run. Commonly used to call a Script Include.	
Script Includes	Contains scripts which can be functions or classes. These scripts run only when called by other scripts (often Business Rules). Any server script which is complicated or reusable should be a Script Include (especially complicated Business Rules).	server
Transform maps	Used for importing data. • defines mapping relationships between tables • can use Business Rules, other scripts and/or other options to import that data Do not always include scripts.	server
UI actions	Creates the ability to choose a specific action such as clicking a button or a link. UI Actions put these items on forms and lists: • buttons • links • context menu items • list choices	• client - when the "Client" box is checked, the script in the script field runs on the client • server - when the "Client" box is unchecked, the script in the script field runs on the server • client - when the "Client" box is checked, the onClick script is available, which can contain any JavaScript but normally calls a function which is specified in the script field • server - all conditions run on the server
UI Context Menus	Defines which "right-click menu" will pop-up in which area, and the menu choices that will be available	• client - onShow scripts run on the client • client - action scripts run on the client

Script types and where they run (continued)

Script	Description	Runs on
	 **Note:** If you use a left-handed mouse configuration, right-click means "click the other button."	• server - dynamic action scripts run on the server • server - all conditions run on the server
UI macros	Contains modular, reusable components that can contain Jelly and are called by UI pages. They also contain different types of scripts and may be called multiple times on the same page. **Note:** Jelly turns XML into HTML.	• server - the UI Macro itself executes on the Server • server - may contain content that runs on the server (Jelly expressions or JavaScript inside Jelly constructs) • client - may generate output that runs on the client (embedded JavaScript within <script> tags)
UI Pages	Used to create and display pages, forms, dialogs, lists and other UI components. Can be displayed on a standalone basis, or called as a usable component, as part of a larger page.  Can contain • Client Scripts, • processing scripts (which are server scripts), • HTML,	• server - Jelly XML runs on the server to produce HTML • client - HTML may contain embedded JavaScript that runs on the client • client - client scripts run on the client • server - processing scripts run on the server

Script types and where they run (continued)

Script	Description	Runs on
	• Jelly, • UI Macros, • and also can call other scripts. **Note:** Jelly turns XML into HTML.	
UI Policies	Defines the behavior and visibility of fields on a form. • mandatory • visible • read only Use UI Policies rather than client scripts whenever possible. • UI Policies are always attached to one table • UI Policies often have a condition which must be true in order for them to run	• client - scripts in the "execute if true" field or "execute if false" field run on the client • server - all conditions run on the server
UI Properties	Designates what the instance will look like.	• server - properties set on the server • client - the results get rendered on the client no scripts
UI Scripts	Contains client scripts stored for re-use. Only used when called from other scripts. Not recommended for use.	client
Validation Scripts	Validates that values are in a specified format. For example, a validation script can verify that the only value allowed in a specific field is an integer.	client
Workflow editor	Used to create or change a workflow. Scripts can be run at any point in a workflow, or different scripts can be run at different points. Scripts also can be found inside every workflow activity and can be modified (although do so with extreme caution).	server - script and any conditions run on the server

Glide class overview

The ServiceNow Glide classes expose JavaScript APIs that enable you to conveniently work with tables using scripts.

Using the Glide APIs, you can perform database operations without writing SQL queries, display UI pages, and define UI actions. The following tables provide brief descriptions of the Glide classes and links to detailed information.

Server-side Glide classes

Class	Description
GlideRecord	Use this class for database operations instead of writing SQL queries. GlideRecord is a special Java class that can be used in JavaScript exactly as if it were a native JavaScript class. A GlideRecord is an object that contains records from a single table. See GlideRecord.
GlideElement	Use this class to operate on the fields of the current GlideRecord. See GlideElement.
GlideSystem	Use this class to get information about the system. See GlideSystem.
GlideAggregate	Use this class to perform database aggregation queries, such COUNT, SUM, MIN, MAX, and AVG, for creating customized reports or calculations in calculated fields. See GlideAggregate.
GlideDateTime	Use this class to perform date-time operations, such as date-time calculations, formatting a date-time, or converting between date-time formats. See GlideDateTime.

Client-side Glide Classes

Class	Description
GlideAjax	Use this class to execute server-side code from the client. See GlideAjax.
GlideDialogWindow	Use this class to display a dialog window. See GlideDialogWindow.
GlideForm	Use this class to customize forms. See GlideForm.
GlideList2	Use this class to customize (v2) lists, including normal lists and related lists. See GlideList2.
GlideMenu	Use this class to customize UI Context Menu items. See GlideMenu.
GlideUser	Use this class to get session information about the current user and current user roles. See GlideUser.

Glide stack

Glide is an extensible Web 2.0 development platform written in Java that facilitates rapid development of forms-based workflow applications (work orders, trouble ticketing, and project management, for example).

User interface stack technology map

Java packages		Technologies used
	User Interface (Browser)	• AngularJS • HTML • CSS • JavaScript
com.glide.ui com.glide.jelly	GlideServlet	Apache Jelly
com.glide.script	Business Rules	Mozilla Rhino
com.glide.db	Persistence	JDBC

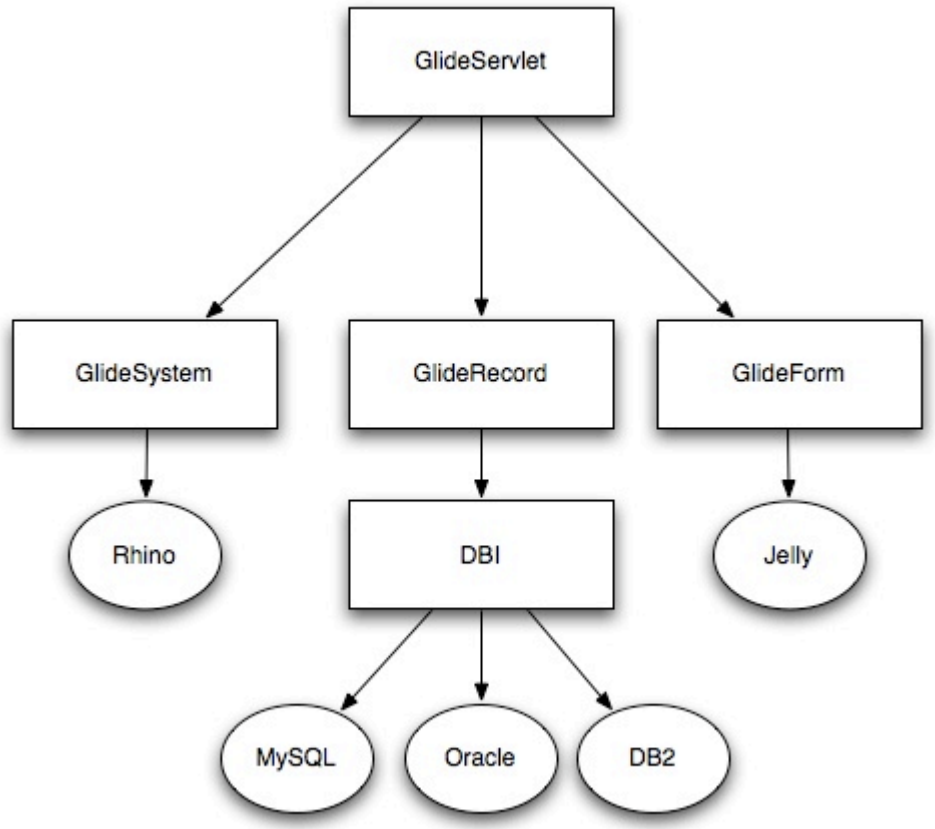
User interface stack technology map descriptions

Name	Description	Attributes
GlideServlet	The primary driver of Glide, and the only servlet in the system, is found in GlideServlet.java. The GlideServlet:	• Handles inbound action requests • Renders pages • Merges data with forms • Presents to user • Interfaces with script layer
Business Rules		• ECMA / JavaScript implementation based on Mozilla Rhino • Interfaces with persistence layer using JDBC recordset interface • Merges persistence layer meta-data with data for easy correlation
Persistence		• Persistence means any store ○ RDBMS ○ LDAP ○ File system • Uniform access regardless of store type

User interface stack technology map descriptions (continued)

Name	Description	Attributes
		• Provides QUID and meta-data capabilities • Interfaces presented to callers ○ RecordSet ○ TableDescriptor ○ ElementDescriptor

Glide servlet



Execution order of scripts and engines

Scripts, assignment rules, business rules, workflows, escalations, and engines all take effect in relation to a database operation, such as insert or update. In many cases, the order of these events is important.

Note:
Client-based code that executes in the browser, using Ajax or running as JavaScript, will always execute before the form submission to the server.

The order of execution is as follows:

1. Before business rules: Scripts configured to execute before the database operation with an order less than 1000.
2. Before engines. The following are not executed in any specific order:

- Approval engine (for task and sys_approval_approver tables)
- Assignment rules engine (for task tables)
- Escalation engine
- Data policy engine
- Field normalization engine
- Role engine - keeps role changes in sync with sys_user_has_role table (for sys_user, sys_user_group, sys_user_grmember, and sys_user_role tables)
- Execution plan engine (for task tables)
- Update version engine - creates version entry when sys_update_xml entry is written (for sys_update_xml table)
- Data lookup engine inserts or updates
- Workflow engine (for default workflows)

3. Before business rules: Scripts configured to execute before the database operation with an order greater than or equal to 1000.
4. The data base operation (insert, update, delete).
5. After business rules: Scripts configured to execute after the database operation with an order less than 1000.
6. After engines. The following are not executed in any specific order:
   - Label engine
   - Listener engine
   - Table notifications engine
   - Role engine - keeps role changes in sync with sys_user_has_role table (for sys_user, sys_user_group, sys_user_grmember and sys_user_role tables)
   - Text indexing engine
   - Update sync engine
   - Workflow engine (for deferred workflows)
   - Trigger engine (for all Workflow Studio flows)
7. Email notifications. The following are executed based on the weight of the notification record:
   - Notifications sent on an insert, update, or delete
   - Event-based notifications
8. After business rules (Only active records). Scripts configured to execute after the database operation with an order greater than or equal to 1000.

Note:
Like After business rules, Async business rules execute their logic after a database operation occurs. Unlike After business rules, Async business rules execute asynchronously, running in the background simultaneously with other processes. Async business rules run after the user submits the form and after the scheduler runs the scheduled job created from the business rule. The system creates a scheduled job from the business rule after the user submits the form but before any action is taken on the record in the database.

Script evaluation of fields by data type

Script fields evaluate data based on the field type of the input.

Evaluation of fields by data types

Type	Evaluates to in script	Example
String	The string	"dog" > "dog"
Decimal	A number with up to two decimal points	12.34 > 12.34
Integer	A number with zero decimal points	12 > 12
True / False	true or false	 > true  > false
Date	A date formatted as yyyy-mm-dd	2008-11-04
Date-time	A day and time formatted as yyyy-mm-dd hh:mm:ss	2008-11-04 06:46:20
Duration	A date that is equal to January 1st 1970 00:00:00 + the amount of time of the duration being stored **Note:** This date corresponds to the system time zone. If a different user time zone has been specified, the date and time value may appear different for that user.	> "1970-01-01 00:00:00" > "1970-01-02 02:03:04"
Choice	Returns the contents of the value field for the `sys_choice` record associated	> "2" (Note that this value is is a string)

Evaluation of fields by data types (continued)

Type	Evaluates to in script	Example
	with that choice. See: Choice List for more information on returning the value associated with a particular item in a choice list.	
Journal	Returns a string of all entries made to that journal field. See Journal Fields for scripting of journal type fields	The web server is down > The web server is down
Reference	Returns the `sys_id` of the record that is referenced	> "287ee6fea9fe198100ada7950d0b1b73"
Image	Returns the path to the image	> images/icons/image_name.gif
URL	Returns a string	> "http://www.service-now.com"
Glide Lists	Returns a string of comma-separated Sys IDs	> 5137153cc611227c000bbd1bd8cd2007,46d14f04a9fe19810142e40c6b071512

Scripting alert, info, and error messages

You can send messages to customers as alerts, informational messages, or error messages.

Business rule and other general use scripts

Script	Result
`current.field_name.setError("Hello World");`	Adds `Hello World` below the specified field in a red error message.
`gs.addInfoMessage("Hello World");`	Adds `Hello World` at the top of the browser window in a blue info message.
`gs.addErrorMessage("Hello World");`	Adds `Hello World` at the top of the browser window in a red error message.
`gs.print("Hello World");`	Writes `Hello World` to the text log on the file system but not to the sys_log table in the database.
`gs.log("Hello World");`	Writes `Hello World` to the database and the log file.

Business rule and other general use scripts (continued)

Script	Result
	Note: gs.log can adversely affect performance if used too frequently.

Client side scripts

Script	Result
`alert("Hello World");`	Displays a window with `Hello World` and an **OK** button. **Note:** Rather than use JavaScript *`alert()`*, for a cleaner look, you can display an error on the form itself with thee *`showFieldMsg()`* and *`hideFieldMsg()`* methods. For more information, see Display field messages.
`confirm("Hello World");`	Displays a window with `Hello World?` and **OK** and **Cancel** buttons.
`g_form.showFieldMsg("field_name", "Hello World", "error");`	Adds `Hello World` below the specified field in a red error message.
`g_form.hideFieldMsg("field_name");`	Hides the most recent message for the specified field.

Important: The methods in this table are only for use in client scripts.

It is also possible to add other custom messages to your forms if necessary using client scripting.

The text size of info and error messages at the top of the screen is customizable. Two properties control this. If you configured your forms, you may need to add these properties.

Note: The client script alert option is not available for use with Automated Test Framework (ATF).

Error and alert text size properties

Property	Description
`css.outputmsg.info.text.font-size`	Sets the size for info messages. Default is 11pt.
`css.outputmsg.error.text.font-size`	Sets the size for error messages. Default is 11pt.

Now Assist for code generation

Use ServiceNow® Now Assist for code generation for help writing scripts quickly with AI-generated code based on text or code prompts.

https://player.vimeo.com/video/1022256119?h=444ebb1578&badge=0&autopause=0&player_id=0&app_id=58479

Get started

Explore  Learn about how developers script with AI-generated code.	Configure  Configure code generation functionality for an instance.	Generate scripts  Write scripts quickly with AI-generated code.
Edit scripts  Edit scripts quickly with AI-generated code.	Reference  Get details about properties, roles, and more.	

Troubleshoot and get help

- ServiceNow Community
- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

AI limitations

This application uses artificial intelligence (AI) and machine learning, which are rapidly evolving fields of study that generate predictions based on patterns in data. As a result, this application may not always produce accurate, complete, or appropriate information. Further, there is no guarantee that this application has been fully trained or tested for your use case. To mitigate these issues, it is your responsibility to test and evaluate your use of this application for accuracy, harm, and appropriateness for your use case, employ human oversight of output, and refrain from relying solely on AI-generated outputs for decision-making purposes. This is especially important if you choose to deploy this application in areas with consequential impacts such as healthcare, finance, legal, employment, security, or infrastructure. You agree to abide by ServiceNow's AI Acceptable Use Policy, which may be updated by ServiceNow.

Data processing

This application requires data to be transferred from ServiceNow customers' individual instances to a centralized ServiceNow environment, which may be located in a different data center region from the one where your instance is, and potentially to a third-party cloud provider, such as Microsoft Azure. This data is handled per ServiceNow's internal policies and procedures, including our policies available through our CORE Compliance Portal.

Data collection

ServiceNow collects and uses the inputs, outputs, and edits to outputs of this application to develop and improve ServiceNow technologies including ServiceNow models and AI products. In addition, this application will collect information about scripts (and associated script records) in which Now Assist for code generation is called. Customers can opt out of future data collection at any time, as described in the Now Assist Opt-Out page.

For more information, see the Now Assist documentation.

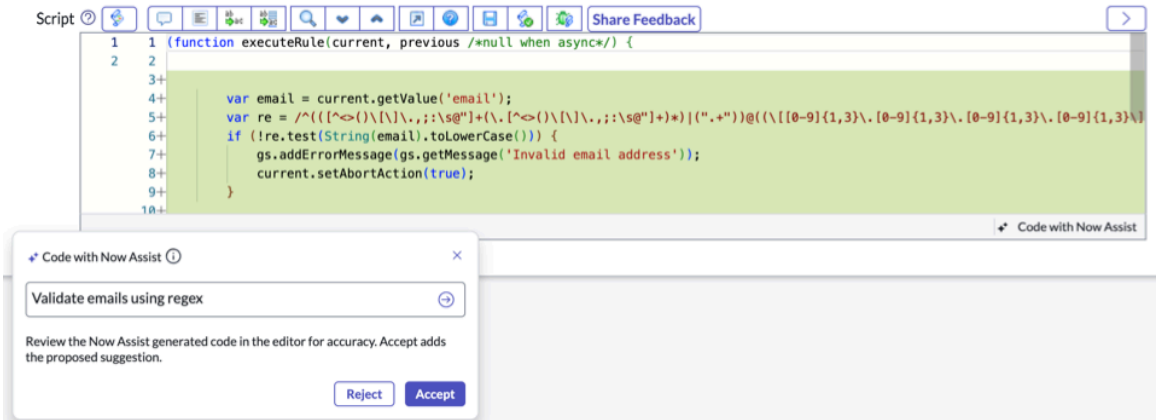
Exploring code generation

Learn about how AI-generated code can empower developers scripting on the ServiceNow AI Platform.

Code generation overview

Now Assist for Creator activates the code generation skill. With code generation, you provide text describing the code to generate and get code suggestions in the JavaScript editor on forms in the ServiceNow AI Platform and in Script steps in Workflow Studio. Developers with varying levels of experience in scripting on the ServiceNow AI Platform can benefit from using code generation to get started writing custom scripts or iterate on scripts more efficiently.

To generate code suggestions, you describe the goal of the code to generate in the Code with Now Assist dialog box. The code suggestion appears in the lines following your prompt but isn't added to your script until it's accepted.

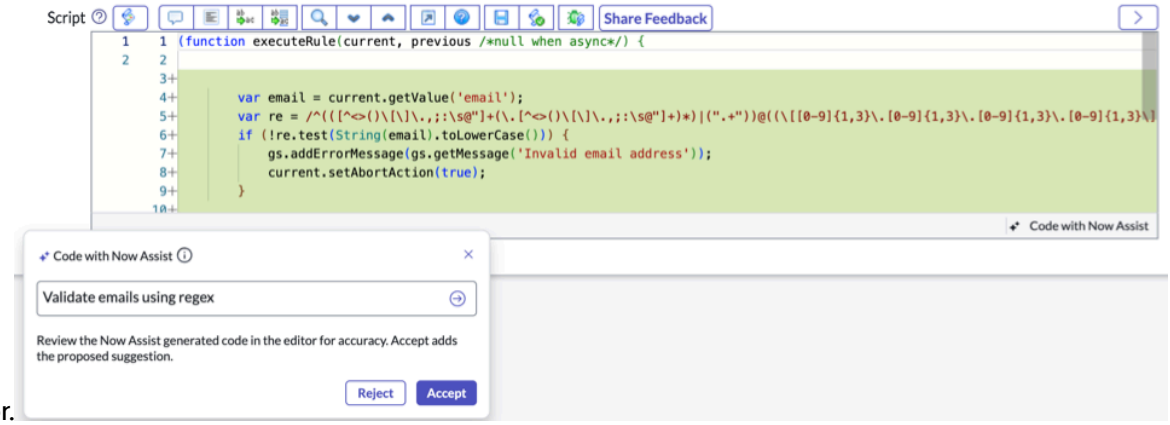


Note:
Developers must be assigned the now.assist.creator role to use code generation. For more information about using code generation, see Generate scripts with AI-powered code generation.

Code generation workflow

1. From the script editor, a developer opens the Code with Now Assist dialog box and describes the code they want to generate.
2. The developer triggers generating a code suggestion.

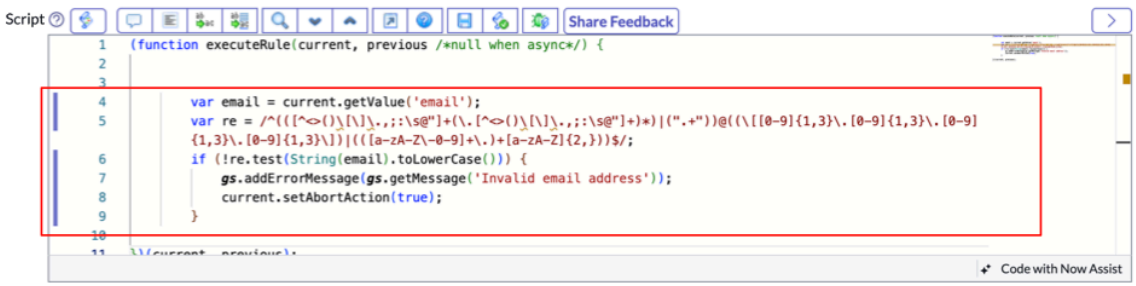
In the following example, a developer describes what they want the script to do in the Code with Now Assist dialog box. The code suggestion appears highlighted in the script editor.



3. The developer reviews the AI-generated code suggestion and either accepts or rejects it.
   - If they accept it, the code is added to the script, and they can make any necessary edits based on their review.
   - If they reject it, the code isn't added to the script, and they can rephrase their prompt to generate a new code suggestion.

   In the following example, a line next to the line numbers indicates which code was created by AI and hasn't been edited. If you edit AI-generated code, the line indicator doesn't appear for those lines of code.

   AI-generated code lines



Optionally, you can turn on code completion functionality to use code or single-shot prompts in script editors with Now Assist for code generation.

Code generation benefits

Benefit	Feature	Users
Improve quality of scripts, automate repetitive coding tasks, and reduce time spent searching for or recalling code	Text-to-code and code complete	Developers
Iterate on scripts efficiently	Code editing	Developers
Identify code generated by AI	Tracking AI-generated code	Developers, administrators

General guidelines for code generation

Use these general guidelines for code generation to get better code suggestions and create useful and accurate scripts.

Writing prompts

Write clear and specific but concise prompts

Specify the expected outcome and context, including necessary details like task requirements, specific APIs if you know them, and any constraints.

Experiment with different prompts

As you refine and experiment, the Now LLM Service uses this feedback to learn and improve.

- Try adjusting task instructions and incorporating examples, and then observe how code suggestions differ with different prompt styles and levels of detail.
- Try including a short code snippet as an example of how to start the script with a single-shot prompt.
- Keep track of your prompts, including any modifications, and instructions for generating prompts to meet your specifications. This tracking enables easy regeneration of past results for comparative analysis.

Example prompts for code generation

Strong prompt	Weak prompt	Notes
Get incidents with related tasks	Get incidents with tasks	Includes sufficient detail.
Use glide aggregate to count number of P1 incidents closed between 3rd March to 13 April assigned to admin	Count P1 incidents between 3-3 and 4-13	Includes the API name and more specific language.
If open change request is P1, do not allow reducing the severity unless it's the creator	Don’t allow changing P1 change requests	Includes more specific instructions on what shouldn't change.
Gliderecord of the most recent change	Latest change	Includes the API name and more specific language.

Reviewing code

Review code

Implement strict and detailed reviews of the AI-generated code to determine its accuracy, efficiency, and how well it adheres to your coding standards.

Test code

Validate the code by running it against test cases in controlled environments to ensure that it functions according to your requirements.

Configuring code generation

Configure how developers can use code generation on an instance.

Install Now Assist for code generation

Install the ServiceNow® Now Assist for Creator application from the ServiceNow® Store to get Now Assist for code generation.

Before you begin

- Review the Now Assist for Creator application listing in the ServiceNow Store for information on dependencies, licensing or subscription requirements, and release compatibility. Now Assist for Creator installs the Now Assist for code generation application (sn_now_assist_code).

Role required: admin

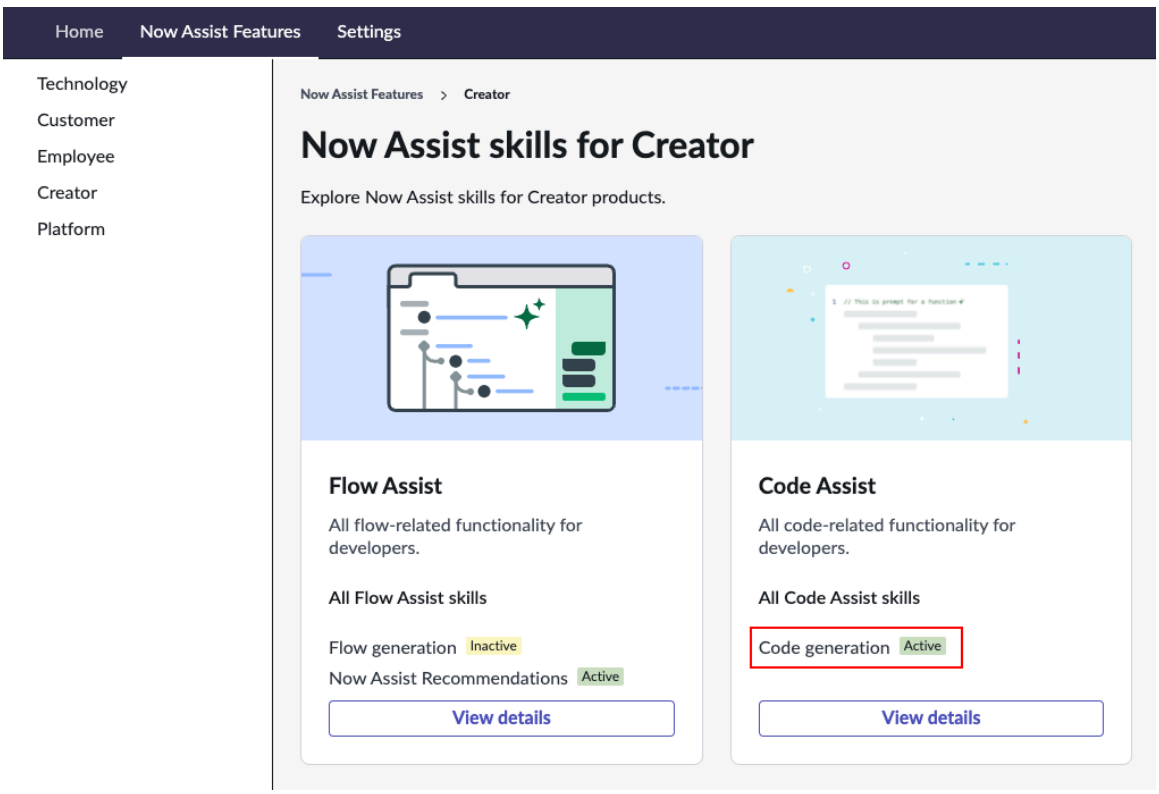
Procedure

1. From the Now Assist for Creator application page on the ServiceNow Store, select **Request App**.
2. After approval has been granted, on your instance, navigate to **All** > **System Applications** > **All Available Applications** > **All**.
3. Using the search bar, search for the Now Assist for Creator application (sn_now_creator).
4. Select **Install**.
5. Verify that Now Assist for Creator is installed:

   a. Navigate to **All** > *Now Assist Admin* > **Features**.

   b. In the workflow list, select **Creator**.

   c. On the Code Assist card, verify that the code generation skill is active.



For more information about Now Assist Admin, see Now Assist.

What to do next
Grant the now.assist.creator role to each user you want to use code generation.

Related topics
Install Now Assist for Creator

Tracking AI-generated code
Control when to track and indicate that code is AI-generated.

When you accept a code suggestion, the lines of code that are AI-generated are tracked and indicated by a line next to the line numbers in script editors. If you edit AI-generated code, the line indicator doesn't appear for those lines of code.

You can turn on and off tracking of AI-generated code and configure how long to record and indicate AI-generated code in script editors. By default, AI-generated code is recorded and indicated for six months.

Configure how long to indicate AI-generated code
Show that code was generated by AI in script editors for a specified duration after accepting code suggestions.

Before you begin
Role required: admin

Procedure
1. In the navigation filter, enter `sys_auto_flush.list`.
2. Filter the Auto Flushes [sys_auto_flush] list by entering *`sn_now_assist_code_ai_generated_lines`* in the **Tablename** column filter.
3. Select *`sn_now_assist_code_ai_generated_lines`* to open the record.
4. In the **Age in seconds** field, enter the duration to record and indicate AI-generated code after its creation in seconds.
5. Select **Update**.

Result
AI-generated code created before the configured duration isn't recorded or indicated in script editors.

Turn off tracking AI-generated code
Turn off tracking and indicating which code is AI-generated.

Before you begin
Role required: admin

About this task
Tracking and indicating which code is AI-generated involves changes that are included in update sets. If you don't want to include changes related to this in update sets, you can turn off tracking and indicating AI-generated code.

Procedure
1. In the navigation filter, enter `sys_properties.list`.
2. Filter the System Properties [sys_properties] list by entering *`sn_now_assist_code.show_ai_code_line_marker`* in the **Name** column filter.

3. Select *sn_now_assist_code.show_ai_code_line_marker* to open the property record.
4. In the **Value** field, enter `false`.
5. Select **Update**.

Result
AI-generated code isn't recorded or indicated in script editors.

Generate scripts with AI-powered code generation

Generate scripts from text, code, or a combination of both with AI-powered code generation.

When code generation is enabled on an instance, a Now Assist icon (✨) appears in the script editor.

Developers must be assigned the now.assist.creator role to use code generation.

Generate scripts from text

Write scripts quickly with AI-generated code by telling Now Assist what you want the script to do.

Before you begin
Learn how to write prompts to generate better code suggestions. For more information, see General guidelines for code generation.

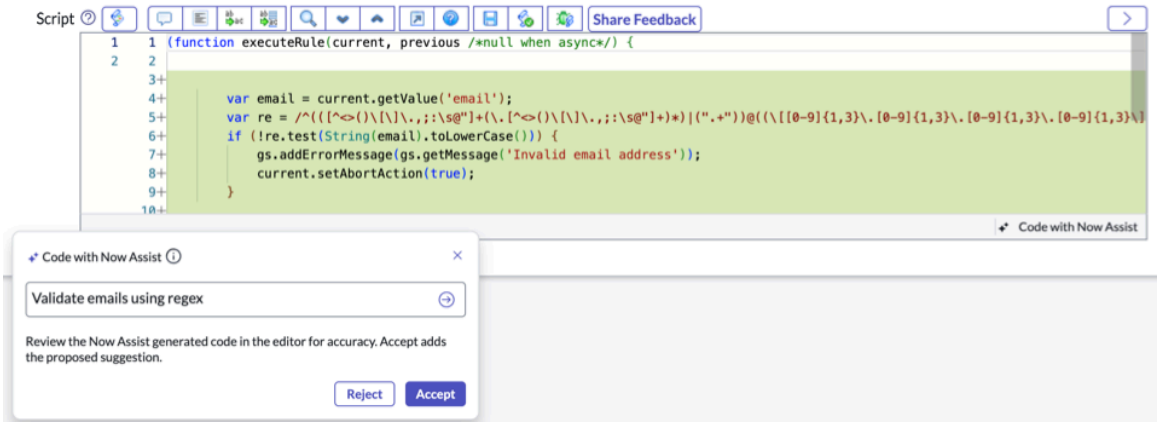
Role required: now.assist.creator

Procedure
1. Navigate to a form with a script field.

   Example
   For example, to open a script include form, navigate to **All** > **System Definition** > **Script Includes** and select **New** or enter `sys_script_include.do` in the navigation filter.
2. In the script editor, place your cursor where you want to add code.
3. Right-click and select **Open Code with Now Assist** or use one of the following keyboard shortcuts:
   - Windows: Ctrl-Enter
   - Mac: Cmd-Enter

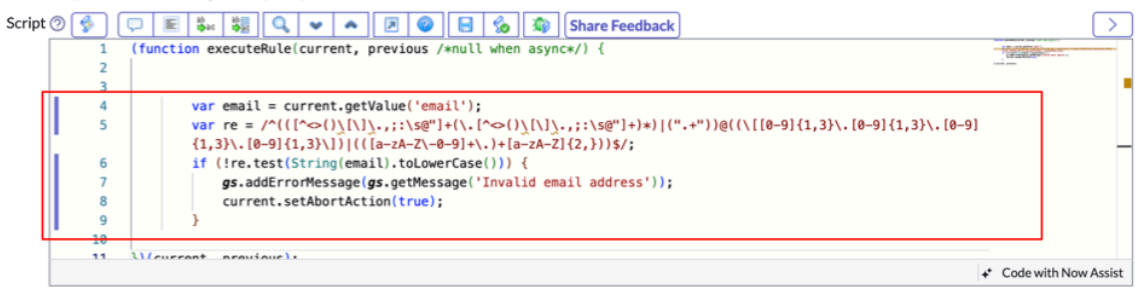
   Tip:
   Select the Help icon (🔵) to access the list of relevant keyboard shortcuts.
4. In the **Code with Now Assist** dialog box, enter text that describes the desired goal of the code to generate.

   The text you enter must be fewer than 1,000 characters.
5. Press Enter to generate a code suggestion.
   The code suggestion appears highlighted in the script editor.



6. Review the code suggestion and complete one of the following steps:
   - To include it in your script and make any edits, select **Accept**.
   - To regenerate a suggestion, revise the text in the dialog box and select the arrow icon ( ).
   - To remove it from the script, select **Reject**.

   When you accept a code suggestion, a line next to the line numbers indicates which code was created by AI and hasn't been edited. If you edit AI-generated code, the line indicator doesn't appear for those lines of code.



Trouble?
If the code suggestion doesn't meet your requirements, try rephrasing your prompt according to the prompt guidance and generating another code suggestion.

7. Select **Submit** or **Update** to save your changes.

Generate scripts from code

Write scripts quickly with AI-generated code completion.

Before you begin

Code completion must be turned on for the instance.

Role required: now.assist.creator

Procedure

1. Navigate to a form with a script field.

   Example
   For example, to open a script include form, navigate to **All** > **System Definition** > **Script Includes** and select **New** or enter `sys_script_include.do` in the navigation filter.

2. In a script field, enter code or a combination of text and code:

Example

- Enter the beginning of a function or other code to be automatically completed. For example:

```
var email = current.getValue('email');
var regex =
```

- Enter a combination of text in a code comment that describes the desired goal of the code to generate followed by an example of how you want the code to start. For example:

```
// Validate emails using regex
var email = current.getValue('email');
var regex =
```

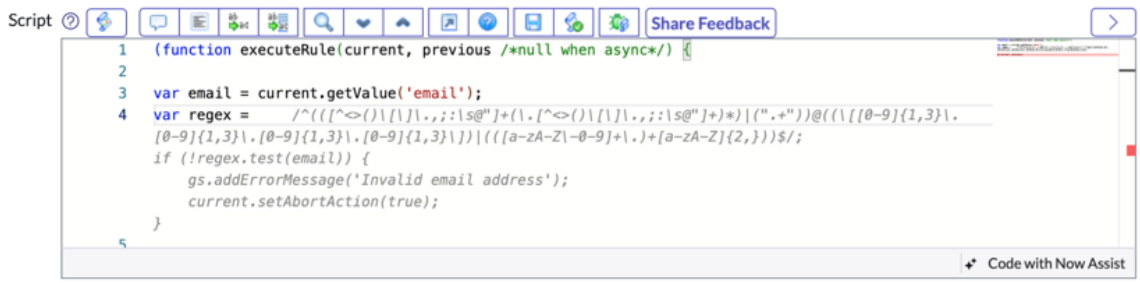
3. Right-click and select **Auto-generate code completion** or use one of the following keyboard shortcuts to generate a code suggestion:
   - Windows: Ctrl-Windows logo key-Enter
   - Mac: Ctrl-Cmd-Enter

   Tip:
   Select the Help icon ( ) to access the list of relevant keyboard shortcuts.

   The code preceding your cursor must be fewer than 1,000 characters when triggering code generation.

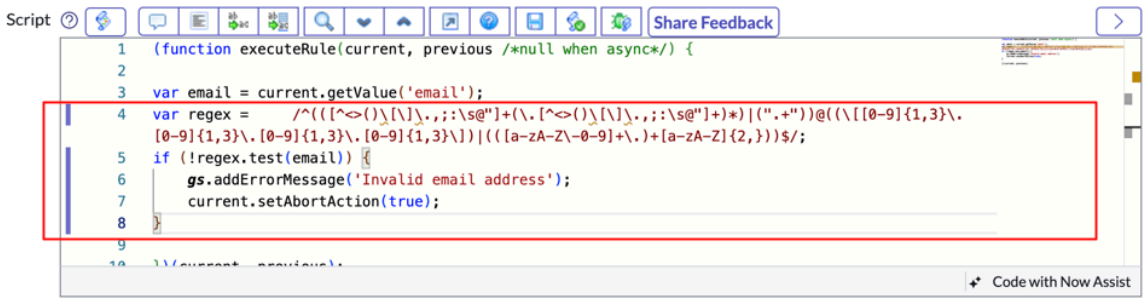
   You can't edit the prompt after triggering the code generation. If you need to edit your prompt before the code suggestion is returned, you can cancel the code generation by pressing the Backspace key.

   The spinner icon ( ) appears while generating a suggestion. The code suggestion appears in the lines following your prompt but isn't added to your script until you accept it.



4. Accept the code to include it in your script or reject it to remove it from the script.
   - Accept: Press the Tab key or the right arrow key. Selecting within the suggested code also accepts the suggestion.
   - Reject: Press the Escape key, left arrow key, or up arrow key. Typing or selecting anywhere outside of the suggested code within the script also removes the suggestion.

   When you accept a code suggestion, a line next to the line numbers indicates which code was created by AI and hasn't been edited. If you edit AI-generated code, the line indicator doesn't appear for those lines of code.



Trouble?
If the code suggestion doesn't meet your requirements, try modifying your code and generating another code suggestion.

5. Select **Submit** or **Update** to save your changes.

Edit code with Now Assist

Edit scripts quickly by telling Now Assist how to improve the code.

Before you begin

Role required: now.assist.creator

About this task

When code generation is enabled on an instance, a Now Assist icon (✨) appears in the script editor.

Procedure

1. Navigate to a script.

   Example
   For example, to open a script include form, navigate to **All** > **System Definition** > **Script Includes** and select a script include.

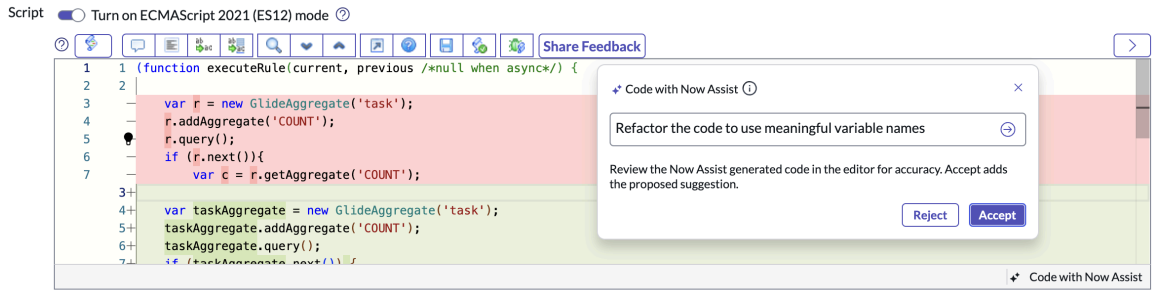
2. In the script editor, highlight the code to edit.
3. Right-click and select **Open Code with Now Assist** or use one of the following keyboard shortcuts:
   - Windows: Ctrl-Enter
   - Mac: Cmd-Enter

   Tip:
   Select the Help icon (?) to access the list of relevant keyboard shortcuts.

4. In the **Edit code with Now Assist** dialog box, enter text that describes how you want to refactor the code.

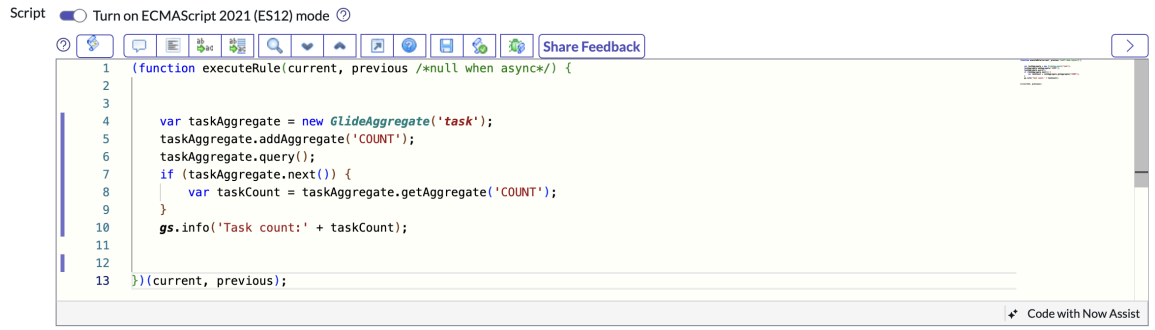
   The text you enter must be fewer than 1,000 characters.

5. Press Enter to generate an edited code suggestion.
   A view comparing the original code and the edited code suggestion appears in the script editor.



6. Review the edited code suggestion and complete one of the following steps:
   - To overwrite the original code with the edited code suggestion, select **Accept**.
   - To regenerate a suggestion, revise the text in the dialog box and select the arrow icon ( ).
   - To remove the edited code from the script and keep only the original code, select **Reject**.

   When you accept a code suggestion, a line next to the line numbers indicates which code was created by AI and hasn't been edited. If you edit AI-generated code, the line indicator doesn't appear for those lines of code.



Trouble?
If the code suggestion doesn't meet your requirements, try rephrasing your prompt according to the prompt guidance and generating another code suggestion.

7. Select **Submit** or **Update** to save your changes.

Code generation reference

Reference topics provide additional information about configuration properties, roles, and more.

Code generation properties

You can adjust how code generation functions on an instance using several advanced properties.

Note:
To open the System Properties [sys_properties] table, enter `sys_properties.list` in the navigation filter.

Properties for code generation

Property	Description
sn_now_assist_code.enable_code_assist	Enables using code generation in supported script editors.

Properties for code generation (continued)

Property	Description
	• Type: true | false • Default value: true • Location: System Property [sys_properties] table • Learn more: You can also enable code generation from Now Assist Admin. For more information, see Activate a Now Assist skill.
sn_now_assist_code.enable_prompt_modal	Enables using the Code with Now Assist dialog box to provide text prompts. • Type: true | false • Default value: true • Location: System Property [sys_properties] table • Learn more: Generate scripts from text
sn_now_assist_code.enable_code_edit	Enables editing code with Now Assist. • Type: true | false • Default value: true • Location: System Property [sys_properties] table • Learn more: Edit code with Now Assist
sn_now_assist_code.show_ai_code_line_marker	Enables tracking which lines of code are AI-generated. This code is indicated by a line next to the lines of code in a script editor. If you edit AI-generated code, the line indicator doesn't appear for those lines of code. • Type: true | false • Default value: true • Location: System Property [sys_properties] table • Learn more: Tracking AI-generated code

Properties for code generation (continued)

Property	Description
sn_now_assist_code.collect_schema_for_code_assist	Specifies whether to collect the data schema of the table a business rule or client script runs on when using code generation. If you set this property to false, you can receive code suggestions quicker but may get less contextual suggestions. • Type: true | false • Default value: true • Location: System Property [sys_properties] table

Code generation roles

The following roles are installed for use with code generation.

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

Now Assist Creator [now.assist.creator]

Write scripts using AI-powered code generation.

Groups

This role is assigned to no groups by default.

Contains Roles

This role contains no roles.

Elevated

This role is not an elevated role.

Special considerations

None

JavaScript syntax editor

The syntax editor provides support for editing JavaScript scripts.

The syntax editor has these features.

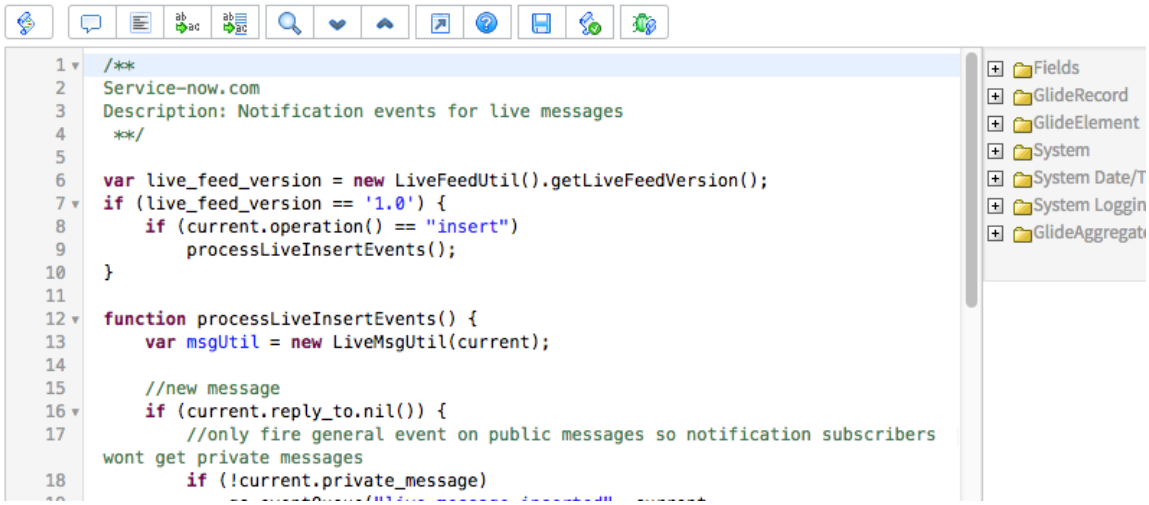
- JavaScript syntax coloring, indentation, line numbers, and automatic creation of closing braces and quotes
- JavaScript support
- Linting using the ESLint utility

Note:
Modify or view default linting configurations by accessing the *glide.ui.syntax_editor.linter.eslint_config* property in the System Property [sys_properties] table. See Available system properties for more information.

- Context menu for script includes, API, and tables
- Script macros for common code shortcuts

This feature requires the Syntax Editor (com.glide.syntax_editor) plugin.

JavaScript editor



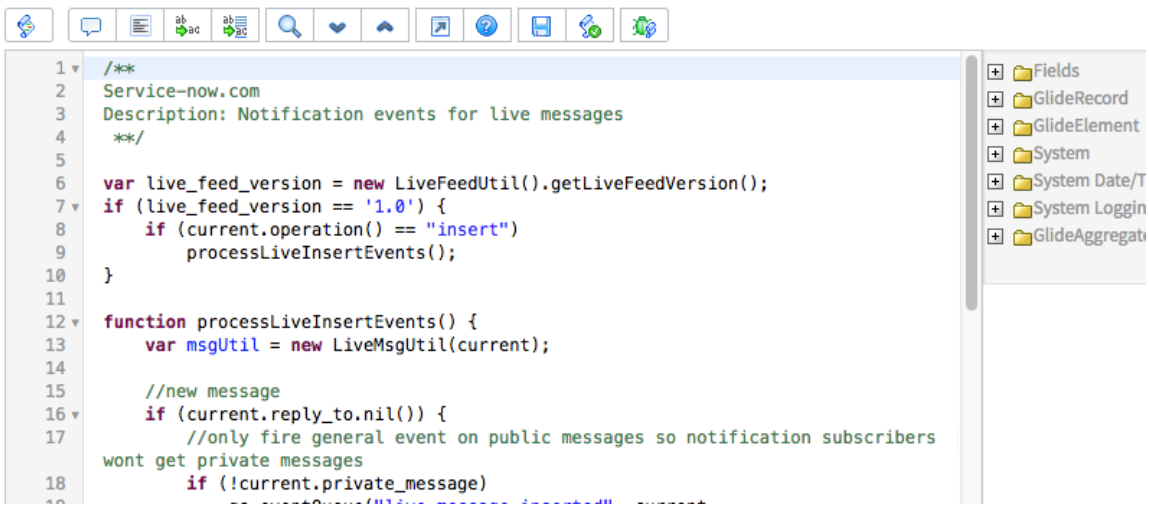
Syntax editor plugin

Enable the syntax editor plugin to use the syntax editor.

The syntax editor enables the following features for all script fields:

- JavaScript syntax coloring, indentation, line numbers, and automatic creation of closing braces and quotes
- Code editing functions
- Code syntax checking
- Script macros for common code shortcuts

JavaScript syntax editor



The syntax editor can be disabled or enabled by modifying the *glide.ui.javascript_editor* property in the sys_properties.list. In addition, administrators can configure the syntax editor to show error and warning indicators next to a line of code that contains an error by modifying the

glide.ui.syntax_editor.show_warnings_errors property. For information on the sys_properties.list, refer to Available system properties.

Note:
Administrators can disable or enable the syntax editor for all users, regardless of user preference.

Syntax editor JavaScript support

The syntax editor provides editing functions to support editing JavaScript scripts.

JavaScript editing functions

Icon	Keyboard Shortcut	Name	Description
	N/A	Toggle Syntax Editor	Disables the syntax editor. Click the button again to enable the syntax editor.
	Access Key + R	Format Code	Applies the proper indentation to the script.
	Access Key + C	Comment Selected Code	Comments out the selected code.
	Access Key + U	Uncomment Selected Code	Removes comment codes from the selected code.
	N/A	Check Syntax	Checks the code for syntax errors. By default, the system automatically checks for syntax errors as you type in a script field. If an error or warning is found, the syntax editor displays a bullet beside the script line containing the error or warning. This check occurs on all script fields.
	Access Key + \	Start Searching	Highlights all occurrences of a search term in the script field and locates the first occurrence. Click the icon, then enter the search term and press **Enter**. You can use regular expressions enclosed in slashes to define the search term. For example, the term */a{3}/* locates *aaa*.
	Access Key + [	Find Next	Locates the next occurrence of the current search term in the script field. Use *Start Searching* to change the current search term.
	Access Key +]	Find Previous	Locates the previous occurrence of the current search term in the script field. Use *Start Searching* to change the current search term.
	Access Key + W	Replace	Replaces the next occurrence of a text string in the script field. **1.** Click the icon, then enter the string to replace and press **Enter**. You can use regular expressions enclosed in slashes to define the string to replace. For example, the term */a{3}/* locates *aaa*. **2.** Enter the replacement string and press **Enter**.
	Access Key + ;	Replace All	Replaces all occurrences of a text string in the script field.

Icon	Keyboard Shortcut	Name	Description
			1. Click the icon, then enter the string to replace and press **Enter**. You can use regular expressions enclosed in slashes to define the string to replace. For example, the term */a{3}/* locates *aaa*. 2. Enter the replacement string and press **Enter**.
	N/A	Save	Saves changes without leaving the current view. Use this button in full screen mode to save without returning to standard form view.
	Access Key + L	Toggle Full Screen Mode	Expands the script field to use the full form view for easier editing. Click the button again to return to standard form view. This feature is not available for Internet Explorer.
	Access Key + P	Help	Displays the keyboard shortcuts help screen.

JavaScript editing tips

- To fold a code block, click the minus sign beside the first line of the block. The minus sign only appears beside blocks that can be folded. To unfold the code block, click the plus sign.
- To insert a fixed space anywhere in your code, press Tab.
- To indent a single line of code, click in the leading white space of the line and then press Tab.
- To indent one or more lines of code, select the code and then press Tab. To decrease the indentation, press Shift + Tab.
- To remove one tab from the start of a line of code, click in the line and press Shift + Tab.

JavaScript resources

Scripts use ECMA 262 standard JavaScript. Helpful resources include:

- Mozilla: http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference
- ECMA Standard in PDF format: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf
- History and overview: http://javascript.crockford.com/survey.html
- JavaScript number reference: http://www.hunlock.com/blogs/The_Complete_Javascript_Number_Reference

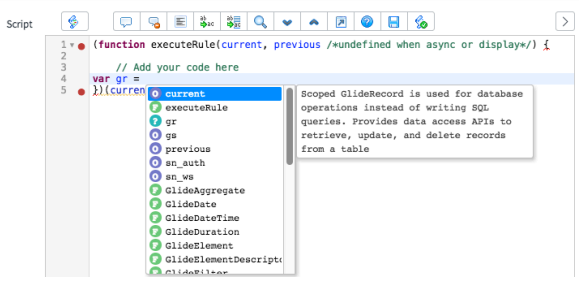
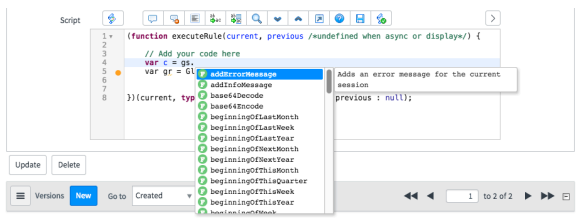
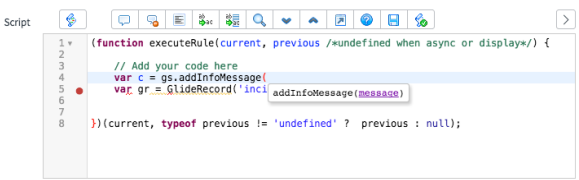
Syntax editor keyboard shortcuts and actions

The syntax editor offers keyboard shortcuts and actions to assist in writing code.

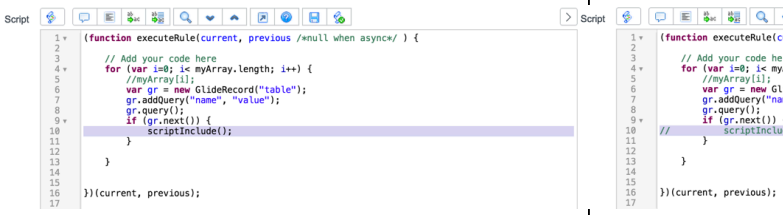
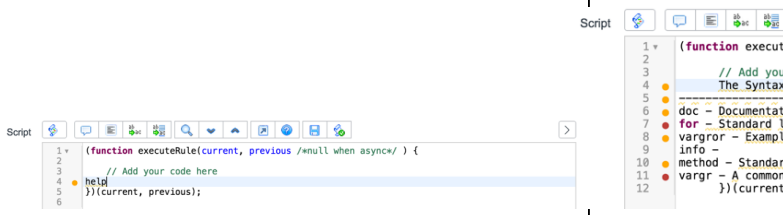
Syntax editor keyboard shortcuts and actions for writing code

Keyboard shortcut or action	Description	Example
Write code		

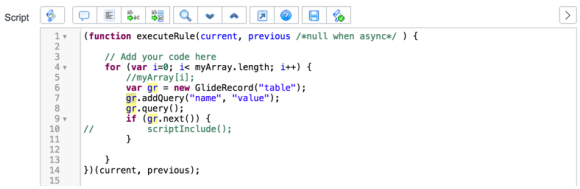
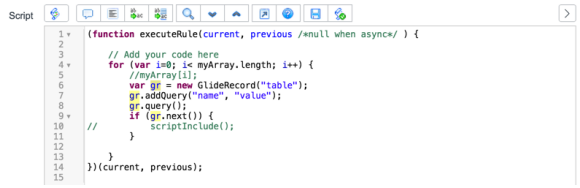
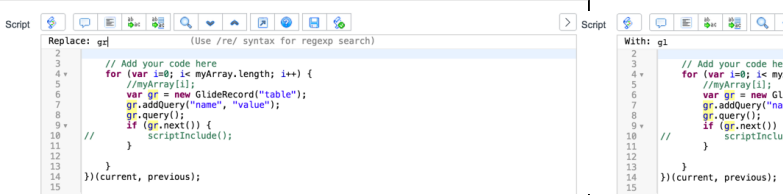
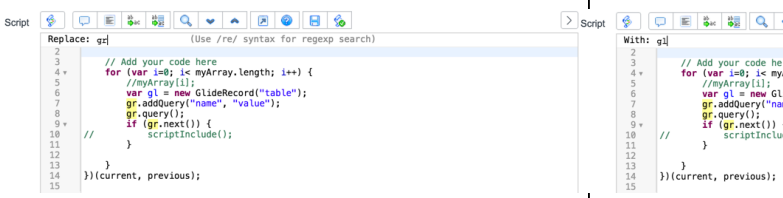
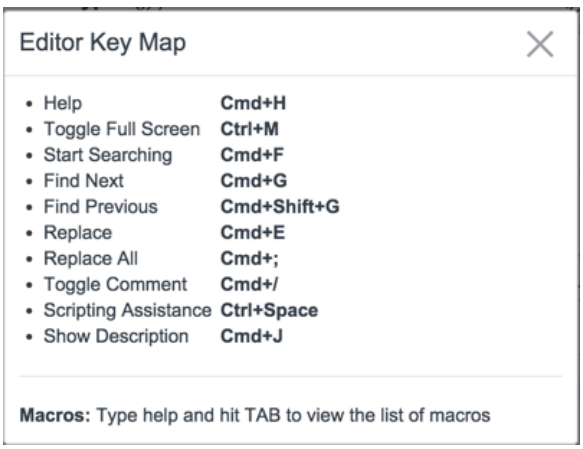
Syntax editor keyboard shortcuts and actions for writing code (continued)

Keyboard shortcut or action	Description	Example
Scripting assistance Control +Spacebar	Displays a list of valid elements at the insertion point such as: • Class names • Function names • Object names • Variable names Double-click an entry to add it to the script. **Note:** The elements are based on server or client type of script. However, these elements are available based on the UI type you select. For example, spUtil is available for Service Portal client scripts and g_navigation is available for Desktop scripts.	
Enter a period character after a valid class name.	Displays a list methods for the class. Double-click an entry to add it to the script.	
Enter an open parenthesis character after a valid class, function, or method name.	Displays the expected parameters for the class or method. Enter the expected parameters as needed.	
Toggle full screen mode Control+M	Switches between displaying the form with the full screen and displaying it normally.	

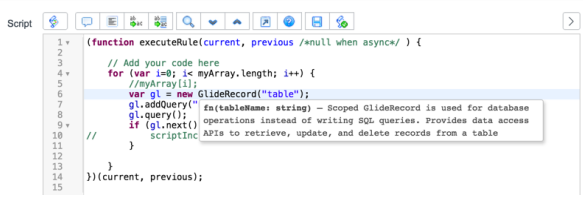
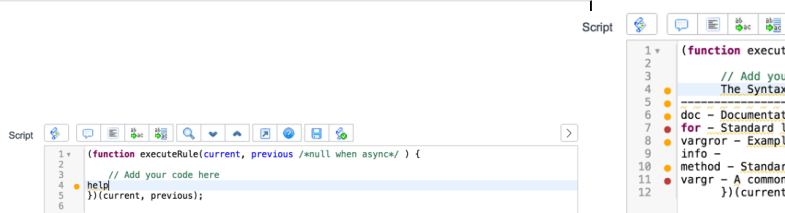
Syntax editor keyboard shortcuts and actions for writing code (continued)

Keyboard shortcut or action	Description	Example
Format code • Windows: Control +Shift+B • Mac: Command +Shift+B	Formats the selected lines to improve readability.	
Toggle comment • Windows: Control+/ • Mac: Command +/	Adds or removes the comment characters `//` from the selected lines.	
Insert macro text **1.** In the **Script** field, type the macro keyword text. For example `help`. **2.** Press Tab.	Inserts macro text at the current position.	
Search		
Start search • Windows: Control+F • Mac: Command +F	Highlights all occurrences of a search term in the script field and locates the first occurrence. You can create regular expressions by enclosing the search terms between slash characters . For example, the search term `/a{3}/` locates the string `aaa` .	

Syntax editor keyboard shortcuts and actions for writing code (continued)

Keyboard shortcut or action	Description	Example
Find next • Windows: Control+G • Mac: Command +G	Locates the next occurrence of the current search term in the script field. Use **Start Searching** to change the current search term.	
Find previous • Windows: Control +Shift+G • Mac: Command +Shift+G	Locates the previous occurrence of the current search term in the script field. Use **Start Searching** to change the current search term.	
Replace • Windows: Control+E • Mac: Command +E	Replaces the next occurrence of a text string in the script field.	
Replace all • Windows: Control+; • Mac: Command +;	Replaces all occurrences of a text string in the script field.	
Help		
Help • Windows: Control+H • Mac: Command +H	Displays the list of syntax editor keyboard shortcuts.	

Syntax editor keyboard shortcuts and actions for writing code (continued)

Keyboard shortcut or action	Description	Example
Show description • Windows: Control+J • Mac: Command +J	Displays API documentation for the scripting element at the cursor's current location.	
Show macros **1.** In the **Script** field, type `help`. **2.** Press Tab.	Displays the list of available syntax editor macros as text within the script field.	

Syntax editor macros

Script macros provide shortcuts for typing commonly used code. To insert macro text into a script field, enter the macro keyword followed by the Tab.

Syntax editor macros are defined in the Editor Macros [syntax_editor_macro] table. To create or modify a script macro, see Create or modify a script macro.

vargr

- Inserts a standard *GlideRecord* query for a single value.
- Output:

```
var gr = new GlideRecord("$0");
gr.addQuery("name", "value");
gr.query();
if (gr.next()) {

}
```

vargror

- Inserts a *GlideRecord* query for two values with an OR condition.
- Output:

```
var gr = new GlideRecord('$0');

var qc = gr.addQuery('field', 'value1');

qc.addOrCondition('field', 'value2');
gr.query();
```

```
while (gr.next()) {


}
```

for

- Inserts a standard recursive loop with an array.
- Output:

```
for (var i=0; i< myArray.length; i++) {
 //myArray[i];

}
```

info

- Inserts a *GlideSystem* information message.
- Output:

```
gs.addInfoMessage("");
```

method

- Inserts a blank JavaScript function template.
- Output:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~
   * Description:
   * Parameters:
   * Returns:

 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~*/
: function() {

},
```

doc

- Inserts a comment block for describing a function or parameters.
- Output:

```
/**

 * Description:

 * Parameters:

 * Returns:
*/
```

Create or modify a script macro

Administrators can define new script macros or modify existing script macros.

Before you begin
Role required: admin

About this task
Script macros provide shortcuts for typing commonly used code. Several script macros are available by default. Administrators can define new or modify existing script macros.

Procedure
1. Navigate to **All** > **System Definition** > **Syntax Editor Macros**.
2. Click **New** or select the macro to edit.
3. Define the macro details with the fields listed in the table below.

Editor macro fields

Field	Description
Name	Macro keyword text users type to insert macro text.
Comments	Description of the macro. This text appears when the user types *help*.
Text	Full macro text that replaces the name in the editor.

Context menu in the syntax editor
View the context menu for script includes, Glide APIs, and tables in the JavaScript syntax editor.

With the context menu options, your users can navigate to:

- Script include definitions
- Glide API documentation
- System and custom table definitions and data

In the syntax editor, bold font is used for tokens that have a context menu. Right-click the token to view context menu options. If you use a Mac, you can use the Command-click shortcut.

Context menu options

Token type	Context menu option	Description
Script include	Open Definition	Definition of the script include in a new window.
	Find References	List of files referencing the script include.

Context menu options (continued)

Token type	Context menu option	Description
		Note: • To view the preview of the file, click the preview script icon. To open the file in a new window, click **Open File**. • To view all files that reference the script include, click **Show All Files**.
Glide API	Show Documentation	Documentation page of the Glide API.
Table	Show Definition	Definition of the system or custom table in a new window.
	Show Data	Records in the table that are based on the role of the current user.

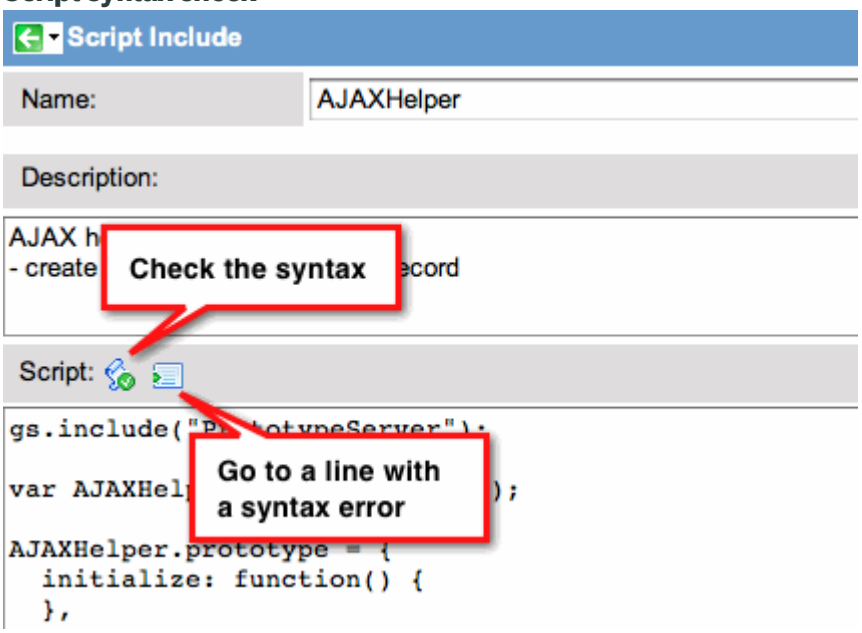
Enable or disable the context menu in the script editor using the *glide.ui.syntax_editor.context_menu* property in the System Property [sys_properties] table. See Available system properties for more information.

Note:
Context menu options can be accessed only if the browser supports SharedWorker. For example, Google Chrome and Mozilla Firefox.

Script syntax error checking

All script fields provide controls for checking the syntax for errors and for locating the error easily when one occurs. The script editor places the cursor at the site of a syntax error and lets you search for errors in scripts by line number.

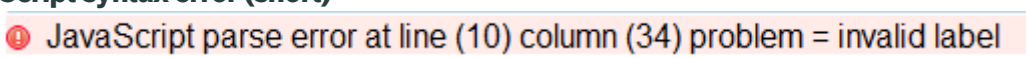
Script syntax check



The script editor notifies you of syntax errors in your scripts in the following situations.

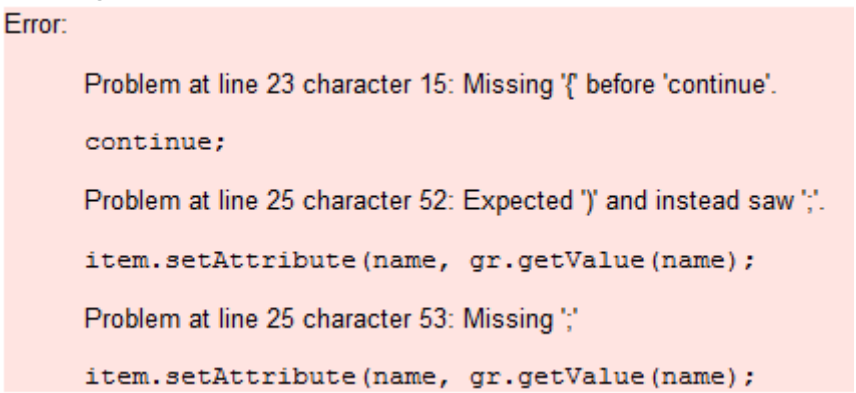
- Save a new record or update an existing record. A banner appears at the bottom of the editor showing the location of the first error (line number and column number), and the cursor appears at the site of the error. Warnings presented at **Save** or **Update** show only one error at a time.

  Script syntax error (short)

- Click the syntax checking icon before saving or updating a record. A banner appears at the bottom of the editor showing the location of all errors in the script, and the cursor appears at the site of the first error.

  Script syntax error

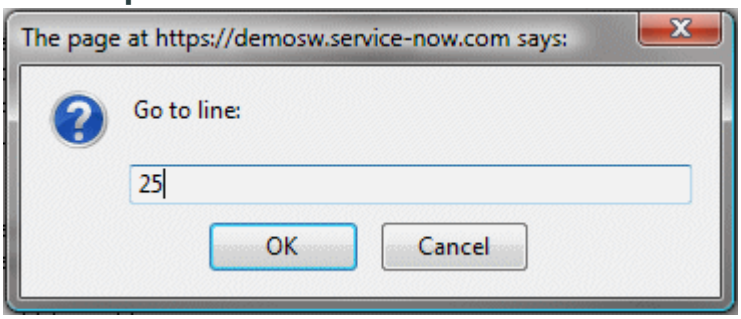
Searching for errors by line

To locate the exact position of the error in a large script, click the **Go to line** icon.

This feature is particularly useful when you are encounter a syntax error in a log file rather than in the ServiceNow record itself. In this case, you can navigate to the record and search for errors by line number. In the dialog box that appears, enter the line number of an error, and then click **OK**. Your view moves to the site of the error, and the cursor marks the correct line and column.

Note:
For this feature to function, you must disable the Syntax Editor.

Go to script error



Navigate to a line number

When the syntax editor is disabled, users can navigate to a specific line in the code using the Go to line icon ( ).

Before you begin

Disable the Syntax Editor.

Role required: admin

Procedure

1. Click the Go to line icon ( ).

   Note:
   This icon is not available when the editor is enabled.

2. Enter a number in the field and then press **Enter**.

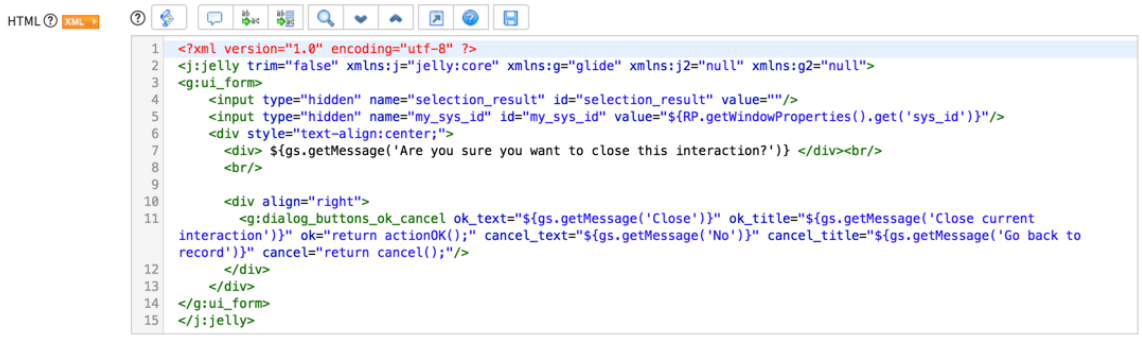
HTML syntax editor

The HTML syntax editor provides support for editing HTML and Jelly scripts and defines what's rendered when the page is displayed. The HTML syntax editor can contain either static XHTML or dynamically generated content defined as Jelly, and can call script includes and UI Macros.

The syntax editor has these features.

- HTML and Jelly script support
- HTML and Jelly syntax coloring, indentation, line numbers, and automatic creation of closing braces and quotes
- Auto-suggestions for HTML and Jelly tags
- Script macros for common code shortcuts

HTML syntax editor



HTML and Jelly editing functions

Icon	Keyboard shortcut	Name	Description
	N/A	Toggle syntax editor	Disables the syntax editor. Click the Toggle syntax editor icon ( ) again to enable the syntax editor.
	Cmd+/	Toggle comment	Comments the selected code.
	Cmd+E	Replace	Replaces the next occurrence of a text string in the script field. **1.** Click the Replace icon ( ), then enter the string to replace, and press#Enter. You can use regular expressions enclosed in slashes to define the string to replace. For example, the term `/a{3}/` locates `aaa`. **2.** Enter the replacement string and press#Enter.
	Cmd	Replace All	Replaces all occurrences of a text string in the script field. **1.** Click the Replace all icon ( ), then enter the string to replace and press#Enter. You can use regular expressions enclosed in slashes to define the string to replace. For example, the term `/a{3}/` locates `aaa`. **2.** Enter the replacement string and press#Enter.
	Cmd+F	Start Searching	Highlights all occurrences of a search term in the script field and locates the first occurrence. Click the Start searching icon ( ), then enter the search term and press Enter.
	Cmd+G	Find Next	Locates the next occurrence of the current search term in the script field. Click the Start searching icon ( ) to change the current search term.
	Cmd +Shift +G	Find Previous	Locates the previous occurrence of the current search term in the script field. Click the Start searching icon ( ) to change the current search term.

HTML and Jelly editing functions (continued)

Icon	Keyboard shortcut	Name	Description
	Ctrl+M	Toggle Full Screen	Expands the script field to use the full form view for easier editing. Click the Toggle full screen icon ( ) again to return to standard form view. This feature is not available for Internet Explorer.
	Cmd+H	Help	Displays the keyboard shortcuts help screen.
	N/A	Save	Saves changes without leaving the current view. Click the Save icon ( ) in full screen mode to save without returning to standard form view.

Editing tips

- To insert a fixed space anywhere in your code, press Tab.
- To indent a single line of code, click in the leading white space of the line and then press Tab.
- To indent one or more lines of code, select the code and then press Tab. To decrease the indentation, press Shift+Tab.
- To remove one tab from the start of a line of code, click in the line and press Shift+Tab.

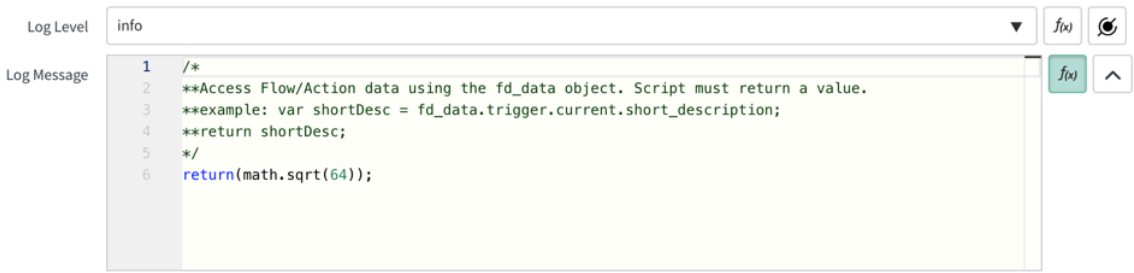
Code editor

The code editor provides support to use programming language services in a text editor and is used in scripts.

The code editor has these features for the supported language services and Inline scripts.

- Syntax coloring, indentation, line numbers, and automatic creation of closing braces and quotes
- Auto-suggestions and auto-completions

Code editor



Editing tips

- To insert a fixed space anywhere in your code, press Tab.
- To indent a single line of code, click in the leading white space of the line and then press Tab.
- To indent one or more lines of code, select the code and then press Tab. To decrease the indentation, press Shift+Tab.

- To remove one tab from the start of a line of code, click in the line and press Shift+Tab.
- To declare variables, use the `var` keyword so that they remain within the proper JavaScript scope.

Server-side scripting

Server scripts run on the server or database. They can change the appearance or behavior of ServiceNow or run as business rules when records and tables are accessed or modified.

Server-side Glide APIs (Application Programming Interfaces) provide classes and methods that you can use in scripts to perform server-side tasks.

Immediately invoked function expressions

The system uses immediately invoked function expressions when a script runs in a single context, such as in a Create a transform map. Functions that run from multiple contexts use Script includes instead.

By enclosing a script in an immediately invoked function expression, you can:

- Ensure that the script does not impact other areas of the product, such as by overwriting global variables.
- Pass useful variables or objects as parameters.
- Identify function names in stack traces.
- Eliminate having to make separate function calls.

An immediately invoked function expression follows this format:

```
(function functionName(parameter){

  //The script you want to run

})('value');//Note the parenthesis indicating this function
 should run.
```

You can declare functions within the immediately invoked function expression. These inner functions are accessible only from within the immediately invoked function expression.

```
(function functionName(parameter){

  function helperFunction(parameter){//return some value}

  var value = helperFunction(parameter);//Valid function call.

  //perform any other script actions

})('value');

var value2 = helperFunction(parameter);//Invalid. This function
 is not accessible from outside the self-executing function.
```

Glide Server APIs

ServiceNow provides APIs for the Glide Server.

GlideAggregate

The *GlideAggregate* class is an extension of *GlideRecord* and allows database aggregation (COUNT, SUM, MIN, MAX, AVG) queries to be done. This can be helpful in creating customized reports or in calculations for calculated fields.

Note:
This functionality requires a knowledge of JavaScript.

For additional information, refer to GlideAggregate API.

GlideAggregate examples

GlideAggregate is an extension of *GlideRecord* and its use is probably best shown through a series of examples.

Note:
This functionality requires a knowledge of JavaScript.

Here is an example that simply gets a count of the number of records in a table:

```
var count = new GlideAggregate('incident');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if(count.next())
   incidents = count.getAggregate('COUNT');
```

There is no query associated with the preceding example. If you want to get a count of the incidents that were open, simply add a query as is done with *GlideRecord*. Here is an example to get a count of the number of active incidents.

```
var count = new GlideAggregate('incident');
count.addQuery('active','true');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if(count.next())
   incidents = count.getAggregate('COUNT');
```

To get a count of all the open incidents by category the code is:

```
var count = new GlideAggregate('incident');
count.addQuery('active','true');
count.addAggregate('COUNT','category');
count.query();
while(count.next()){
  var category = count.category;
  var categoryCount = count.getAggregate('COUNT','category');
  gs.log("The are currently "+ categoryCount +" incidents with a
 category of "+ category);}
```

The output is:

```
 *** Script: The are currently 1.0 incidents with a category of
 Data
       *** Script: The are currently 11.0 incidents with a
 category of Enhancement
```

```
      *** Script: The are currently 1.0 incidents with a
 category of Implementation
      *** Script: The are currently 197.0 incidents with a
 category of inquiry
      *** Script: The are currently 13.0 incidents with a
 category of Issue
      *** Script: The are currently 1.0 incidents with a
 category of
      *** Script: The are currently 47.0 incidents with a
 category of request
```

The following is an example that uses multiple aggregations to see how many times records have been modified using the *MIN*, *MAX*, and *AVG* values.

```
var count = new GlideAggregate('incident');
count.addAggregate('MIN','sys_mod_count');
count.addAggregate('MAX','sys_mod_count');
count.addAggregate('AVG','sys_mod_count');
count.groupBy('category');
count.query();
while(count.next()){
  var min = count.getAggregate('MIN','sys_mod_count');
  var max = count.getAggregate('MAX','sys_mod_count');
  var avg = count.getAggregate('AVG','sys_mod_count');
  var category = count.category.getDisplayValue();
  gs.log(category +" Update counts: MIN = "+ min +" MAX = "+ max
 +" AVG = "+ avg);}
```

The output is:

```
      *** Script: Data Import Update counts: MIN = 4.0 MAX =
 21.0 AVG = 9.3333
      *** Script: Enhancement Update counts: MIN = 1.0 MAX =
 44.0 AVG = 9.6711
      *** Script: Implementation Update counts: MIN = 4.0 MAX =
 8.0 AVG = 6.0
      *** Script: inquiry Update counts: MIN = 0.0 MAX = 60.0
 AVG = 5.9715
      *** Script: Inquiry / Help Update counts: MIN = 1.0 MAX =
 3.0 AVG = 2.0
      *** Script: Issue Update counts: MIN = 0.0 MAX = 63.0 AVG
 = 14.9459
      *** Script: Monitor Update counts: MIN = 0.0 MAX = 63.0
 AVG = 3.6561
      *** Script: request Update counts: MIN = 0.0 MAX = 53.0
 AVG = 5.0987
```

The following is a more complex example that shows how to compare activity from one month to the next.

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count','category');
agg.orderBy('category');
agg.addQuery('opened_at','>=','javascript:gs.monthsAgoStart(2)
');
agg.addQuery('opened_at','<=','javascript:gs.monthsAgoEnd(2)');
```

```
agg.query();
while(agg.next()){
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count','category');
  agg2.orderBy('category');

 agg2.addQuery('opened_at','>=','javascript:gs.monthsAgoStart(3)
');

 agg2.addQuery('opened_at','<=','javascript:gs.monthsAgoEnd(3)
');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last ="";
  while(agg2.next()){
     last = agg2.getAggregate('count','category');}
  gs.log(category +": Last month:"+ count +" Previous Month:"+
 last);

}
```

The output is:

```
 *** Script: Monitor: Last month:6866.0 Previous Month:4468.0
 *** Script: inquiry: Last month:142.0 Previous Month:177.0
 *** Script: request: Last month:105.0 Previous Month:26.0
 *** Script: Issue: Last month:8.0 Previous Month:7.0
 *** Script: Enhancement: Last month:5.0 Previous Month:5.0
 *** Script: Implementation: Last month:1.0 Previous Month:0
```

The following is an example to obtain distinct count of a field on a group query.

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count');
agg.addAggregate('count(distinct','category');
agg.addQuery('opened_at', '>=',
 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=',
 'javascript:gs.monthsAgoEnd(2)');
//
agg.groupBy('priority');
agg.query();
while (agg.next()) {
// Expected count of incidents and count of categories within
 each priority value (group)
  gs.info('Incidents in priority ' + agg.priority + ' = ' +
 agg.getAggregate('count') +
            ' (' + agg.getAggregate('count(distinct','category')
 + ' categories)');
}
```

The output is:

```
*** Script: Incidents in priority 1 = 13 (3 categories)
*** Script: Incidents in priority 2 = 10 (5 categories)
*** Script: Incidents in priority 3 = 5 (3 categories)
*** Script: Incidents in priority 4 = 22 (6 categories)
```

You can implement the SUM aggregate with or without the use of the *groupBy()* method. If you do not use the *groupBy()* method, the result of the SUM is the cumulative value for each different value of the field for which you request the SUM. For example, if you SUM the total_cost field in the Fixed Asset table, and the Fixed Asset table contains 12 total records:

- Three records with a total_cost of $12
- Four records with a total_cost of $10
- Five records with a total_cost of $5

When you SUM the record set, the *getAggregate()* method returns three different sums: $36, $40, and $25.

The following code illustrates implementing the SUM aggregate without using the *groupBy()* method:

```
var totalCostSum = new GlideAggregate('fixed_asset');
totalCostSum.addAggregate('SUM', 'total_cost');
totalCostSum.query();

while (totalCostSum.next()) {
  var allTotalCost = 0;
  allTotalCost = totalCostSum.getAggregate('SUM', 'total_cost');
  aTotalCost = totalCostSum.getValue('total_cost');
  gs.print('Unique field value: ' + aTotalCost + ', SUM = ' +
 allTotalCost + ', ' + allTotalCost/aTotalCost + ' records');
}
```

The output for this example is:

```
*** Script: Unique field value: 12, SUM = 36, 3 records
*** Script: Unique field value: 10, SUM = 40, 4 records
*** Script: Unique field value: 5, SUM = 25, 5 records
```

Using the same data points as the prior example, if you use the *groupBy()* method, the SUM aggregate returns the sum of all values for the specified field.

The following example illustrates implementing the SUM aggregate using the *groupBy()* method:

```
var totalCostSum = new GlideAggregate('fixed_asset');
totalCostSum.addAggregate('SUM', 'total_cost');
totalCostSum.groupBy('total_cost');
totalCostSum.query();
if(totalCostSum.next()){  // in case there is no result
   var allTotalCost = 0;
   allTotalCost = totalCostSum.getAggregate('SUM',
 'total_cost');
   gs.print('SUM of total_cost: = ' + allTotalCost);
}
```

The output for this example is:

```
*** Script: SUM of total_cost: 101
```

GlideRecord

GlideRecord is a special Java class (*GlideRecord.java*) that can be used in JavaScript exactly as if it was a native JavaScript class.

GlideRecord:

- is used for database operations instead of writing SQL queries.
- is an object that contains zero or more records from one table. Another way to say this is that a GlideRecord is an ordered list.

A GlideRecord contains both records (rows) and fields (columns). The field names are the same as the underlying database column names. For additional information, refer to GlideRecord - Scoped.

Note:
Use of `gs.sql())` scripting syntax was discontinued in Geneva. Use standard *GlideRecord* syntax in its place.

Using GlideRecordSecure

GlideRecordSecure is a class inherited from *GlideRecord* that performs the same functions as *GlideRecord*, and also enforces ACLs.

Non-writable fields

Be aware that, when using *GlideRecordSecure*, non-writable fields are set to NULL when trying to write to the database. By default, *canCreate()* on the column is replaced with *canWrite()* on the column. If that returns false, the column value is set to NULL.

Getting the object type

You can check if an object type is ScopedGlideRecordSecure by calling the *toString()* method.

Checking the returned GlideRecord real type

The current object can be passed as *ScopedGlideRecordSecure*. In most cases, you can call *current.toString()* to return the current object type.

The following line returns `[object ScopedGlideRecordSecure]` for a *ScopedGlideRecordSecure* object:

```
gs.info(current.toString());
```

If the returned object type is different than expected

Calling a scoped function that returns a GlideRecord object as its type returns different results depending on the scope it's called from.

- Calling the scoped function within application scope returns the expected object type of `GlideRecord` or `GlideRecordSecure`.
- Calling the scoped function within global scope returns the object type is `ScopedGlideRecord` or `ScopedGlideRecordSecure`.

Checking for NULL values

If an element cannot be read because an ACL restricts access, a NULL value is created in memory for that record. With GlideRecord, you must explicitly check for any ACLs that might restrict read access to the record. To do so, an if statement such as the following is required to check if the record can be read:

```
if ( !grs.canRead() ) continue;
```

With *GlideRecordSecure*, you do not need to explicitly check for read access using *canRead()*. Instead, you can use *next()* by itself to move to the next record. The following example provides a comparison between *GlideRecord* and *GlideRecordSecure*.

```
var count  = 0;
var now_GR  = new GlideRecord('mytable');
now_GR. query();
while (now_GR. next()) {
    if (!now_GR. canRead()) continue;
    if (!now_GR. canWrite()) continue;
    if (!now_GR. val. canRead() || !now_GR. val. canWrite())
        now_GR. val = null;
    else
        now_GR. val = "val-" + now_GR. id;
    if (now_GR. update())
        count ++;
}
```

```
var count  = 0;
var grs  = new GlideRecordSecure('mytable');
grs. query();
while (grs. next()) {
    grs. val = "val-" + grs. id;
    if (grs. update())
        count ++;
}
```

Examples

These are two simple examples using *GlideRecordSecure*.

```
var att  = new GlideRecordSecure('sys_attachment');
att. get('$[sys_attachment.sys_id]');
var sm  = GlideSecurityManager.get();
var checkMe  = 'record/sys_attachment/delete';
var canDelete  = sm.hasRightsTo(checkMe,att);
gs. log('canDelete: ' + canDelete);
canDelete;
```

```
var grs = new GlideRecordSecure('task_ci');
grs.addQuery();
grs.query();
var count  = grs. getRowCount();
if (count  > 0 ) {
    var allocation  = parseInt(10000/count) / 100;
    while (grs.next()) {
      grs.u_allocation = allocation;
      grs.update();
```

```
    }
}
```

GlideSystem

The `GlideSystem` API provides methods for retrieving information.

The `GlideSystem` (referred to by the variable name '`gs`' in business rules) provides a number of convenient methods to get information about the system, the current logged in user, etc. For example, the method `addInfoMessage()` permits communication with the user.

```
gs.addInfoMessage('Email address added for notification');
```

Many of the `GlideSystem` methods facilitate the easy inclusion of dates in query ranges and are most often used in filters and reporting.

For additional information, see GlideSystem.

GlideDateTime

The `GlideDateTime` class provides methods for performing operations on `GlideDateTime` objects, such as instantiating `GlideDateTime` objects or working with `glide_date_time` fields.

In addition to the instantiation methods described below, a GlideDateTime object can be instantiated from a `glide_date_time` field using the `getGlideObject()` method (for example, `var gdt = gr.my_datetime_field.getGlideObject();`).

Some methods use the Java Virtual Machine time zone when retrieving or modifying a date and time value. Using these methods may result in unexpected behavior. Use equivalent local time and UTC methods whenever possible.

GlideDate and GlideDateTime examples

The `GlideDate` and `GlideDateTime` APIs are used to manipulate date and time values.

Note:
This functionality requires a knowledge of JavaScript.

For additional information, refer to GlideDate API and GlideDateTime API.

You can create a `GlideDateTime` object from a `GlideDate` object by passing in the `GlideDate` object as a parameter to the `GlideDateTime` constructor. By default, the `GlideDateTime` object is expressed in the internal format, yyyy-MM-dd HH:mm:ss and the system time zone UTC.

```
var gDate = new GlideDate();
gDate.setValue('2015-01-01');
gs.info(gDate);

var gDT = new GlideDateTime(gDate);
gs.info(gDT);
```

Output:

```
2015-01-01
2015-01-01 00:00:00
```

See also Modify a GlideDateTime field value.

Set a duration field value in script

Examples of JavaScript that can be used to set the value of a duration field.

Note:
Negative duration values are not supported.

Using the GlideDateTime.subtract() method

The *subtract(GlideDateTime start, GlideDateTime end)* method in GlideDateTime enables you to set the duration value using a given start date/time and end date/time. An example on how to set the duration for the time a task was opened is:

```
var duration = GlideDateTime.subtract(start, end);
```

If you want to work with the value returned as a number to use in date or duration arithmetic, convert the return to milliseconds:

```
var time = GlideDateTime.subtract(start,end).getNumericValue();
```

If you want to set a duration to the amount of time between some event and the current date/time:

```
<duration_field> = GlideDateTime.subtract(new
 GlideDateTime(<start_time>.getValue()),gs.nowDateTime());
```

The time values presented to *GlideDateTime.subtract* are expected to be in the user's time zone and in the user's format.

Setting a default value of a duration field

Setting the default value for a duration field is similar to the method used in the previous topic.

Setting the duration field value in a client script

This script sets a *duration_field* value in a client script. Replace *duration_field* with the field name from your instance.

```
g_form.setValue('<duration_field>','11 01:02:03');
```

Calculating and setting a duration using a client script

Here is an example of how to return a value and populate it using a client script.

Create an `onChange` client script that includes the following code. You can modify this script if you need the calculation to happen in an `onLoad` script or some other way.

```
function onChange(control, oldValue, newValue, isLoading){
var strt = g_form.getValue('<start_field>');
var end = g_form.getValue('<end_field>');
var ajax = new GlideAjax('AjaxDurCalc');
  ajax.addParam('sysparm_name','durCalc');
  ajax.addParam('sysparm_strt',strt);
  ajax.addParam('sysparm_end',end);
  ajax.getXMLWait();
  var answer = ajax.getAnswer();
  g_form.setValue('<duration_field>', answer);}
```

Create a system script include file called *AjaxDurCalc* that handles the request. It may be reused for other functions as well.

```
var AjaxDurCalc = Class.create();
AjaxDurCalc.prototype =
 Object.extendsObject(AbstractAjaxProcessor,{
 durCalc:function(){return
 GlideDuration.subtract(this.getParameter('sysparm_strt'),this.g
etParameter('sysparm_end'));}});
```

Changing the duration field value

If you manipulate a duration value with addition/subtraction of some amount of time, use the functions that allow you to get and set the numeric value of the duration. A unit of measure for a duration numeric value is milliseconds. The following is an example that adds 11 seconds to the *duration* field in the current record.

```
var timems = current.duration.dateNumericValue();
timems = timems + 11*1000;
current.duration.setDateNumericValue(timems);
```

Formatting the Resolve Time

To format the **Resolve Time** or the **Business Resolve Time** fields as durations, which displays them as a duration instead of a large integer, add the following attribute to those fields:

```
format=glide_duration
```

Modify the dictionary entry for the field and add the attribute. If there is an existing attribute, separate multiple attributes with commas.

Setting the maximum unit of measurement

The *max_unit* dictionary attribute defines the maximum unit of time used in a duration. For example, if `max_unit=minutes`, a duration of 3 hours 5 minutes 15 seconds appears as 185 minutes 15 seconds. To set the maximum unit of duration measurement, add the following dictionary attribute to the *duration* field:

```
max_unit=<unit>
```

Date and time format guidelines

You can specify a date format with a sequence of specific date and time pattern strings. A pattern string consists of one or more uppercase and lowercase letters from A to Z. Any text within quotation marks is ignored and is instead copied into the date output.

String	Description	Output Format	Example
G	Era designator	Text	AD
y	Year	Year	2019; 19
Y	Week in year	Year	2019; 19
M	Month in year (within date)	Month	July; Jul; 07
L	Month in year (standalone value)	Month	July; Jul; 07
w	Week in year	Number	52
W	Week in month	Number	1
D	Day in year	Number	365

String	Description	Output Format	Example
d	Day in month	Number	2
F	Day of week in month	Number	3
E	Day name in week	Text	Wednesday; Wed
u	Day number of week	Number	3
a	a.m. or p.m.	Text	p.m.
H	Hour in day from 0 through 23	Number	0
k	Hour in day from 1 through 24	Number	24
K	Hour in a.m. or p.m. from 0 through 11	Number	0
h	Hour in a.m. or p.m. from 1 through 12	Number	12
m	Minute in hour	Number	59
s	Second in minute	Number	1
S	Millisecond	Number	500
z	Time zone in default format	Time zone in default format	Pacific Standard Time; PST
Z	Time zone in RFC 822 format	Time zone in RFC 822 format	-0800
X	Time zone in ISO 8601 format	Time zone in ISO 8601 format	-08; -0800; -08:00

Classic Business rules

A business rule is a server-side script that runs when a record is displayed, inserted, updated, or deleted, or when a table is queried.

Business rules are scripts that run when certain server-side conditions are met. Business rule conditions include when to run a business rule in relation to a database operation, and what record operations the business rule applies to. There are other scripting options available on the platform for client-side conditions, such as client scripts and UI actions.

Note:
Business rules are a classic automation solution that rely on scripting. Use Workflow Studio for any new process automation to create automations that are easier to extend, reuse, understand, and upgrade. As many organizations have business rules in production, use this documentation to learn how to work with existing business rules.

How business rules work

To configure business rules, you first need to determine when the business rule should run and what action it should take.

When business rules run

Business rules run based on two sets of criteria.

- When to run the business rule in relation to a database operation.
- What record operation the business rule applies to.

The following options are provided to determine when the business rule should run.

When the business rule should run

Option	When the rule runs
Before	After the user submits the form but before any action is taken on the record in the database.
After	After the user submits the form and after any action is taken on the record in the database.
Async	After the user submits the form and after the scheduler runs the scheduled job created from the business rule. The system creates a scheduled job from the business rule after the user submits the form but before any action is taken on the record in the database. **Note:** Newly created business rules will run during upgrades. If a record has an asynchronous business rule that makes decisions based on the data in the record, multiple updates to the record in quick succession can cause the business rule to execute out of order or incorrectly. If multiple async business rules update the same record, the updates performed by one script could be overwritten by another script or made in an unexpected sequence because the order of execution isn't guaranteed. You can use the After option for business rules or System Events as an alternative in these situations.
Display	Before the form is presented to the user, just after the data is read from the database.

Note:

- Asynchronous business rules do not have access to the previous version of a record. Therefore, the `changes()`, `changesTo()`, and `changesFrom()` `GlideElement` methods do not work with async rule script. However, the condition builder and condition field (advanced view) both support the `changes()`, `changesTo()`, and `changesFrom()` methods.
- Business rules do not honor ACLs until you want them to be honored. For more information, see Relationship between Business Rules and Access Control Rules (ACLs)

The following options are provided to determine what record operations the business rule applies to.

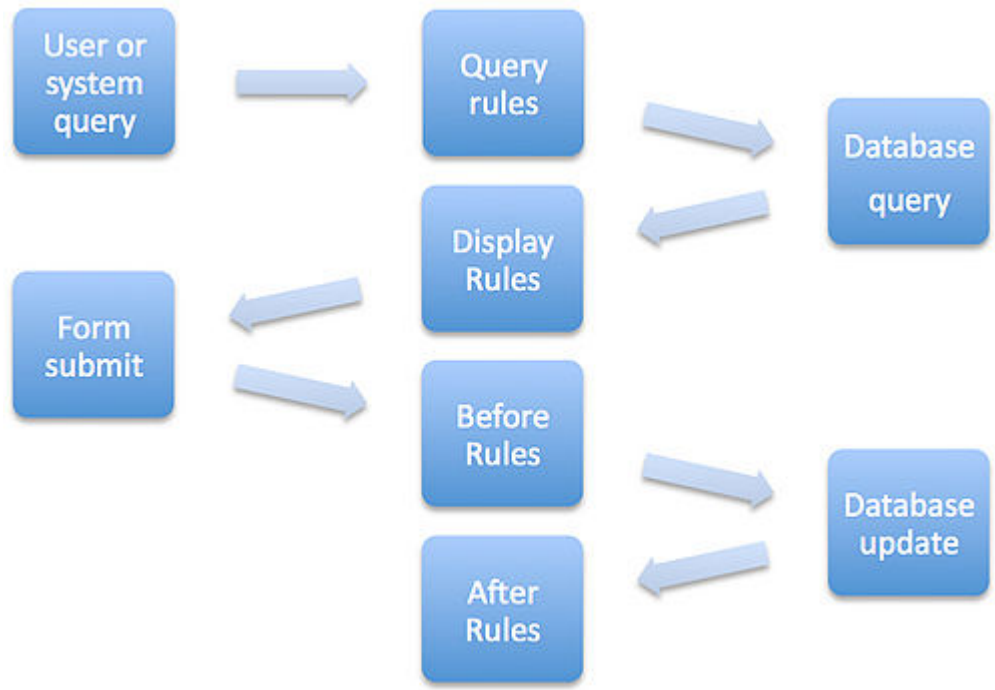
What record operation the business rule applies to

Option	When the rule runs
Insert	When the user creates a new record and the system inserts it into the database.
Update	When the user modifies an existing record.
Query	When the user sends a query for a record or list of records to the database. Typically you should use query operation for before business rules.
Delete	When the user deletes a record.

Note:
Business rules only run record operations when called from the GlideRecord API. Some applications intentionally bypass business rule processing to perform record operations directly. In addition, business rules ignore API calls run with the setWorkflow() method set to false.

This image shows when different types of business rules run:

Business rule processing flow



Note:
Business rules apply consistently to records regardless of whether they are accessed through forms, lists, or web services. This is one major difference between business rules and client scripts, which apply only when the form is edited.

Business rule actions

Business rules can perform a variety of actions. Common types of actions are:

- Changing field values on a form that the user is updating. Field values can be set to specific values available for that field, values copied from other fields, and relative values determined by the user's role.
- Displaying information messages to the user.
- Changing values of child tasks based on changes to parent tasks.
- Preventing users from accessing or modifying certain fields on a form.
- Aborting the current database transaction. For example, if certain conditions are met, prevent the user from saving the record in the database.

Administrators can set field values, create information messages, and abort transactions without writing a script.

Prevent recursive business rules

Avoid using `current.update()` in a business rule script. The `update()` method triggers business rules to run on the same table for insert and update operations, leading to a business rule calling itself over and over. Changes made in before business rules are automatically saved when all before business rules are complete, and after business rules are best used for updating related, not current, objects. When a recursive business rule is detected, the system stops it and logs the error in the system log. However, `current.update()` causes system performance issues and is never necessary.

You can prevent recursive business rules by using the `setWorkflow()` method with the false parameter. The combination of the `update()` and `setWorkflow()` methods is only recommended in special circumstances where the normal before and after guidelines mentioned above don't meet your requirements.

Business rules in scoped applications

Every business rule is assigned to either a private application scope or to the global scope.

The types of business rules you can create and how you can access those rules varies depending on the scope of the business rule and the scope of the table it runs on.

Note:
The term global can refer to two different aspects of a business rule: the table it runs on and the scope it runs in. Business rules can either run on specific tables or be global. In addition, they can be in the global scope or in a private application scope.

Business rules on specific tables

Most business rules run on a specific table, which is defined in the **Table** field. You can create business rules on tables in the same scope and on tables that allow configuration records from another application scope.

For tables that are in a different scope than the business rule record, the types of rules are limited.

- You can create a rule where **When is async** with any of the following options:
  - **Insert**, **Update**, and **Delete** database operations. You cannot select **Query**.
  - **Set field values** actions and scripts (the **Script** field).
- You can create a rule where When is before with any of the following options:
  - **Insert**, **Update**, and **Delete** database operations. You cannot select **Query**.
  - **Set field values** actions only. You cannot write scripts and you cannot abort the database transaction.
- You cannot create any other types of business rules on tables in a different scope.

Business rules on specific tables cannot be accessed by other business rules or scripts.

Global business rules

Warning:
Consider using script includes instead of global business rules. Script includes load only on request while global business rules load on every page in the system.

Global business rules are business rules where the **Table** field is set to **Global**. Global business rules may be accessible on multiple tables and from other scripts, depending on their scope

protection. For a global business rule, define the scope protection by setting the **Accessible from** field:

- **This application scope only**: prevents applications in a different scope than the business rule from calling this business rule.
- **All application scopes**: allows any application to call this business rule.

Note:
Global business rules do not support domain separation.

Scripts in scoped business rules

When you write a script in a business rule, you can access:

- Any script includes and global business rules in the same scope as the business rule.
- Script includes and global business rules that allow applications in a different scope to call them. To call functions from another scope, you must specify the scope of the function.
- For business rules in a unique scope, you can access the scoped system APIs only.

Create a business rule

You can create any type of business rule to run when a record is displayed, inserted, updated, or deleted, or when a table is queried.

About this task

Note:
These instructions and examples provide general guidance for how to implement this functionality. For help with unique use cases, refer to the Developer Community Forum, where you can ask questions, interact with other developers, and search for existing solutions.

Procedure

1. Navigate to **All** > **System Definition** > **Business Rules**.
2. Click **New**.
3. Fill in the fields, as appropriate.

Note:
You might need to configure the form to see all fields.

Business Rule fields

Field	Description
Name	Enter a name for the business rule.
Table	Select the table that the business rule runs on. **Note:** The list shows only tables and database views that meet the scope protections for business rules. Business rules defined for a database view can run only on **Query**. A business rule for a database view cannot run on insert, update, or delete.
Application	Application that contains this business rule.

Field	Description
Accessible from	Scope protection for a global business rule. **Note:** This field is visible only when the **Table** field is set to **Global**. It does not apply to rules that run on specific tables.
Active	Select this check box to enable the business rule.
Advanced	Select this check box to see the advanced version of the form.
When to run	
When	[Advanced] Select when this business rule should execute: **display**, **before**, **async**, or **after** the database operation is complete. **Note:** Consider setting the **Order** for **async** business rules as the system uses this value when creating the associated scheduled job. Newly created **async** business rules run automatically on upgrade. Existing **async** business rules can be migrated to use the new async behavior.
Order	[Advanced] Enter a number indicating the sequence in which this business rule should run. If there are multiple rules on a particular activity, the rules run in the order specified here, from lowest to highest.
Insert	Select this check box to execute the business rule when a record is inserted into the database.
Update	Select this check box to execute the business rule when a record is update.
Delete	[Advanced] Select this check box to execute the business rule when a record is deleted from the database.
Query	[Advanced] Select this check box to execute the business rule when a table is queried.
Filter Conditions	Use the condition builder to determine when the business rule should run based on the field values in the selected Table. If you select the **Advanced** option, you can also use the **Condition** field to build a condition with a script. Conditions defined with the **Filter Conditions** field and advanced **Conditions** field are evaluated at the same time. **Note:** Filters based on string compares are case-sensitive.
Role Conditions	Select the roles that users who are modifying records in the table must have for this business rule to run.
Actions	
Set field values	Set values for fields in the selected Table using the choice lists: ○ The field ○ The assignment operator:

Field	Description
	▪ **To:** An exact value ▪ **Same as:** The value of another field ▪ **To (dynamic):** A value relative to the user configuring the business rule or a user with a specific role ○ The value
Add message	Select this check box and enter a message that appears when this business rule is run
Abort action	Select this check box to abort the current database transaction. For example, on a before insert business rule, if the conditions are met, do not insert the record into the database. If you select this option, you cannot perform additional actions on the record, such as setting field values and running scripts. You can still display a message to users by selecting the **Add message** check box and composing the message.
Advanced	
Condition	Create a JavaScript conditional statement to specify when the business rule should run. By adding the condition statement to this field, you tell the system to evaluate the condition separately and run the business rule only if the condition is true. If you decide to include the condition statement in the **Script** field or the **Filter Conditions** field, leave this field blank. Conditions defined with the **Filter Conditions** field and advanced **Conditions** field are evaluated at the same time. To have the instance reevaluate the condition statement a second time before running an async business rule, add the system property *glide.businessrule.async_condition_check* and set the value to true.
Script	[Advanced] Create a script that runs when the defined condition is true. ○ onAfter ○ onAsync ○ onBefore ○ onDisplay For more information and examples, see Example business rule scripts.
Related list: Versions	
Versions	Shows all versions of the business rule. Use this list to compare versions or to revert to a previous version.

4. Click **Submit**.

Trouble?
If you run into issues with your business rule, see the Business Rule FAQ [KB0965707] article in the Now Support Knowledge Base.

Global variables in business rules

Predefined global variables are available for use in business rules.

Use the following predefined global variables to reference the system in a business rule script.

Global variable	Description
current	Current state of the record being referenced. See "Prevent null pointer exceptions" below to check for nulls before using this variable.
previous	State of the referenced record prior to any updates made during the execution context, where the execution context begins with the first update or delete operation and ends after the script and any referenced business rules are executed. If multiple updates are made to the record within one execution context, *previous* will continue to hold the state of the record before the first update or delete operation. Available on update and delete operations only. Not available on async operations. See "Prevent null pointer exceptions" below to check for nulls before using this variable.
g_scratchpad	Scratchpad object is available on display rules, and is used to pass information to the client to be accessed from client scripts.
gs	References to GlideSystem functions.

The variables *current*, *previous*, and *g_scratchpad* are global across all business rules that run for a transaction.

Prevent null pointer exceptions

In some cases, there may not be a *current* or *previous* state for the record when a business rule runs, which means that the variables will be null. To check for null before using a variable, add the following code to your business rule:

```
if (current == null) // to prevent null pointer exceptions.
return;
```

Define variables

User-defined variables are globally scoped by default. If a new variable is declared in an order 100 business rule, the business rule that runs next at order 200 also has access to the variable. This may introduce unexpected behavior.

To prevent such unexpected behavior, always wrap your code in a function. This protects your variables from conflicting with system variables or global variables in other business rules that are not wrapped in a function. Additionally, variables such as *current* must be available when a function is invoked in order to be used.

The following script is vulnerable to conflicts with other code. If the variable *now_GR* is used in other rules, the value of the variable may unexpectedly change.

```
var now_GR = new GlideRecord('incident');
now_GR.query();
while(now_GR.next()) {

   //do something

}
```

When this script is wrapped in a function, the variable is available only within the function and does not conflict with other functions using a variable named *now_GR*.

```
myFunction();

function myFunction() {
  var now_GR = new GlideRecord('incident');
  now_GR.query();
  while(now_GR.next()) {
    //do something
} }
```

Use business rules and client scripts to control field values

Implement both business rules and client scripts for a field to enable users to set record values properly using both forms and lists, and to see immediate changes to the values in forms as edits are made.

The problem with using only a client script or a business rule to control updates to a field is that fields can be changed on either a form or a list. Client scripts and UI policies run on forms only (client-side) and do not apply to list editing. Allowing list editing with client scripts running on fields in a form can result in incorrect data being saved to the record. For systems in which client scripts or UI policies apply to forms, either disable list editing or create appropriate business rules or access control to control the setting of values in the list editor. A side effect of this is that security measures implemented in client scripts are easy to circumvent. The user only needs to edit the field in a list.

Business rules on a form are not dynamic, the user must update the record for the change to be seen. This makes using client scripts the preferred method for controlling field values on forms.

When using both a business rule and client script to control field values, the update behavior is the same across the system. This means that updated values are not different depending on whether a list of form is used to make the change. This means that the same functionality must be implemented twice, once in a client script and once in a business rule or access control.

Example: Use a business rule to create email addresses during user record import

An organization has a client script that sets the email address for a user to `first.last@company.com`. Administrators do this so they can see the email address immediately when they enter the user's information. The administrator then performs a bulk import of users from a spreadsheet containing the users' first and last names. The expectation is that each user's email address will be set automatically, as they are when they edit the form. Since the client script runs only on the form (the interface to the record), it has no effect on data imported into the record from outside that interface, and no email addresses are created. To solve this problem, the administrator implements a business rule that runs when the import occurs and creates the email addresses.

Example: Prevent list edit for a field that is not editable in the form

An organization wants to hide the **Priority** field on an incident form if the assignment group is **Development**. They create a UI policy on the incident form to do this, but their users can still see and edit the **Priority** field using the list editor. To rectify this, apply an access control to prevent read access to the **Priority** field when the assignment group is **Development**.

Using NULL as a field value

The string NULL has a particular role in scripts and is a reserved word.

The reserved word is NULL in all capital letters. A field with the value **Null** or **null**, for example, is acceptable. Only use NULL to clear out a particular field.

Any NULL field values obtained from an import set data source are inserted into the staging table as empty field values. You should not use the term NULL as a field value in import set transform maps or anywhere in the **First name** or **Last name** fields. Also, do not use NULL in reference fields as the system interprets the value as a string containing the word NULL, not as a reserved word.

Display business-rules

Display rules are processed when a user requests a record form.

The data is read from the database, display rules are executed, and the form is presented to the user. The current object is available and represents the record retrieved from the database. Any field changes are temporary since they are not yet submitted to the database. To the client, the form values appear to be the values from the database; there is no indication that the values were modified from a display rule. This is a similar concept to calculated fields.

The primary objective of display rules is to use a shared scratchpad object, *g_scratchpad*, which is also sent to the client as part of the form. This can be useful when you need to build client scripts that require server data that is not typically part of the record being displayed. In most cases, this would require a client script making a call back to the server. If the data can be determined prior to the form being displayed, it is more efficient to provide the data to the client on the initial load. The form scratchpad object is an empty object by default, and used only to store name:value pairs of data.

To populate the form scratchpad with data from a display rule:

```
// From display business rule
g_scratchpad.someName = "someValue";
g_scratchpad.anotherName = "anotherValue";

// If you want the client to have access to record fields not
 being displayed on the form
g_scratchpad.created_by = current.sys_created_by;

// These are simple examples, in most cases you will probably
 perform some other
// queries to test or get data
```

To access the form scratchpad data from a client script:

```
// From client script
if(g_scratchpad.someName == "someValue") {
  //do something special
}
```

Task Active State Management business rule

This business rule determines whether the active field value needs to change based on changes to the **State** field.

The Task Active State Management business rule is executed when the **State** is changed for a task record. Its execution order is 50, and it runs before most other task business rules.

If the current task table has the *close_states* attribute defined on its table, or if it is inherited from a higher-level table, then the rule determines whether the active field needs to change. This is done by comparing the previous and current state values.

- If the state changes from an active state to an inactive state, the **Active** field is set to false.
- If the state changes from an inactive state to an active state, the **Active** field is set to true, effectively re-activating or re-opening the task.

It is recommended that you leverage the *(current.active.changesTo([true/false])* action in your business rule, as opposed to creating rules on each task table that mark tasks as inactive or active.

Example business rule scripts

Find an example business rule script that helps you with a requirement of your organization.

Note:
These instructions and examples provide general guidance for how to implement this functionality. For help with unique use cases, refer to the Developer Community Forum, where you can ask questions, interact with other developers, and search for existing solutions.

Compare date fields in a business rule

It is possible to compare two date fields or two date and time fields in a business rule, and abort a record insert or update if they are not correct.

For example, you may want a start date to be before an end date. The following is an example script:

```
if ((!current.u_date1.nil()) && (!current.u_date2.nil())) {
  var start =
 current.u_date1.getGlideObject().getNumericValue();
  var end = current.u_date2.getGlideObject().getNumericValue();
  if (start > end) {
    gs.addInfoMessage('start must be before end');
    current.u_date1.setError('start must be before end') ;
    current.setAbortAction(true);
 } }
```

This example has been tested in global scripts, and may need changes to work in scoped scripts. In addition to possibly needing API changes, security is more strict in scoped scripts.

As a good practice, make the business rule a before rule for insert and update actions. In the example script:

- *u_date1* and *u_date2* are the names of the two date fields. Replace these names with your own field names.
- The first line checks that both fields actually have a value.
- The next two lines create variables that have the dates' numerical values.
- The next two lines create different alert messages for the end user: one at the top of the form and one by the *u_date1* field in the form.
- The last line aborts the insert or update if the date fields are not correct.

Here is a more complex example of the above comparison. If you have more than one pair of start and end dates, you can use arrays as shown. Additionally, this script requires the input dates to be within a certain range, in this case, no fewer than 30 days in the past and no more than 365 days in the future.

```
// Enter all start and end date fields you wish to check, as
 well as the previous values
```

```
// Make sure that you keep the placement in the sequence the
 same for all pairs
var startDate = new
 Array(current.start_date,current.work_start);
var prevStartDate = new
 Array(previous.start_date,previous.work_start);
var endDate = new Array(current.end_date,current.work_end);
var prevEndDate = new
 Array(previous.end_date,previous.work_end);

// The text string below is added to the front of ' start must
 be before end'
var userAlert = new Array('Planned','Work');

// Set the number of Previous Days you want to check
var pd = 30;
// Set the number of Future Days you want to check
var fd = 365;

// You shouldn't have to modify anything below this line

var nowdt = new GlideDateTime();
nowdt.setDisplayValue(gs.nowDateTime());
var nowMs = nowdt.getNumericValue();
var pdms = nowMs;

// Subtract the product of previous days to get value in
 milliseconds
pdms -= pd * 24 * 60 * 60 * 1000;
var fdms = nowMs;

// Add the product of future days to get value in miliseconds
fdms += fd * 24 * 60 * 60 * 1000;
var badDate = false;

 // Iterate through all start and end date / time fields
for (x = 0; x < startDate.length; x ++) {
  if ((!startDate[x].nil()) && (!endDate[x].nil())) {
    var start = startDate[x].getGlideObject().getNumericValue();
    var end = endDate[x].getGlideObject().getNumericValue();
    if (start > end) {
      gs.addInfoMessage(userAlert[x] + ' start must be before
 end');
      startDate[x].setError(userAlert[x] + ' start must be
 before end');
      badDate = true; }
    else if ((prevStartDate[x]) != (startDate[x])) {
      if (start < pdms) {
         gs.addInfoMessage(userAlert[x] + ' start must be fewer
 than ' + pd + ' days ago');
         startDate[x].setError(userAlert[x] + ' start must be
 fewer than ' + pd  + ' days ago');
         badDate = true; } }
    else if ((prevEndDate[x]) != (endDate[x])) {
      if (end > fdms) {
         gs.addInfoMessage(userAlert[x] + ' end must be fewer
 than ' + fd + ' days ahead');
```

```
        endDate[x].setError(userAlert[x] + ' end must be fewer
 than ' + fd + ' days ahead');
        badDate  = true ;
} } } }
if (badDate == true ) {
  current. setAbortAction ( true ) ; }
```

Parse XML payloads

Fields in XML format can be parsed with the system's *getXMLText* function.

Fields that get inserted into the database in XML format, such as the payload of an `ecc_event` row, can be parsed with the system's *getXMLText* function. The *getXMLText* function takes a string and an XPATH expression. For example:

```
var name = gs.getXMLText("<name>joe</name>", "//name");
```

returns the string 'joe'.

Assuming that the field "payload" contains XML, the function call might look like:

```
var name = gs.getXMLText(current.payload, "//name");
```

For information on XPATH, visit w3schools.

Abort a database action in a before business-rule

In a before business rule script, you can cancel or abort the current database action using the *setAbortAction()* method.

For example, if the before business rule is executed during an insert action, and you have a condition in the script that calls `current.setAbortAction(true)`, the new record stored in current is not created in the database. The business rule continues to run after calling *setAbortAction()* and all subsequent business rules will execute normally. Calling this method only prevents the database action on current object from occurring.

You can use the *isActionAborted()* method to determine if the current database action (insert, update, delete) is going to be aborted. *isActionAborted()* is initialized for new threads and the *next()* method explicitly sets its value to false.

Note:
setAbortAction() can only be executed from the same scope as the record whose action is being aborted. `current.setAbortAction` is not honored if executed in a business rule that is defined in a different scope.

Determine the operation that triggered the business rule

You can write a script for a business rule that is triggered on more than one database action.

If you want the business rule script to dynamically branch depending on the action that triggered the event, you can use the *operation()* function. For example:

```
if(current.operation() == "update") {
  current.updates ++; }
  if(current.operation() == "insert") {
    current.updates = 0; }
```

Use an OR condition in a business rule

An **OR** condition can be added to any query part within a business rule.

An **OR** condition can be added to any query part within a business rule with the *addOrCondition()* method. The example below shows a query for finding all the incidents that have either a 1 or a 2 priority. The first *addQuery()* condition is defined as a variable and is used in the **OR** condition.

```
var inc = new GlideRecord('incident');
var qc = inc.addQuery('priority','1');
qc.addOrCondition('priority','2');
inc.query();
while(inc.next()) {
  // processing for the incident goes here
}
```

The following script is a more complex example, using two query condition variables doing the equivalent of `(priority = 1 OR priority = 2) AND (impact = 2 OR impact = 3)`. The results of the **OR** condition are run with two variables, *qc1* and *qc2*. This allows you to manipulate the query condition object later in the script, such as inside an `IF` condition or `WHILE` loop.

```
var inc = new GlideRecord('incident');
var qc1 = inc.addQuery('priority','1');
qc1.addOrCondition('priority','2');
var qc2 = inc.addQuery('impact','2');
qc2.addOrCondition('impact','3');
inc.query();
while(inc.next()) {
  // processing for the incident goes here
}
```

Reference a Glide list from a business rule

A field defined as a glide list is an array of values stored in a single field.

Here are some examples of how to process a glide_list field when writing business rules. Generally a glide_list field contains a list of reference values to other tables.

Examples

For example, the **Watch list** field within tasks is a glide_list containing references to user records.

The code below shows how to reference the field.

```
// list will contain a series of reference (sys_id) values
 separated by a comma
// array will be a javascript array of reference values
var list = current.watch_list.toString();
var array = list.split(",");
for (var i=0; i < array.length; i++) {
   gs.print("Reference value is: " + array[i]);
}
```

Output:

```
*** Script: Reference value is: 62826bf03710200044e0bfc8bcbe5df1
*** Script: Reference value is: c2826bf03710200044e0bfc8bcbe5d45
*** Script: Reference value is: 5f74e421c0a8010e01ec0d74a7ee2cc6
*** Script: Reference value is: 06826bf03710200044e0bfc8bcbe5d57
```

You can also get the display values associated with the reference values by using the *getDisplayValue()* method as shown below.

```
// list will contain a series of display values separated by a
 comma
// array will be a javascript array of display values
var list = current.watch_list.getDisplayValue();
var array = list.split(",");
for (var i=0; i < array.length; i++) {
   gs.print("Display value is: " + array[i]);
}
```

Output:

```
*** Script: Display value is: Abel Tuter
*** Script: Display value is:  Ashley Leonesio
*** Script: Display value is:  Charles Beckley
*** Script: Display value is:  Cherie Fuhri
```

Use indexOf("searchString") to find a string in a Glide list

Use *indexOf*("*searchString*") to return the location of the string passed into the method if the glide list field, such as a Watch list, has at least one value in it.

If the field is empty, it returns `undefined`. To avoid returning an undefined value, do any of the following:

- Force the field to a string, such as: *watch_list.toString().indexOf*("*searchString*")
- Check for an empty Glide list field with a condition before using *indexOf()*, such as: if (*watch_list.nil()* || *watch_list.indexOf*("*searchString*") == -1)

Lock user accounts

You can lock user accounts if the user is not active.

The following business rule script locks user accounts if the user is not active in the LDAP directory or the user does not have self-service, itil, or admin access to the instance.

```
// Lock accounts if bcNetIDStatus != active in LDAP and user
 does not
// have self-service, itil or admin role
var rls = current.accumulated_roles.toString();
if(current.u_bcnetidstatus == 'active' && (rls.indexOf(',itil,')
 > 0 ||
  rls.indexOf(',admin,') > 0 ||
  rls.indexOf(',ess,') > 0 )) {
  current.locked_out = false; }
else {
  current.locked_out = true; }

var now_GR = new GlideRecord("sys_user");
now_GR.query();
while(now_GR.next()) {
  now_GR.update();
  gs.info("updating " + gr.getDisplayValue());
}
```

Default before-query business rule

You can use a query business rule that executes before a database query is made.

Use this query business rule to prevent users from accessing certain records. Consider the following example from a default business rule that limits access to incident records.

- Name: incident query
- Table: Incident
- When: before, query
- Script:

```
if(!gs.hasRole("itil") && gs.isInteractive()) {
  var u = gs.getUserID();
  var qc =
 current.addQuery("caller_id",u).addOrCondition("opened_by",u).a
ddOrCondition("watch_list","CONTAINS",u);
  gs.print("query restricted to user: " + u); }
```

This example prevents users from accessing incident records unless they have the itil role, or are listed in the **Caller** or **Opened by** field. So, for example, when self-service users open a list of incidents, they can only see the incidents they submitted.

Note:
You can also use access controls to restrict the records that users can see.

Script includes

Script includes are used to store JavaScript that runs on the server.

Create script includes to store JavaScript functions and classes for use by server scripts. Each script include defines either an object class or a function.

Consider using script includes instead of global business rules because script includes are only loaded on request. See Privacy settings on Glide AJAX enabled script includes and Discovery script includes for more information.

For additional examples of scripts, see Useful scripts.

Script include form

Script includes have a name, description, and, script. They also specify whether they are active or not, and whether they can be called from a client script. View existing or create a new script include using the Script Include form.

To access script includes, navigate to **All** > **System Definitions** > **Script Includes**.

Script include form

Field	Description
Name	The name of the script include. If you are defining a class, this must match the name of the class, prototype, and type. If you are using a classless (on-demand) script include, the name must match the function name.
API Name	The internal name of the Script Include. Used to call the Script Include from out-of-scope applications.
Glide AJAX enabled	The script include is available to client scripts, list/report filters, reference qualifiers, or if specified as part of the URL. Glide AJAX enabled script includes

Script include form (continued)

Field	Description
(or Client callable)	are invoked from `GlideAjax` and require that users satisfy an ACL associated with the script include. When selected, the **Access Controls** Related Link is available. See Privacy settings on Glide AJAX enabled script includes for more information.
Mobile callable	The script include is available to client scripts called from mobile devices.
Sandbox enabled	The script include is available to scripts invoked from the script sandbox, such as a query condition. **Important:** Script includes should only be made available to the script sandbox if necessary. For more information about the sandbox, see Configuring Script sandbox property.
Application	The application where this script include resides.
Accessible from	Sets which applications can access this script include: **All application scopes** Can be accessed from any application scope. **This application scope only** Can be accessed only from the current application scope.
Active	Enables the script include when selected. Deselect the active field to disable the script include.
Description	Provides descriptive content regarding the script include.
Script	Defines the server side script to run when called from other scripts. The script must define a single JavaScript class or a global function. The class or function name must match the **Name** field.
Package	The package that contains this script include.
Created by	The user who created this script include.
Updated by	The user who most recently updated this script include.
Protection policy	Sets the level of protection for the script include: **None** Allows anyone to read and edit this downloaded or installed script include. **Read-only** Allows anyone to read values from this downloaded or installed script include. No one can change script values on the instance on which they download or install the script include. **Protected**

Script include form (continued)

Field	Description
	Provides intellectual property protection for application developers. Customers who download the script include cannot see the contents of the script field. The script is encrypted in memory to prevent unauthorized users from seeing it in plain text.
Related lists on the form view:	
Versions	Shows all versions of the script include. Use this list to compare versions or to revert to a previous version. See Versions.
Access Controls	Becomes available when the **Glide AJAX enabled** is selected and is hidden from standard script includes. Use to protect a script include against unauthorized use when public access is not granted.

Use script includes

Script includes are found under System Definition or System UI. You can call existing script includes from a script or create a new script include.

To create an entirely new script include, you can follow the format of any of the existing script includes. In this example, the name of your script include is `NewInclude` and there is a single function called `myFunction`. It is important that the name of the script include match the name of the class, prototype, and type. When you create a new script include and give it a name, the system provides a code snippet with the class and prototype properly set up.

```
var NewInclude =Class.create();

NewInclude.prototype={
  initialize :function(){},

  myFunction :function(){<Put function code here>},

  type :'NewInclude'};
```

You could then use the `myFunction` line like this:

```
var foo =new NewInclude();
foo.myFunction();
```

Glide AJAX enabled script includes

Glide AJAX enabled Script Includes make the script include available to client scripts, list/report filters, reference qualifiers, or if specified as part of the URL.

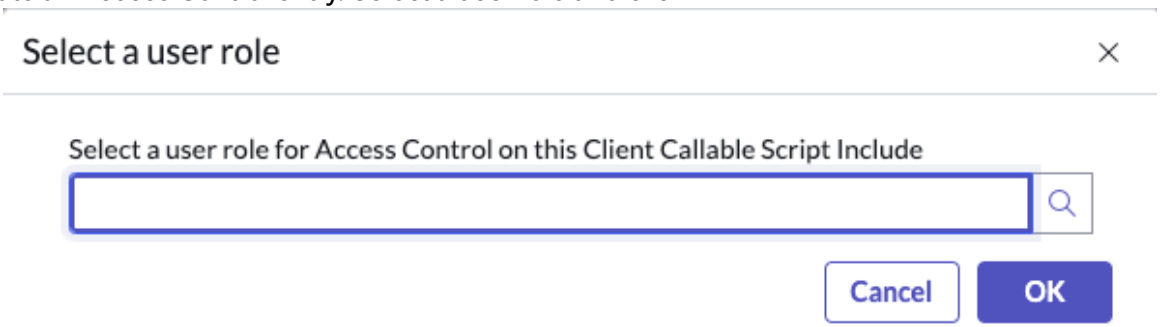
Before you begin

Role required: admin

Procedure

1. Navigate to **System Definition** > **Script Includes**.
2. Select **New** or select an existing script include for viewing or editing.
   See Use script includes for additional information on writing script includes.
3. Complete the form and select the **Glide AJAX enabled** option.

A role selector pops up to select a user role and automatically create an Access Control entry. Select a user role and click



OK.

Note:
To disable the role selector window, set the `glide.script.ccsi.enable_acl_create_ux` to **false**.

A script include record with a role-based Access Control is created. The Access Control Related Link becomes available with the selection of the **Glide AJAX enabled** check box.

Privacy settings on Glide AJAX enabled script includes

Privacy settings for Glide AJAX enabled script includes determine who can access a Glide AJAX enabled script include.

Private privacy-setting

The private privacy-setting means that guests who access public pages cannot access the Glide AJAX enabled script-include. A private script cannot be executed by a non-logged-in user.

Public privacy-setting

A public privacy-setting means that the client script can be executed by non-logged-in users that create an appropriate HTTP request. This can create a security problem if the client script provides confidential information.

The following script includes remain public by default because Make UI pages public or private need to access them:

- GlideSystemAjax
- SysMessageAjax
- KnowledgeMessagingAjax
- KnowledgeAjax
- PasswordResetAjax

Set privacy on all Glide AJAX enabled script includes

Change the privacy setting on all Glide AJAX enabled script includes.

To provide further control over all Glide AJAX enabled script includes, administrators can add the `glide.script.ccsi.ispublic` property. This property changes the visibility of Glide AJAX enabled script includes by making them all public or private. Configure the property as follows:

Configure property

Title	Property
Name	glide.script.ccsi.ispublic
Type	true|false
Value	false

Note:
To learn more about this property, see Require authentication by default for client-callable script includes [Updated in Security Center 1.3] in Instance Security Hardening Settings.

Change privacy on a single Glide AJAX enabled script include

Change the privacy setting for a single Glide AJAX enabled script include by adding the `isPublic()` function.

The `isPublic()` setting takes precedence over the `glide.script.ccsi.ispublic` property. For example, if the property is set to **false**, making all Glide AJAX enabled script-includes private, and a script sets `isPublic()` to **true**, the script is public.

To change the privacy for a single Glide AJAX enabled script include, add the following method to the script include:

```
isPublic:function(){return[true/false];},
```

Make the NewInclude client script private.

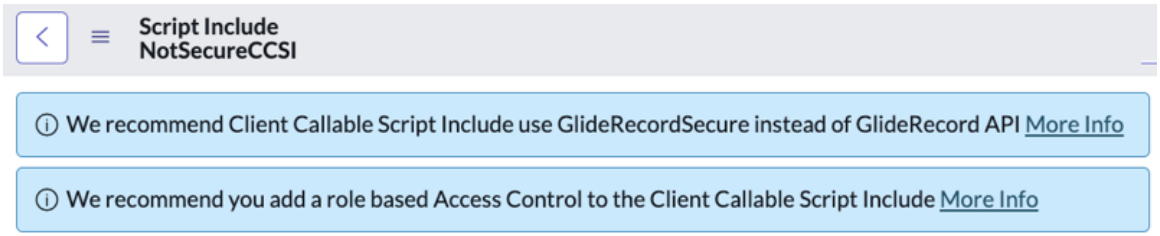
```
var NewInclude =Class.create();

NewInclude.prototype={
   initialize:function(){},

   myFunction:function(){//Put function code here},
   isPublic:function(){return false;},

   type:'NewInclude'};
```

Security on Glide AJAX enabled script includes

Protect your Glide AJAX enabled script includes against unauthorized use. For all records which are created a customer application, recommendations display that may help reduce security risk.

When creating a Glide AJAX enabled script include, the system displays the following security recommendations if they have not yet been configured:

- Add or define an Access Control, unless the script include has public access.
- Use `GlideRecordSecure` instead of `GlideRecord` API for better security, if the script queries the database.



Note:
To disable the security recommendation messages, set the property `glide.script.ccsi.customer_scoped.security_msgs_enabled` to **false** in the sys_properties table. The default value is set to **true**.

See Instance Security Hardening Settings for additional information on security compliance.

Discovery script includes

Discovery script includes define JavaScript classes that you can use to accomplish Discovery tasks.

Note:
Users with discovery_admin role can write script includes. Follow best practices for server-side and client-side scripting to prevent security issues. See knowledge article KB0550828 for more information.

Using GlideRecordUtil to Work with GlideRecords

`GlideRecordUtil` is a utility class that provides methods that are useful for working with GlideRecords during Discovery. Refer to GlideRecordUtil for descriptions of available methods.

Getting a GlideRecord Instance

To get a `GlideRecord` instance for a given configuration item, and of the correct class and table, use the `getCIGR(sys_id)` method. For example, the following code gets the GlideRecord of a CI with the sys_id of 2dfd7c8437201000deeabfc8bcbe5d56:

```
var now_GR = new
 GlideRecordUtil().getCIGR("2dfd7c8437201000deeabfc8bcbe5d56");
```

To retrieve any hierarchical table without knowing its class type, use the `getGR(base_table, sys_id)` method. For instance, to get a GlideRecord for a computer class CI, you might have to distinguish whether it is a computer class, Windows server, or Linux server class. Using this method guarantees a GlideRecord with the correct class. Different classes have different attributes. In this use case, a Windows server has attributes different from a Linux server. The following example shows how to get a GlideRecord in the correct class with its attributes.

```
var now_GR = new
 GlideRecordUtil().getGR( "cmdb_ci_computer", "2dfd7c8437201000deeabfc8b
cbe5d56");
```

Getting All the Fields In a GlideRecord

The `getFields(now_GR)` method returns a JavaScript object, such as a hashmap, of all the fields or attributes that exist in a given GlideRecord.

```
var now_GR = new
 GlideRecordUtil().getGR("cmdb_ci_computer", "2dfd7c8437201000deeabfc8bc
be5d56");
var fields = new GlideRecordUtil().getFields(now_GR);
gs.log(fields.join(" ")); // List all the fields that are in a computer CI
```

Populating GlideRecord Object Fields

The *populateFromGR(hashmap, gr, ignore)* method enables you to take a GlideRecord object and populate its fields and values into a JavaScript object. The third argument (*ignore*) is an optional JavaScript object that enables you to exclude certain fields. For example, you may not care about *sys_created_by* or *sys_updated_by* fields in a GlideRecord.

```
var objectToPopulate = { };
var now_GR = new
 GlideRecordUtil().getGR("cmdb_ci_computer", "2dfd7c8437201000deeabfc8bc
be5d56");
var ignore = {"sys_created_on": true, "sys_updated_by": true};
new GlideRecordUtil().populateFromGR(objectToPopulate, now_GR,
 ignore);
// Now the objectToPopulate contains field/value pairs from the computer GlideRecord
```

The *mergeToGR(hashmap, gr, ignore)* method enables you to populate a GlideRecord with a field/value-paired object. The ignore argument stops specified fields from being updated. The following code example updates a computer record's *name* and *os* fields, but does not update the *sys_created_by* field:

```
var now_GR = new
 GlideRecordUtil().getGR("cmdb_ci_computer", "2dfd7c8437201000deeabfc8bc
be5d56");
var obj = {"name": "xyz", "os": "windows 2000", "sys_created_by", "aleck.lin"};
var ignore = {"sys_created_by": true};
new GlideRecordUtil().mergeToGR(obj, gr, ignore);
gr.update();
```

Getting Table Hierarchies

The *getTables(table)* method returns a list of table hierarchies, as shown in the following example:

```
var tables = new GlideRecordUtil().getTables("cmdb_ci_linux_server");
gs.log(tables.join(","));
// The result would be "cmdb_ci, cmdb_ci_computer, cmdb_ci_server, cmdb_ci_linux_server".
```

Using DiscoveryException and AutomationException

When writing Discovery sensors and sensor-related scripts, you may want to use DiscoveryException or AutomationException to indicate that an exception has come from Discovery.

The *DiscoveryException* script include extends *AutomationException*, which extends the *GenericException* class. The following example uses *DiscoveryException* to throw an exception:

```
function foo() {
   if(//condition matches) throw new DiscoveryException("The message", "The cause"); }
```

The first argument takes the message of the exception and the second argument (optional) takes the cause of the exception. You may also want to catch the exception and log it as shown in the example below:

```
try {
  foo();
}
catch(e) {
   if(e instanceof DiscoveryException)
     gs.log("A DiscoveryException occurred. It is " + e. getMessage() + "
caused by " + e.getCause()); }
```

The above example also applies for *AutomationException*. *DiscoveryException* is typically used to provide exception processing specifically for Discovery, while *AutomationException* is used for exception processing that applies to both Orchestration and Discovery.

Processors

Processors provide a customizable URL endpoint that can execute arbitrary server-side JavaScript code and produce output such as TEXT or JSON. Creating custom processors is deprecated.

Note:
This feature is deprecated. While legacy, existing custom processors continue to be supported, creating new custom processors has been deprecated. Instead, use the Scripted REST APIs.

Warning:
When creating a processor, ensure that you use parameter names that are specific to your processor. For example, if your processor exports a list of legal records, and a necessary parameter is the recipient's email address, don't use “email” as the parameter name. Create a more processor specific parameter name, such as legal_export_recipient_email. Failure to do so, and using instance parameter names, such as id, table, sys_id, service, catalog_id, or view (and others), can cause unexpected results.

When to create processors

Do not create custom processors. This feature is deprecated. Please use the REST APIs instead of creating custom processors. The remaining information is left for existing processors only.

Processor form

Field	Description
Name	Unique name of the processor.
Type	Programming language of the processor script. Options include: • java: do not select this option • script
Application	Application containing this record.
Active	Flag to enable or disable the record.

Field	Description
CSRF protect	Option to protect the processor from running unless the instance uses a CSRF token.
Description	Description of the processor's function or purpose.
Parameters	List of available input parameters. Specify parameter values in the URL as *<parameter name>=<parameter value>*. **Note:** Parameter names must be processor-specific. Do not choose common parameter names that another processor might use. If you use a common parameter name, such as `id`, `sys_id` or `table` in a processor, it can break other functionality, since the processor wins when that parameter exists in a URL. For example, a processor with an `id` parameter, regardless of the Path value in the same record, breaks the Service Portal, which depends on that parameter for page identification.
Path	URI path used to call this processor. Call a processor from the URL as: `https://<instance name>.service-now.com/<Path>.do`
Script	Immediately Invoked Function Expression to run when the system calls this processor. The function automatically provides input parameters for the following API objects. • g_request • g_response • g_processor
Protection policy	Policy to use to protect this record's script. Options include: • None • Read-only • Protected

Processor API components

Processors have access to dedicated API classes, objects, and methods.

Processor API components

Class, object, or method	Description
`g_response`	An object of type HttpServletResponse. See GlideServletResponse.
`setContentType('text/html;charset=UTF-8')`	A GlideServletResponse method to set the content type of the response being sent to the client.
`g_request`	An object of type HttpServletRequest. See HttpServletRequest.
`getParameter()`	A glide method to get the value of a URL parameter.
`canRead()`	A GlideRecord method to determine if the user can read data from a table. See GlideRecord.
`g_processor`	A simplified servlet for processors. See GlideScriptedProcessor.
`writeOutput()`	A GlideScriptedProcessor method to display information on the client.
`g_target`	An object containing the target table name of a processor URL. For example, a processor containing the URI `incident.do` applies to the Incident table.

Secure and protect a processor

You can protect your processor against unauthorized use by using role restrictions, and protect it by requiring a CSRF token.

About this task

Note:
This feature is deprecated. While legacy, existing custom processors continue to be supported, creating new custom processors has been deprecated. Instead, use the Scripted REST APIs.

You can re-use a table's user role restrictions to protect it from access by your processor. This protection method assumes the processor will access table data.

Procedure

1. Create or select a user role that has access to the table the processor script calls.
2. Navigate to **System Definition** > **Processors**.
3. In **Script**, add the following code block.

```
var now_GR = new GlideRecord('your_table_name');
// canRead() compares the table's ACL to the user making this
 request, and returns true if the logged-in user has read
 access to this table
if(gr.canRead())
{
  // Perform table query here
  g_processor.writeOutput('Success!');
} else {
```

```
  g_processor.writeOutput('You do not have permission to read
 table your_table_name');
}
```

4. Update the code block to use other access restrictions as needed.

   Available access functions include:

   - `canCreate()`
   - `canRead()`
   - `canWrite()`
   - `canDelete()`

5. Click **Update**.

Protect a processor with a CSRF token

You can protect a processor by requiring a CSRF token.

About this task

Script type processors can require a CSRF token check before the processor runs.

Procedure

1. Navigate to **All** > **System Definition** > **Processors**.
2. Open a processor record.
3. Select the **CSRF protect** option.
4. Click **Update**.

Create a simple processor

Create a simple processor to execute a script from a URL query. This feature is deprecated.

Before you begin

You must have your own demonstration instance.

Role required: admin

About this task

Note:
This feature is deprecated. While legacy, existing custom processors continue to be supported, creating new custom processors has been deprecated. Instead, use the Scripted REST APIs.

Procedure

1. Navigate to **All** > **System Definition** > **Processors**.
2. Select **New**.
3. Enter the following information.

4. Select **Submit**.

5. Enter a URL query to the instance with the following format: `https://instance.service-now.com/processor_name.do?parameter=value`
   For example: `https://<instancename>.service-now.com/Hello.do?name=world`.

Create a multi-table processor

Create a multi-table processor that reports the number of rows in any table on your instance. This feature is deprecated.

Before you begin
Role required: admin

About this task

Note:
This feature is deprecated. While legacy, existing custom processors continue to be supported, creating new custom processors has been deprecated. Instead, use the Scripted REST APIs

The multi-table processor protects itself from performance and security violations by confirming that the user is authorized to read the table. It does not report on certain tables that are too large to query safely.

Procedure

1. Navigate to **All** > **System Definition** > **Processors**.
   The list of processors appears.
2. Select**New**.
3. Enter the following information.

Name	TableSize
Type	Choose **Javascript**
Description	Return number of records in a table
Parameters	**SIZE**
Path	<leave empty>

Script:

```
g_response.setContentType('text/html;charset=UTF-8');
if(g_target === 'sys_email' || g_target === 'sys_log' )
{
  g_processor.writeOutput(g_target + ' table is too large to
 quickly count');
  } else {
    var count = new GlideAggregate(g_target);
    if( count.canRead() ) {
      count.addAggregate('COUNT');
      count.query();
      var records = 0;
      if (count.next()) {
        records = count.getAggregate('COUNT');
       }
```

```
      g_processor.writeOutput('table ' + g_target + ' has ' +
 records + ' records');
     } else {
      g_processor.writeOutput('You do not have access to table
 ' + g_target);
  }
}
```

4. Select **Save**.
5. Test the new processor by entering the following URLs:
   `https://<instancename>.service-now.com/incident.do?SIZE` and
   `https://<instancename>.service-now.com/sys_user.do?SIZE`
   Your instance reports the number of records in the table. For example, `table incident has 82 records`.

Create a custom processor

Create a custom processor to execute a script from a URL query. This feature is deprecated.

Before you begin

Role required: admin

About this task

Note:
This feature is deprecated. While legacy, existing custom processors continue to be supported, creating new custom processors has been deprecated. Instead, use the Scripted REST APIs.

When complete, you can do the following tasks:

- Create a new processor
- Run a script from a URL query

The following example steps assume you have your own demonstration instance.

Procedure

1. Navigate to **All** > **System Definition** > **Processors**.
2. Select **New**.
3. In the **Name** field, enter `Hello`.
4. In the **Type** field, select *script*.
5. In the **Path** field, enter `Hello`.
6. In the **Script** field, enter the following code.

```
var name = g_request.getParameter("name");
g_processor.writeOutput("text/plain", "Hello " + name);
```

7. Select **Submit**.
8. Log out of the instance and open a new browser window.
9. Enter a URL query to the instance with the following format: `https://instance.service-now.com/processor_name.do?parameter=value`.

For example: `https://<instance name>.service-now.com/Hello.do?name=world`.

10. When prompted, enter the credentials for a valid user.

Scripts - Background module

Administrators can use the Scripts - Background module to run arbitrary JavaScript code from the server.

The Scripts - Background module consists of the following components.

- Text box to enter JavaScript.
- Selector to specify the application scope.
- **Run script** button.
- List of available scopes.
- **Record for rollback?** check box. Selected by default. Creates a rollback context for the script execution. After the script is run, click the **Script execution and recovery available here** link to go to the **Script Execution History** form where you can rollback the script.
- **Execute in sandbox?** check box. Allows script to run with sandbox like restrictions. If checked, data cannot be inserted, updated, or deleted.
- **Execute as scriptlet?** check box. Allows the script to execute as a scriplet, which gives the script access to the global scope and any server-side functionality or objects. If the check box is not selected, the script executes in global scope, but does not have access to any server-side functionality or objects.
- **Cancel after 4 hours** check box. Selected by default. Cancels the script execution if it continues running after four hours.

When a script is run, the instance displays results, information, and error messages at the top of the screen.

Important:
Running free-form JavaScript can cause system disruption or data loss. Do not run free-form scripts from a production instance.

To stop a transaction, see View and kill active transaction.

For examples of scripts you could run, see Useful scripts.

Installation settings

Installation settings are global business rules with calculated names. Installation settings are calculated just before a record is displayed and facilitate dynamic determination of access and roles. Installation Settings permit the programmatic determination of a setting.

Installation settings controlling access to fields and records are:

- CanRead()
- CanWrite()
- CanCreate()
- CanDelete()

Functions can return true if access is permitted, false if not. No return value uses the permission calculated using roles. The function has access to the current record through the variable current code.

The name of the function checking the permission on a record is formed by prefixing the setting name with the record name:

```
record_nameCanRead()
```

Similarly, the permission on a field in a record is formed by prefixing the function name with the record name, underscore, and field name:

```
record_name_field_nameCanRead()
```

Naming examples:

```
function incidentCanWrite() {} //  can user write to this
 record?
 function incident_numberCanWrite() {}  // can user write to the
 number field?
```

This sample business rule restricts the writing of the name field in the sys_dictionary file when the entry exists:

```
  // the element name cannot be written unless this is a new
 record (not yet in database)
  function sys_dictionary_nameCanWrite() {
    if (current.isNewRecord())
      return;

    return false;
  }
```

Using DurationCalculator to calculate a due date

Using the DurationCalculator script include, you can calculate a due date, using either a simple duration or a relative duration base on schedules.

The following script demonstrates how to use the global API DurationCalculator to calculate a due date. The first part of the script illustrates how to set a start datetime using the setStartDateTime() method and then use the calcDuration() method to determine a due date that is "x" amount of continuous time (seconds) from the specified start datetime. The second half of the script illustrates how to use *`DurationCalculator`* to calculate a due date based on a schedule. Schedules enable you to apply a "filter" on future time, such as only including the days in a work week within the calculation. For example, if you apply a schedule "weekdays" (which only includes Monday through Friday) to your duration calculation, and the start datetime is Friday at 5:00 pm, when you add a duration of two days, your due date would be Tuesday at 5:00 pm. If you did not use a schedule, your due date would be Sunday at 5:00 pm. For additional information on schedules, see Creating and using schedules.

This script can be cut and pasted into the Scripts Background page and run as is. It can also serve as an example for authoring business rules, UI actions, or used any other place that server-side script can be authored.

```
/**
 * Demonstrate the use of DurationCalculator to compute a due
 date.
 *
 * You must have a start date and a duration. Then you can
 compute a
 * due date using the constraints of a schedule.
 */

gs.include('DurationCalculator');
executeSample();
```

```
/**
 * Function to house the sample script.
 */
function executeSample(){

    // First we need a DurationCalculator object.
    var dc = new DurationCalculator();

    // --------------- No schedule examples -----------------

    // Simple computation of a due date without using a
schedule. Seconds
    // are added to the start date continuously to get to a due
date.
    var gdt = new GlideDateTime("2012-05-01 00:00:00");
    dc.setStartDateTime(gdt);
    if(!dc.calcDuration(2*24*3600)){ // 2 days
      gs.log("*** Error calculating duration");
      return;
    }
    gs.log("calcDuration no schedule: " +
dc.getEndDateTime()); // "2012-05-03 00:00:00" two days later

    // Start in the middle of the night (2:00 am) and compute a
due date 1 hour in the future
    // Without a schedule this yields 3:00 am.
    var gdt = new GlideDateTime("2012-05-03 02:00:00");
    dc.setStartDateTime(gdt);
    if(!dc.calcDuration(3600)){
      gs.log("*** Error calculating duration");
      return;
     }
    gs.log("Middle of night + 1 hour (no schedule): "+
dc.getEndDateTime()); // No scheduled start date, just add 1
hour


    // -------------- Add a schedule to the date calculator
---------------------
    addSchedule(dc);

    // Start in the middle of the night and compute a due date 1
hour in the future.
    // Since we start at 2:00 am the computation adds the 1 hour
from the start
    // of the day, 8:00am to get to 9:00am
    var gdt = new GlideDateTime("2012-05-03 02:00:00");
    dc.setStartDateTime(gdt);
    if(!dc.calcDuration(3600)){
      gs.log("*** Error calculating duration");
      return;
    }
    gs.log("Middle of night + 1 hour (with 8-5 schedule): " +
dc.getEndDateTime()); // 9:00 am
```

```
    // Start in the afternoon and add hours beyond quiting time.
 Our schedule says the work day
    // ends at 5:00pm, if the duration extends beyond that, we
 roll over to the next work day.
    // In this example we are adding 4 hours to 3:00pm which
 gives us 10:00 am the next day.
    var gdt = new GlideDateTime("2012-05-03 15:00:00");
    dc.setStartDateTime(gdt);

    if(!dc.calcDuration(4*3600)){
      gs.log("*** Error calculating duration");
      return;}
    gs.log("Afternoon + 4 hour (with 8-5 schedule): " +
 dc.getEndDateTime()); // 10:00 am.

    // This is a demo of adding 2 hours repeatedly and examine
 the result. This
    // is a good way to visualize the result of a due date
 calculation.
    var gdt = new GlideDateTime("2012-05-03 15:00:00");
    dc.setStartDateTime(gdt);
    for(var i=2; i<24; i+=1){
      if(!dc.calcDuration(i*3600)){
        gs.log("*** Error calculating duration");
        return;
      }
      gs.log("add "+ i +" hours gives due date: " +
 dc.getEndDateTime());
    }

    // Setting the timezone causes the schedule to be
 interpreted in the specified timezone.
    // Run the same code as above with different timezone. Note
 that the 8 to 5 workday is
    // offset by the two hours as specified in our timezone.
    dc.setTimeZone("GMT-2");
    var gdt = new GlideDateTime("2012-05-03 15:00:00");
    dc.setStartDateTime(gdt);
    for(var i=2; i<24; i+=1){
      if(!dc.calcDuration(i*3600)){
        gs.log("*** Error calculating duration");
        return;
      }
        gs.log("add "+ i +" hours gives due date (GMT-2): " +
 dc.getEndDateTime());
    }
}

/**
 * Add a specific schedule to the DurationCalculator object.
 *
 * @param durationCalculator An instance of DurationCalculator
 */
function addSchedule(durationCalculator){
    // Load the "8-5 weekdays excluding holidays" schedule into
 our duration calculator.
    var scheduleName = "8-5 weekdays excluding holidays";
```

```
    var grSched = new GlideRecord('cmn_schedule');
    grSched.addQuery('name', scheduleName);
    grSched.query();
    if(!grSched.next()){
      gs.log('*** Could not find schedule "'+ scheduleName
 +'"');
      return;
    }

 durationCalculator.setSchedule(grSched.getUniqueValue(),"GMT");
}
```

Using DurationCalculator to compute a simple duration

A simple duration is the number of seconds between two date times.

If no schedule is used then this is a simple time date subtraction. If a schedule is used, then the schedule is consulted to remove non-work hours from the computation. Suppose schedule "8-5 weekdays excluding holidays" is used. In this case, the number of work hours from noon Monday to noon Tuesday is nine hours. To compute a simple duration, initialize the global API *`DurationCalculator`* and call the calcScheduleDuration() method.

This script demonstrates how to use *`DurationCalculator`* to compute a simple duration.

```
/**
 * Sample script demonstrating use of DurationCalculator to
 compute simple durations
 *
 */

gs.include('DurationCalculator');
executeSample();

/**
 * Function to house the sample script.
 */
function executeSample(){

    // First we need a DurationCalculator object.
    var dc = new DurationCalculator();

    // Compute a simple duration without any schedule. The
 arguments
    // can also be of type GlideDateTime, such as fields from a
 GlideRecord.
    var dur = dc.calcScheduleDuration("2012-05-01",
 "2012-05-02");
    gs.log("calcScheduleDuration no schedule: " + dur); // 86400
 seconds (24 hours)

    // The above sample is useful in limited cases. We almost
 always want to
    // use some schedule in a duration computation, let's load a
 schedule.
    addSchedule(dc);

    // Compute a duration using the schedule. The schedule
```

```
    // specifies a nine hour work day. The output of this is
 32400 seconds, or
    // a nine hour span.
    dur = dc.calcScheduleDuration("2012-05-23
 12:00:00","2012-05-24 12:00:00");
    gs.log("calcScheduleDuration with schedule: " + dur); //
 32400 seconds (9 hours)

    // Compute a duration that spans a weekend and holiday. Even
 though this
    // spans three days, it only spans 9 work hours based on the
 schedule.
    dur = dc.calcScheduleDuration("2012-05-25 12:00:00",
 "2012-05-29 12:00:00");
    gs.log("calcScheduleDuration with schedule spaning holiday:
 " + dur); // 32400 seconds (9 hours)

    // Use the current date time in a calculation. The output of
 this is
    // dependent on when you run it.
    var now = new Date();
    dur = dc.calcScheduleDuration("2012-05-15", new
 GlideDateTime());
    gs.log("calcScheduleDuration with schedule to now: " +
 dur); // Different on every run.
}

/**
 * Add a specific schedule to the DurationCalculator object.
 *
 * @param durationCalculator An instance of DurationCalculator
 */
function addSchedule(durationCalculator){
    // Load the "8-5 weekdays excluding holidays" schedule into
 our duration calculator.
    var scheduleName = "8-5 weekdays excluding holidays";
    var grSched = new GlideRecord('cmn_schedule');
    grSched.addQuery('name', scheduleName);
    grSched.query();
    if(!grSched.next()){
        gs.log('*** Could not find schedule "'+ scheduleName
 +'"');
        return;
    }
    durationCalculator.setSchedule(grSched.getUniqueValue());
}
```

Using relative duration

Relative duration is very similar to simple duration except a piece of script is used to determine what parts of a day to remove from the difference calculation.

This script is stored in table cmn_relative_duration and can be examined by navigating to **System Scheduler** > **Schedules** > **Relative Durations**. There are some example relative duration scripts in the out-of-the-box instance.

A relative duration sys_id is passed to the method *`calcRelativeDuration()`* of the global API DurationCalculator class after initialization. When this method is called,

the DurationCalculator object is passed to the relative duration script (stored in table cmn_relative_duration) as the variable *calculator*. So, the relative duration script you write and store in cmn_relative_duration has access to the executing DurationCalculator through the variable *calculator*.

The following script demonstrates how to use *DurationCalculator* to calculate a relative duration.

```
/**
 * Sample use of relative duration calculation.
 *
 */

gs.include('DurationCalculator');
executeSample();

/**
 * Function to house the sample script.
 */
function executeSample(){

    // First we need a DurationCalculator object. We will also
 use
    // the out-of-box relative duration "2 bus days by 4pm"
    var dc = new DurationCalculator();
    var relDur = "3bf802c20a0a0b52008e2859cd8abcf2"; // 2 bus
 days by 4pm if before 10am
    addSchedule(dc);

    // Since our start date is before 10:00am our result is two
 days from// now at 4:00pm.
    var gdt = new GlideDateTime("2012-05-01 09:00:00");
    dc.setStartDateTime(gdt);
    if(!dc.calcRelativeDuration(relDur)){
      gs.log("*** calcRelativeDuration failed");
      return;
    }
    gs.log("Two days later 4:00pm: "+ dc.getEndDateTime());

    // Since our start date is after 10:00am our result is three
 days from
    // now at 4:00pm.
    var gdt = new GlideDateTime("2012-05-01 11:00:00");
    dc.setStartDateTime(gdt);
    if(!dc.calcRelativeDuration(relDur)){
      gs.log("*** calcRelativeDuration failed");
      return;
    }
    gs.log("Three days later 4:00pm: "+ dc.getEndDateTime());}

/**
 * Add a specific schedule to the DurationCalculator object.
 *
 * @param durationCalculator An instance of DurationCalculator
 */
function addSchedule(durationCalculator){
```

```
    // Load the "8-5 weekdays excluding holidays" schedule into
 our duration calculator.
    var scheduleName ="8-5 weekdays excluding holidays";
    var grSched =new GlideRecord('cmn_schedule');
    grSched.addQuery('name', scheduleName);
    grSched.query();
    if(!grSched.next()){
      gs.log('*** Could not find schedule "'+ scheduleName
 +'"');
      return;}

 durationCalculator.setSchedule(grSched.getUniqueValue(),"GMT"
);}
```

Querying tables in script

Using methods in the GlideRecord API, you can return all records from a table, return records from a table that satisfy specific conditions, or return records that include a string from a single table or from multiple tables in a text index group.

Query tables using the GlideRecord API. For API reference, see GlideRecord - Scoped.

Return all records from a table

To query a table, first create a GlideRecord object. To create a GlideRecord, create the following in script:

```
var target = new GlideRecord('incident');
```

Creating a GlideRecord creates a `target` variable which is a GlideRecord object for the incident table.

To process all records from the incident table, add the following script:

```
target.query(); // Issue the query to the database to get all
 records
while (target.next()) {
  // add code here to process the incident record
}
```

This script issues the *`query()`* to the database. Each call to *`next()`* would load the next record for processing.

Return records from a table that satisfy query conditions

Most of the time, you want to retrieve a specific record or a specific set of records, and you have some criteria (query conditions) that define the records you want to obtain. For example, say that you want to obtain all the incident records that have a priority value of 1. Here is the code that would accomplish that.

```
var target = new GlideRecord('incident');
target.addQuery('priority',1);
target.query(); // Issue the query to the database to get
 relevant records
while (target.next()) {
  // add code here to process the incident record
}
```

Notice that the example script includes `target.addQuery('priority', 1);`. This line indicates that you only want the records where the *priority* field is equal to 1. In general, most queries that you want to perform are equality queries; queries where you want to find records with a field equal to a value. For this reason, you do not need to provide an equality operator. However, lets say you wanted to find all incidents where the *priority* field is greater than 1. In this case, you would provide the operator that you want to apply to the query.

```
var target = new GlideRecord('incident') ;
target.addQuery('priority','>',1);
target.query(); // Issue the query to the database to get
 relevant records
while (target.next()) {
  // add code here to process the incident record
}
```

Return records from a table that include a string

Use the `'123TEXTQUERY321'` reserved name to search for string matches in all fields on a table. For example, this script returns records from the Incident table with field values that include the 'email' string.

```
var now_GR = new GlideRecord('incident');
gr.addQuery('123TEXTQUERY321', 'email');
gr.query();
```

`'123TEXTQUERY321'` is a reserved option for the *name* parameter in the `addQuery()` method. You can use this option in an encoded query string. For example, instead of `gr.addQuery('123TEXTQUERY321', 'email');`, you can use `gr.addEncodedQuery('123TEXTQUERY321=email')`.

String search is case-insensitive. The system returns the same results whether you search for `email`, `Email`, or `EMAIL`.

Note:
Before you can query using a string search, you must configure text indexing (and optionally search attributes) for the table you want to search. For more information, see Configure a single table for indexing and searching.

Return records from multiple tables in a text index group that include a string

Use the `'123TEXTINDEXGROUP321'` reserved name to search for a string in a table from a text index group. This option returns results with relevancy scores computed using the text index group's settings.

Note:
You can only query one table in the text index group at a time. Use this option when you want to issue multiple single-table queries and merge their results, ordering them by relevancy score. The advantage of this option is that search result relevancy scores for the individual table queries are normalized consistently for all tables in the text index group.

For example, this script returns records from the Knowledge table that include the keyword 'email', with relevancy scores computed for the 'portal' index group.

```
var now_GR = new GlideRecord('kb_knowledge');
gr.addQuery('123TEXTQUERY321', 'email');
gr.addQuery('123TEXTINDEXGROUP321', 'portal');
gr.query();
```

You can create a similar query for each additional table in the 'portal' index group that you want to search, and then merge the individual queries' results, displaying the results with the highest relevancy scores first. Because all of these search queries use the same text index group search settings, the relevancy scores for their results are all normalized consistently. If you searched the same set of tables without using the `gr.addQuery('123TEXTINDEXGROUP321', 'portal')` method, the individual queries' relevancy scores would be normalized differently and would not be a useful basis for sorting the merged result set.

`'123TEXTINDEXGROUP321'` is a reserved option for the *name* parameter in the `addQuery()` method. You can use this option in an encoded query string. For example, instead of `gr.addQuery('123TEXTINDEXGROUP321', 'portal');`, you can use `gr.addEncodedQuery('123TEXTINDEXGROUP321=portal')`.

Multi-table string search is case-insensitive. The system returns the same results whether you search for `email`, `Email`, or `EMAIL`.

Note:
Before you can query tables in an index group, you must configure text indexing and search attributes for those tables and include them in the index group. For more information, see Configure multiple tables for indexing and searching. All tables in the index group must use the V4 indexing format.

Available JavaScript operators

Describes the operators that can be used within an *addQuery()* request.

Available JavaScript Operators

Field	Definition	addQuery
=	Field must be equal to value supplied.	addQuery('priority', '=', 1);
>	Field must be greater than value supplied.	addQuery('priority', '>', 1);
<	Field must be less than value supplied.	addQuery('priority', '<', 3);
>=	Field must be equal or greater than value supplied.	addQuery('priority', '>=', 1);
<=	Field must be equal or less than value supplied.	addQuery('priority', '<=', 3);
!=	Field must not equal the value supplied.	addQuery('priority', '!=', 1);
STARTSWITH	Field must start with the value supplied. The example shown on the right returns all records where the *short_description* field starts with the text Error.	addQuery('short_description', 'STARTSWITH', 'Error');
CONTAINS	Field must contain the value supplied somewhere in the text. The example shown on the right returns all records where the *short_description* field contains the text Error anywhere in the field. **Note:** The LIKE operation is not supported. Administrators must use CONTAINS in the query.	addQuery('short_description', 'CONTAINS', 'Error');

Available JavaScript Operators (continued)

Field	Definition	addQuery
IN	Takes a map of values that allows commas, and gathers a collection of records that meet some other requirement. Behaves as `Select * from <table> where short_description IN ('Error', 'Success', 'Failure')`, which is identical to `Select * from <table> where short_description='Error'`. For example, to query all variable values that belong to a specific Activity, use the *IN* clause, and store their sys_ids in a map, or comma-separated list. Then query the variable value table and supply this list of sys_ids.	addQuery('short_description', 'IN', 'Error,Success,Failure');
ENDSWITH	Field must terminate with the value supplied. The example shown on the right returns all records where the *short_description* field ends with text Error.	addQuery('short_description', 'ENDSWITH', 'Error');
DOES NOT CONTAIN	Selects records that do NOT match the pattern in the field. This operator does not retrieve empty fields. For empty values, use the operators "is empty" or "is not empty."The example shown on the right returns all records where the *short_description* field does not have the word "Error."	addQuery('short_description', 'DOES NOT CONTAIN', 'Error');
NOT IN	Takes a map of values that allows commas, and gathers a collection of records that meet some other requirement. Behaves as: `Select * from <table> where short_description NOT IN ('Error')`.	addQuery('short_description', 'NOT IN', 'Error,Success,Failure');
INSTANCEOF	Special operator that retrieves only records of a specified "class" for extended tables. The code example on the right, shows how to retrieve all configuration items that are classified as computers.	addQuery('sys_class_name', 'INSTANCEOF', 'cmdb_ci_computer');

For additional information on the operators that are available for filters and queries, see Operators available for filters and queries.

There are also some special methods that you can use to search for data that is NULL or NOT NULL. To search for all incidents where the *short_description* field has not been supplied (is null), use the following query:

```
var target = new GlideRecord('incident');
target.addNullQuery('short_description');
target.query(); // Issue the query to the database to get all
 records
while (target.next()) {
  // add code here to process the incident record
}
```

To find all incidents in which a *short_description* has been supplied, use the following query:

```
var target = new GlideRecord('incident');
target.addNotNullQuery('short_description');
target.query(); // Issue the query to the database to get all
 records
while (target.next()) {
  // add code here to process the incident record
}
```

For more information on the *GlideRecord* API and its available methods, see GlideRecord.

GlideRecord query examples

These examples demonstrate how to perform various *GlideRecord* queries.

query

```
var rec = new GlideRecord('incident');
rec.query();
while(rec.next()) {
  gs.print(rec.number + ' exists'); }
```

update

```
var rec = new GlideRecord('incident');
rec.addQuery('active',true);
rec.query();
while(rec.next()) {
  rec.active = false;
  gs.print('Active incident ' + rec.number = ' closed');
  rec.update(); }
```

insert

```
var rec = new GlideRecord('incident');
rec.initialize();
rec.short_description = 'Network problem';
rec.caller_id.setDisplayValue('Joe Employee');
rec.insert();
```

delete

```
var rec = new GlideRecord('incident');
rec.addQuery('active',false);
rec.query();
while(rec.next()) {
  gs.print('Inactive incident ' + rec.number + ' deleted');
  rec.deleteRecord(); }
```

Querying Service Catalog Tables

You cannot directly query the variables of the Service Catalog Request Item table [sc_req_item]. Instead, query the Variable Ownership table, [sc_item_option_mtom], by adding two queries, one for the variable name and another for the value. The query returns the many-to-many relationship, which you can dot-walk to the requested item. The following example finds the

request items that have the variable *item_name* with a value of *item_value* and displays the request item numbers:

```
var now_GR = new GlideRecord('sc_item_option_mtom');
gr.addQuery('sc_item_option.item_option_new.name','item_name');
gr.addQuery('sc_item_option.value','item_value');
gr.query();

while(gr.next()) {
  gs.addInfoMessage(gr.request_item.number); }
```

For additional information see GlideRecord.

Running order guides automatically

Service catalog order guides allow customers to make a single service catalog request that can generate several ordered items. Administrators can configure order guides to run automatically, from a workflow or a script to generate a set of ordered items without manually submitting a service catalog request. Administrators can also review and reprocess the order guide failures.

As a use case, an onboarding workflow for a new employee can run an order guide to automatically order items for that employee.

Note:
You can only save catalog items, not the order guide (that is, initial landing page options).

Running order guides from scripts

Running order guides with a server-side script is more complex than using workflows, but it allows more flexibility and can be used in non-workflow situations.

For example, you can use order guide scripts with UI actions or server-side business rules.

Note:
When order guides run automatically, order guide UI policies are not enforced. Also, options in the Choose Options screen cannot be selected, so make sure that order guide rules define sensible defaults for these options to avoid processing failures.

Use the *SNC.ScriptableOrderGuide* Java class to run order guides with a script.

Use the *SNC.ScriptableOrderGuide(String orderGuideId)* constructor to create a new ScriptableOrderGuide object.

Method summary

Method	Return Value	Description
process(String json)	boolean	Runs the order guide using the JSON encoded string parameter as the input for the order guide. Returns *true* or *false* depending on whether processing was successful or not. **Note:** Both *opened_by* and *requested_for* parameters must be passed to the order guide, and both must have valid user record sys_id values.

Method	Return Value	Description
		If processing is successful and a request is created by the order guide, you can retrieve the request GlideRecord using *getRequest()*. If the processing fails, you can retrieve the failure GlideRecord using *getFailure()*, then submit the script for reprocessing using reprocess.
reprocess(GlideRecord failure)	boolean	Runs the order guide again using the JSON encoded string parameter stored in the failure GlideRecord.
getMessage()	String	Retrieves the message populated after processing or reprocessing.
getRequest()	GlideRecord	Retrieves the request GlideRecord.
getFailure()	GlideRecord	Retrieves the failure GlideRecord from the Scriptable Order Guide Failures [sc_script_order_guide_failure] table.

Script example

This script processes an order guide called *IT Onboarding SOG*.

```
// Creating the object to later be JSON encoded
var json =
 {"opened_by":"62826bf03710200044e0bfc8bcbe5df1","requested_for"
:"06826bf03710200044e0bfc8bcbe5d8a","department":"221f3db5c61122
84009f4becd3039cc9"};

var now_GR = new GlideRecord("sc_cat_item_guide");
if (gr.get("name","IT Onboarding SOG")) {
  var sog = new SNC.ScriptableOrderGuide(gr.getValue("sys_id"));
  var result = sog.process(new JSON().encode(json));
  if(!result)
    gs.log("Processing the scriptable order guide failed with
 message: " + sog.getMessage());
  else {
    var request = sog.getRequest();
    gs.log("Request created - " + request.sys_id); } }
```

Running order guides from workflows

Running an order guide from a workflow is suitable if you include order guides as part of a broader workflow-based process.

For example, an activity within an onboarding workflow for a new employee can automatically run an order guide to order items for that employee.

Note:
When order guides run automatically, order guide UI policies are not enforced. Also, options in the Choose Options screen cannot be selected, so make sure that order guide rules define sensible defaults for these options to avoid processing failures.

To run order guides from a workflow, use the **Scriptable Order Guide** workflow activity.

Running order guides from workflows

Field	Description
Order Guide	The name of the order guide that this activity processes. For example, **Example Employee Onboarding IT**.
Script	A script passing information to the order guide. This information is sent as a JSON encoded string parameter assigned to the *answer* variable. The script must meet these requirements: • The names of the variables in the script must match the names used within the order guide. For example, if the order guide uses a *department* variable in a rule condition, the script must also pass a *department* parameter. • Both *opened_by* and *requested_for* parameters must be passed to the order guide, and both must have valid user record sys_id values.

Results

- **Success**: the activity successfully processed the order guide. This does not mean a request was created. If a request was created, the request sys_id is added to the workflow scratchpad under the *sc_request* variable.
- **Failure**: while processing the order guide a failure occurred, creating a failure record. If the processing fails, you can view and edit the failure record.

Workflow example

The **Example Employee Onboarding IT Workflow** workflow uses this example to generate IT catalog items for a new employee as part of an onboarding process.

The activity uses this script to:

1. Take a JSON string generated previously from the HR change record.
2. Append the mandatory *opened_by* and *requested_for* parameters to that string.
3. Submit the new string for processing by the order guide.

```
var parameters  = new JSON().decode(current.payload);

// Need to amend the json object to include additional values.
parameters.opened_by = current.opened_by + "";
parameters.requested_for = current.opened_for + "";

answer = new JSON().encode(parameters);
```

View order guide failures

Order guide processing can fail, for example if the order guide being run does not exist. When a failure occurs during the order guide processing, the **Scriptable Order Guide Failures** submodule allows you to review and reprocess the failures. A record is created for each failure and once you fix the errors that caused the initial failure, you can reprocess the failed order guides.

About this task

If a failure occurs, a failure record is created in the Scriptable Order Guide Failures [sc_script_order_guide_failure] table.

To view details of a failure, navigate to **Service Catalog** > **Catalog Administration** > **Scriptable Order Guide Failures**, then open a failure record.

Reprocessing failures: If you have fixed the error that caused the initial failure, you can reprocess failed order guides.

Procedure

1. Navigate to **Service Catalog** > **Catalog Administration** > **Scriptable Order Guide Failures**.
2. Open the failure record.
3. Click the **Reprocess** related link.

To reprocess one or more failures:

a. Navigate to **Service Catalog** > **Catalog Administration** > **Scriptable Order Guide Failures**.
b. Select the check box beside one or more records to reprocess.
c. Select **Reprocess** from the **Actions** choice list.

Scriptable assignment of execution plans

Each catalog item has an associated execution plan, used whenever an item of that type is ordered; if no plan is specified, the default plan is used. This default is effective for most organizations, but your execution plan may need to vary based on additional criteria.

For example, in the base system service catalog, a request for a new PC always uses the PC Delivery Plan. However, this plan may need to vary for unusual circumstances - such as when a requester is working from home, at a remote location.

To provide this flexibility, you can use a script to override the default execution plan on a specific catalog item.

Limitations during script execution

Execution plan scripts have limitations that need to be considered during their implementation.

While the execution plan script runs:

- You cannot interact with any catalog tasks as catalog tasks are only created after the execution plan is selected.
- Some fields such as **total delivery time** and **due date** are not yet calculated, although the request itself is available within the script via `current.request()`.
- Approvals have not yet been generated.

Writing the scripts

Follow these guidelines when writing execution plan scripts.

Execution plan scripts can access the same global variables and other functions as in any other server side execution plan.

- `current` is the currently-requested catalog item, `sc_req_item`.
- `current.delivery_plan()` is the assigned execution plan for this catalog item.

The evaluated value from the script is used as the `sys_id` of the execution plan.

Simple example:

```
current.delivery_plan.setDisplayValue('PC Delivery Plan')
```

If an invalid value is returned, such as undefined or not found, then the existing assigned value is used.

More complex example:

```
getexecutionplan();
function getexecutionplan() {
var location =
 current.request.requested_for.location.getDisplayValue();
// if we're in Atlanta
if (location == 'Atlanta') {
   // use the remote pc delivery plan instead of the normal one
   var remote_plan = new
 GlideRecord('sc_cat_item_delivery_plan');
   remote_plan.addQuery('name', 'Remote PC Delivery Plan');
   remote_plan.query();
   remote_plan.next();
   current.delivery_plan = remote_plan.sys_id;
   return remote_plan_sys_id;
   }
   return current_delivery_plan;
}
```

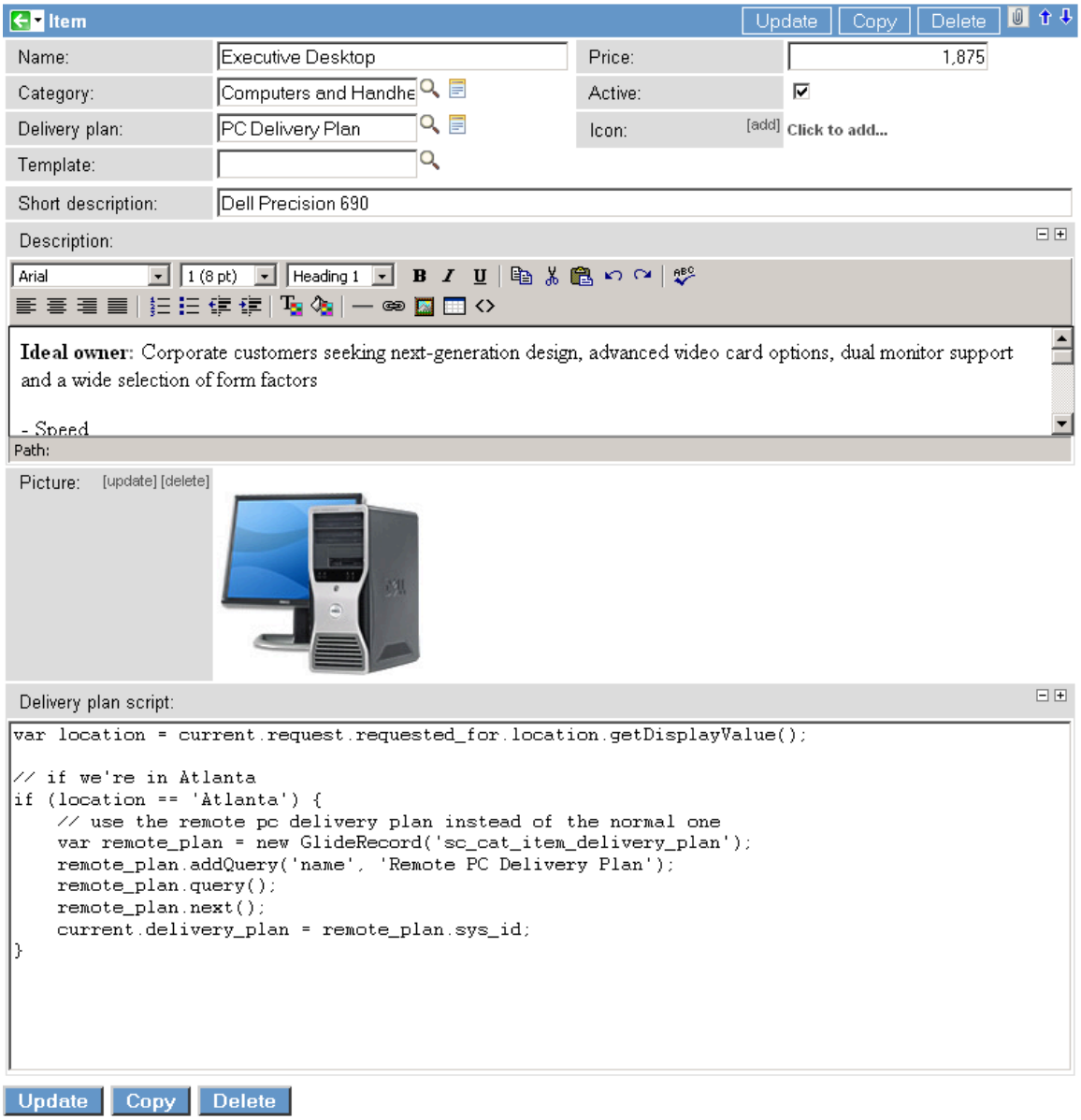
In this example, any time a request is for a user in Atlanta, ServiceNow uses the Remote PC Delivery Plan. Otherwise, the execution plan is not overridden and ServiceNow uses the catalog item's normal execution plan, the PC Delivery Plan.

Add a script to a catalog item

You can add a script to a catalog item so that the script runs each time a user requests that item.

Procedure

1. Navigate to **All** > **Service Catalog** > **Maintain Items**.
2. Select the relevant catalog item to which you wish to add the script.
3. Configure the catalog item form to add the execution plan script field, often named **Delivery Plan Script**.
4. Fill in the script details.



5. Update the item form with your changes.

Result

The script runs each time that item is requested, selecting the execution plan to run with that item.

Use a script to approve an execution plan

You can use an approval rule script to approve an execution plan.

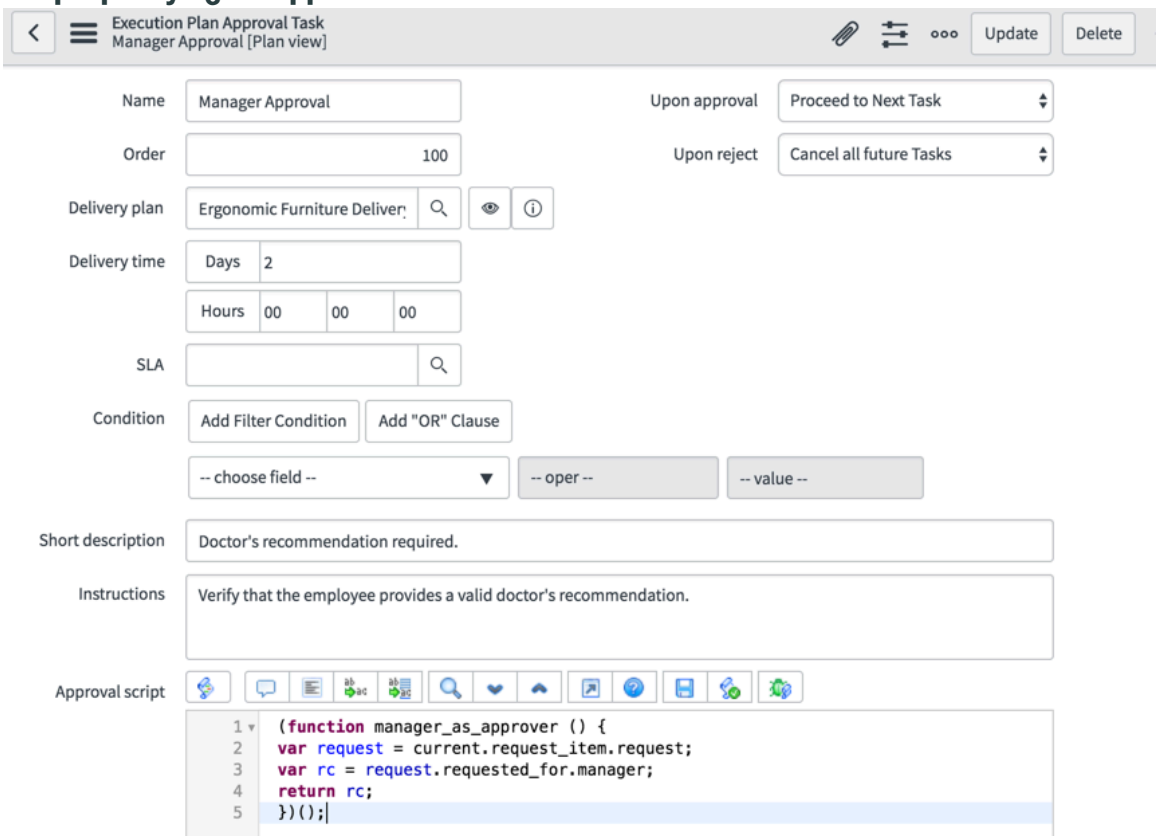
Procedure

1. Retrieve an approval execution plan task.
2. View the **Approval Script** field.
3. Fill in an approval script using the same syntax and rules you would use on an approval rule.

Example:

For example, in the script below, the requester's manager is the approver.

Script specifying the approver



Using regular expressions in server-side scripts

JavaScript regular expressions automatically use an enhanced regex engine, which provides improved performance and supports all behaviors of standard regular expressions as defined by Mozilla JavaScript. The enhanced regex engine supports using Java syntax in regular expressions.

The *SNC.Regex* API is not available for scoped applications. For scoped applications, remove the *SNC.Regex* API and use standard JavaScript regular expressions.

For more information on JavaScript regular expressions, see the Mozilla JavaScript documentation on regular expressions and RegExp.

Using Java syntax in JavaScript regular expressions

The enhanced regex engine includes an additional flag to allow Java syntax to be used in JavaScript regular expressions.

Regular expressions with the additional flag work in all places that expect a regular expression, such as `String.prototype.split` and `String.prototype.replace`. To use Java syntax in a regular expression, use the Java inline flag j, for example `/(?ims)ex(am)ple/j`

Extended regular expression flags

Flag	Description
j	Defines a regular expression that executes using the Java regular expression engine. It can be used to access Java-only features of regular expressions (such as look behind, negative look behind) or to use Java regular expressions without translating them into JavaScript regular expressions. For example: `var regex = /ex(am)ple/j;`

Convert SNC Regex expressions to enhanced regex expressions

When you upgrade to Eureka Patch 5 or later releases, you should convert scripts that use the *SNC.Regex* API to use regular JavaScript expressions.

Procedure

1. From the original expression, such as: `SNC.Regex("/expr/is");`, create a new regular expression object using the pattern with the slashes stripped.

   Example

   ```
   new RegExp('expr');
   ```

2. Move the *SNC.Regex* flags to the start of the expression using Java's inline flag special construct.

   Example

   ```
   new RegExp('(?is)expr');
   ```

3. Add the *j* flag to the `RegExp` to tell the engine to treat the expression as a Java expression.

   Note:
   If you know that the script being converted does not use Java syntax, it is not necessary to use the *j* flag.

   Example

   ```
   new RegExp('(?is)expr', 'j');
   ```

4. Add the *g* flag to handle multiple matches or a global replace.

   Example

   ```
   new RegExp('(?is)expr', 'jg');
   ```

Example:

Using SNC.Regex:

```
var r = new SNC.Regex('/world/');
var str = 'helloworld';
var replaced = r.replaceAll(str, 'there');
// replaced == 'hellothere'
```

Using a JavaScript regular expression:

```
var r = new RegExp('world', 'jg');
var str = 'helloworld';
var replaced = str.replace(r, 'there');
// replaced == 'hellothere'
```

Scriptable service catalog variables

You can use scripting to reference any request item variable from a table in scoped and non-scoped environment.

An example of a variable reference follows.

```
current.variables.<variable_name>
```

Where current refers to the current record, and *<variable_name>* is the name of your variable.

Note:
In order to reference a variable from JavaScript, it must have a name.

When a variable is part of a variable set, you can reference it as current.variables.<variable_name> or current.variables.<variable_set_name>.<variable_name>.

Variable set is also a first-class citizen in Service Catalog. Like variables, a variable set has read, write, and create roles. If roles are provided for a variable set, the roles are applicable for the variables within the set. Roles of an individual variable are overridden by the roles of the variable set.

Print a variable

```
var original = current.variables.original_number;
gs.print(original);
```

Set a variable

```
current.variables.name = "Auto-Generated:" +
 current.variables.asset_tag;
```

Create an inventory item with fields set from variables

```
doCreation();

function doCreation ( ) {
var create = current.variables.create_item;
if (create == 'true') { // we want to create an asset
  var computer = new GlideRecord('cmdb_ci_computer');
  computer.initialize();
  computer.asset_tag = current.variables.asset_tag;
  computer.serial_number = current.variables.serial_number;
  computer.name = current.variables.name;
  computer.manufacturer = current.variables.company;
  computer.insert(); } }
```

Get GlideElementVariable of variables and variable sets associated with a GlideRecord

```
now_GR.variables
```

Get the name value pair of variables associated with a GlideRecord

```
now_GR.variables.getVariableValue();
```

Get a list of GlideElementVariable for variables within a task record

```
now_GR.variables.getElements();
```

Get a list of GlideElementVariable for variables (including multi-row variable set) within a task record

```
now_GR.variables.getElements(true);
```

Note:
The `getElements()` method returns all the variables present on the given task record except for the formatter variables like label, break, container end, container split, and so on.

APIs for GlideElementVariable

- `now_GR.variables.<var_name>.isMultiRow()`: Get whether the GlideElementVariable is a multi-row variable set or a variable.
- `now_GR.variables.<var_name>.getQuestion()`: Get the Question object for a variable. Applicable only for a variable (`isMultiRow()` is false) and not for a multi-row variable set.
- `now_GR.variables.<var_name>.getLabel()`: Get the label of the GlideElementVariable. For a variable, the label of the variable is returned. For multi-row variable set, the title of the variable set is returned.
- `now_GR.variables.<var_name>.canRead()`: Get whether the user can view a variable or multi-row variable set.
- `now_GR.variables.<var_name>.canWrite()`: Get whether the user can edit a variable or multi-row variable set.
- `now_GR.variables.<var_name>.getDecryptedValue()`: Get the decrypted value for a masked variable. Applicable only for a masked variable.
- `now_GR.variables.<var_name>.getRows()`: Get the list of row objects for a multi-row variable set. Applicable only for a multi-row variable set (`isMultiRow()` is true).
- `now_GR.variables.<var_name>.getRowCount()`: Get the number of rows for multi-row variable set. Applicable only for a multi-row variable set (`isMultiRow()` is true).

Example to access variables of GlideRecord for the Task table

```
var now_GR = new GlideRecord('sc_req_item');
if (now_GR.get('635a1f538732030Oe0ef0cf888cb0b73')) {
    var variables = now_GR.variables.getElements();
    for (var i=0;i<variables.length;i++) {
        var question = variables[i].getQuestion();
        gs.log(question.getLabel() + ":" + question.getValue())
    }
}
```

Example to access a multi-row variable set of GlideRecord for the Task table

```
var now_GR = new GlideRecord('sc_req_item');
now_GR.get('02c38dcd87013300e0ef0cf888cb0bb2');

var vars = now_GR.variables.getElements(true);

for (var i=0; i<vars.length; i++) {
    var now_V = vars[i];
    if (now_V.isMultiRow()) {
        var rows = now_V.getRows();
        for (var j=0; j<now_V.getRowCount(); j++) {
            var row = rows[j];
            var cells = row.getCells();
            for (var k=0; k<cells.length; k++) {
                var cell = cells[k];
```

```
                gs.info(cell.getLabel() + ":" +
 cell.getCellDisplayValue())
            }
        }
    }
}
```

Multi-row variable set

Multi-row variable set

Operation	Usage
Table operations	
Return JSON array value as String	`now_GR.variables.table_var`
Set value of a multi-row variable set	`now_GR.variables.table_var = <val>` **Note:** An array of ordered (key, value) pairs is also applicable as input.
Get value of column, var1, of a multi-row variable set	`now_GR.variables.table_var.var1`
Set value of a variable set, var1	`now_GR.variables.table_var.var1 = <val>` **Note:** An array of ordered (key, value) pairs is also applicable as input.
Row operations	
Get the current row count	`now_GR.variables.table_var.getRowCount()`
Returns the row specified by the variable "i" - getRow(<int> i)	`var row = now_GR.variables.table_var.getRow(<int> i);`
Get the cell value for a question column mapped to <var_name>	`row.<var_name>`
Set the cell value for a question column mapped to <var_name>	`row.setCellValue('<var_name>', value)`
Set the cell value for a question column mapped to <var_name>	`row.<var_name> = value`

Multi-row variable set (continued)

Operation	Usage
Add an empty row at the end of the table and return a scriptable object	`var row = now_GR.variables.table_var.addRow()`
Delete a row	`row.deleteRow()`

Notes and limitations

- //Single column of table_Var
- now_GR.variables.table_var.var1
- now_GR.variables.table_var.var1 = <val>

1. You can only set a variable in a before business rule. Variables set in an after rule are not written to the database.
2. There is nothing in place to prevent namespace collision with variables. Creating two variables named computer_speed would result in only one of them showing up; the second one would overwrite the first one.
3. Date/time variables use the same time zone formatting and storage rules as all other dates in the system. They are stored internally in GMT, but translated into the user's local time zone and format for display.

Setting a GlideRecord variable to 'NULL'

GlideRecord variables (including current) are initially null in the database. Setting these back to an empty string, a space, or the JavaScript null value will not result in a return to this initial state.

Note:
This functionality requires a knowledge of JavaScript.

To set it back to the initial state, simply set the value to 'NULL'. Note that the *update()* function does not run on the current object but rather on the record. The object displays the initial value until it is called again from the record.

Note:
Functionality described here requires the **Admin** role.

Example 1

```
var grIncident = new GlideRecord('incident');
grIncident.addNotNullQuery("assigned_to");
grIncident.query();
if (grIncident.next()) {
  gs.log(“The incident record that is going to be updated is “ +
 grIncident.number);
  gs.log("Previous Value of 'Assigned To' field: " +
 grIncident.assigned_to);
  grIncident.assigned_to = "NULL";
  grIncident.update();
```

```
  gs.log("Current Value of 'Assigned To' field: " +
 grIncident.assigned_to);
}
```

Example 2 (Business Rule)

```
current.u_affected_value = 'NULL';
current.update();
```

Schedule Pages

A schedule page is a record that contains a collection of scripts that allow for custom generation of a calendar or timeline display.

Creation of timeline schedule pages requires understanding of the page/event flow and the ability to write client and server side JavaScript.

Schedule pages form

To access schedule pages, navigate to **System Scheduler** > **Schedules** > **Schedule Pages**.

The form provides the following fields, depending upon the *View Type* selected:

Schedule pages form

Field	Field Type	Description
Name	String	General name that is used to identity the current schedule page.
Schedule type	String	The schedule type is a string that is used to uniquely identity the schedule page via the "sy URI parameter. For example, a schedule page could be accessed as follows: `/show_schedule_page.do?sysparm_type=gantt_chart&sysparm_timeline_task_id=d530bf907f0` Alternatively, the schedule page can also be accessed by setting the "*sysparm_page_* of the unique 32 character hexadecimal system identifier of the schedule page.
View Type	Choice	Each view type displays different field combinations. There are two options available: • Calendars • Timelines
Description	String	General description that provides additional information about the current schedule page.
Init funciton name	String	**Note:** This functionality is only used by *Calendar* type schedule pages. The init function name specifies the name of the JavaScript function to call inside the *Clie* type schedule pages.
HTML	String	**Note:** This functionality is only used by *Calendar* type schedule pages. The HTML field is a scriptable section that is parsed by Jelly and injected into the display p calendar. It can be used to pass in variables from the server and define extra fields are nec

Schedule pages form (continued)

Field	Field Type	Description
Client script	String	The client script is a scriptable section that allows for configuring options of the schedule different depending on the schedule page view type and is discussed below.
Server AJAX processor	String	**Note:** This functionality is only used by *Calendar* type schedule pages. The Server AJAX processor is specific to calendar type schedule pages that is used to retu and spans to be displayed.

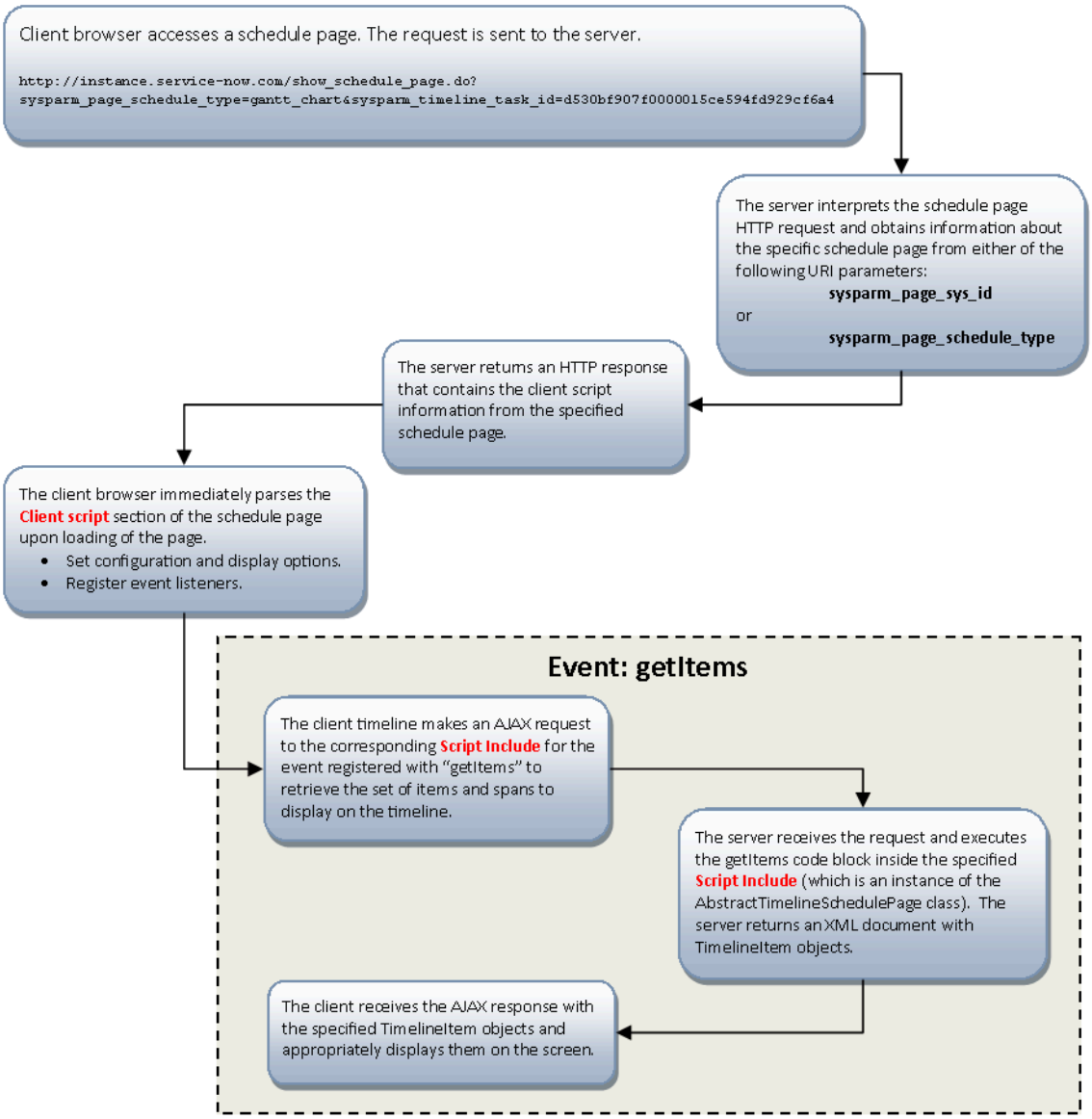
Timeline schedule pages

A Timeline Schedule Page is a specific record that contains configuration information for displaying time based points and spans in a "timeline" like fashion.

The timeline schedule page references a script include that extends from AbstractTimelineSchedulePage to perform dynamic modification to the timeline based on different events and conditions. Both the schedule page and the script include for timeline generation allow for extreme customization and their corresponding application programming interface (API) is documented below.

The following diagram shows the series of events that occur when a timeline schedule page is accessed. Once the timeline has been loaded, all subsequent events, such as events resulting from timeline interaction (e.g. moving a timeline span), follow the same logic flow shown in the gray event box.

Timeline Flow



Applications that use schedule pages to generate time lines

- Project Management
- Maintenance Schedules
- Group On-Call Rotation
- Field Service Management

Timeline schedule page example

The following example demonstrates how to create a timeline schedule page with corresponding script include utilizing a majority of the API described above.

For this example we are going to create an Incident Summary Timeline for a project support manager to visualize all of the new incidents. All new incidents should be displayed as single points where the priority of the incident is distinguished by a different point icon. Additionally, all closed incidents should be displayed on the timeline in a separate group that shows the duration of the incident before it was closed. Since the Project Manager wants to be able to easily close

new items that are resolved without using any form lists, we will handle the vertical move event allowing the new incidents to be dragged into the closed incident group or items within.

Schedule Page

Create a new schedule page with the following properties:

- **Name**: Hardware Incidents
- **Schedule type**: incident_timeline
- **View Type**: Timeline
- **Client Script**:

```
// Set our page configuration
glideTimeline.setReadOnly(false);
glideTimeline.showLeftPane(true);
glideTimeline.showLeftPaneAsTree(true);
glideTimeline.showTimelineText(true);
glideTimeline.showDependencyLines(false);
glideTimeline.groupByParent(true);
glideTimeline.setDefaultPointIconClass('milestone');

// We will define what items to display and provide a custom
 event handler for moving new items to the closed state
glideTimeline.registerEvent('getItems',
 'IncidentTimelineScriptInclude');
glideTimeline.registerEvent('elementMoveY',
 'IncidentTimelineScriptInclude');
```

Script Include

Now that the schedule page has been created we need to generate a matching script include for the events that were registered. Create a new script include with the following properties:

- **Name**: IncidentTimelineScriptInclude
- **Active**: Checked
- **Glide AJAX enabled**: Checked
- **Script**:

```
// Class Imports

var IncidentTimelineScriptInclude = Class.create();
IncidentTimelineScriptInclude.prototype =
 Object.extendsObject(AbstractTimelineSchedulePage, {

  ////////////////////// //
 GET_ITEMS ////////////////////////////////////
  getItems:function() {
    // Specify the page title
    this.setPageTitle('My Custom Incident Summary Timeline');

    var groupNew = new GlideTimelineItem('new');
    groupNew.setLeftLabelText('New Incidents');
    groupNew.setImage('../images/icons/all.gifx');
    groupNew.setTextBold(true);
```

```
   this.add(groupNew);

   var groupClosed = new GlideTimelineItem('closed');
   groupClosed.setLeftLabelText('Closed Incidents');
   groupClosed.setImage('../images/icons/all.gifx');
   groupClosed.setTextBold(true);
   groupClosed.setIsDroppable(true);

   // This allows us to drag an open incident onto the closed
group row.
   this.add(groupClosed);

   // Get all the incidents and let's add only the new/closed
ones appropriately
   var now_GR = new GlideRecord('incident');
   gr.query();
   while(gr.next()) {
      // Only loop through new/closed incidents
      if(gr.incident_state != '1' && gr.incident_state != '7')
continue;

      // Ok, we have a new/closed incident. Create the item and
the span first.
      var item = new GlideTimelineItem(gr.getTableName(),
gr.sys_id);
      var span = item.createTimelineSpan(gr.getTableName(),
gr.sys_id);

      // Specific properties for a new incident
      if(gr.incident_state == '1') { // New
        item.setParent(groupNew.getSysId());
        item.setImage('../images/icons/open.gifx');

span.setTimeSpan(gr.getElement('opened_at').getGlideObject().ge
tNumericValue(),

gr.getElement('opened_at').getGlideObject().getNumericValue());

        // We will show different colors based upon the
priorities only for new incidents
        switch(gr.getElement('priority').toString()) {
          case '1': span.setPointIconClass('red_circle');
break;
          case '2': span.setPointIconClass('red_square');
break;
          case '3': span.setPointIconClass('blue_circle');
break;
          case '4': span.setPointIconClass('blue_square');
break;
          case '5': span.setPointIconClass('sepia_circle');
break;
          default: // Otherwise, the default point icon class
will be used (Milestone)
         }
        }
       // Specific properties for a closed incident
       else if(gr.incident_state == '7') {
```

```
          item.setParent(groupClosed.getSysId());
          item.setImage('../images/icons/closed.gifx');

 span.setTimeSpan(gr.getElement('opened_at').getGlideObject().ge
tNumericValue(),

 gr.getElement('closed_at').getGlideObject().getNumericValue()
); }

        // Common item properties
        item.setLeftLabelText(gr.short_description);

        // Common span properties
        span.setSpanText(gr.short_description);
        span.setTooltip('<strong>' +
 GlideStringUtil.escapeHTML(gr.short_description) +
 '</strong><br>' + gr.number);
        span.setAllowXMove(false);
        span.setAllowYMove(gr.canWrite() ? true:false);
        span.setAllowYMovePredecessor(false);
        span.setAllowXDragLeft(false);
        span.setAllowXDragRight(false);

        // Now we add the actual item
        this.add(item);
        } } ,


   ////////////////////////// //
 ELEMENT_MOVE_Y //
////////////////////////////////////////////////////////////

   /**
   * This is one of the AbstractTimelineSchedulePage event
 handler methods that corresponds to a vertical
   * move. The arguments for this function are defined in the
 API section of the event handler methods.
   */
  elementMoveY: function(spanId, itemId, newItemId) {

    // Get information about the current incident
    var now_GR = new GlideRecord('incident');
    gr.addQuery('sys_id', spanId);
    gr.query();
    if(!gr.next())
      return this.setStatusError('Error', 'Unable to lookup the
 current incident.');

    // Only allow the new incidents to have their state
 adjusted.
    if(gr.incident_state != '1')
      return this.setStatusError('Error','Only new incidents can
 have their state adjusted.');

    // Get information about the dropped GlideTimelineItem. If
 it was dropped in an item on the "New Incidents"
```

```
   // group let's do nothing. If it was dropped in the "Closed
Incidents" then let's adjust the state automatically.
   var grDropped = new GlideRecord('incident');
   grDropped.addQuery('sys_id', newItemId );
   grDropped.query();
   if(!grDropped.next() || grDropped.incident_state == '7') {
      // This means the dropped item was either the 'Closed
Incidents' group (which has no record or sys_id) or an
      // existing incident that is closed. The 'New Incidents'
also has no sys_id; however, the default behavior for
      // items without a sysId is to be non-droppable. This is
why we explicitly denoted the 'Closed Incidents' to
      // be marked as "droppable".

      // Return a dialog prompt
      this.setStatusPrompt('Confirm', 'Are you sure you want to
close: ' +
            '<div style="margin:10px 0 10px
14px;padding:4px;background-color:#EBEBEB;"><strong>' +
             GlideStringUtil.escapeHTML(gr.short_description)
+
             '</strong><br/><div class="font_smaller">' +
now_GR. number + '</div></div>',
             'this._elementMoveYHandler_DoClose', // This
function is for when the OK button is clicked.
             'this._elementMoveYHandler_DoNothing', // This
function is for when the Cancel button is clicked.
             'this._elementMoveYHandler_DoNothing'); // This
function is for when the Close button is clicked.
      } } ,

 _elementMoveYHandler_DoClose: function(spanId, itemId,
newItemId) {
   // Notice that this function takes the same function
arguments as the original function for which it
   // is a custom event handler for.

   // Update the database record from 'New' to 'Closed'.
   var now_GR = new GlideRecord('incident');
   gr.addQuery('sys_id', spanId);
   gr.query();
   gr.next();
   gr.setValue('incident_state', '7');
   gr.update();

   // This will re-render the timeline showing the updated item
in the closed group.
   this.setDoReRenderTimeline(true);

   // Let's show a success message
   this.setStatusSuccess('Success', '<strong>' +
gr.short_description + '</strong> was successfully
closed.'); } ,

 // Since the user clicked cancel or close we simply do
nothing.
```

```
 _elementMoveYHandler_DoNothing: function(spanId, itemId,
newItemId) { }

});
```

Screenshots / Results

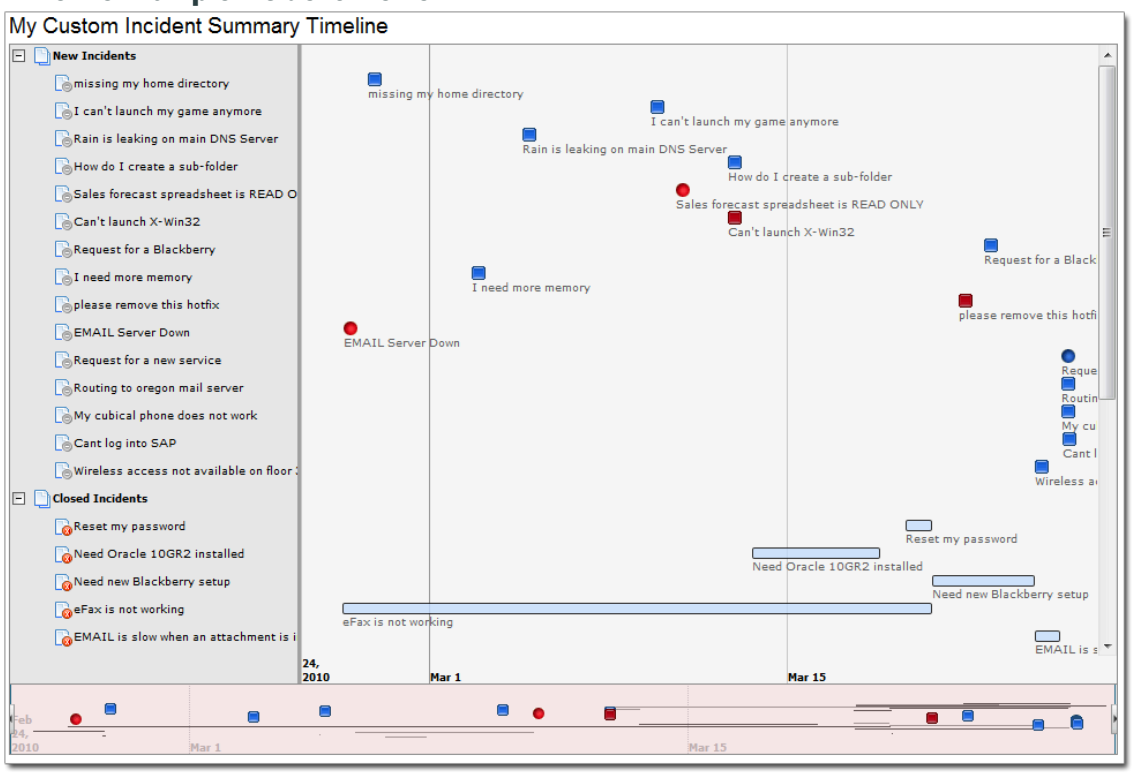
1. Navigate to:

   ```
   http://[YOURINSTANCE]:8080/show_schedule_page.do?
   sysparm_page_schedule_type=incident_timeline
   ```

   Notice the bold text is the value of the schedule page **Schedule type** field.

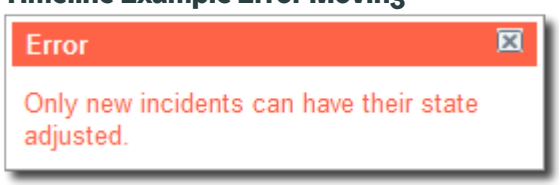
2. The page displays a timeline as specified by the schedule page and script include created. A link to this page can be created and placed as a module or UI action as necessary.

Timeline Example Incident Preview



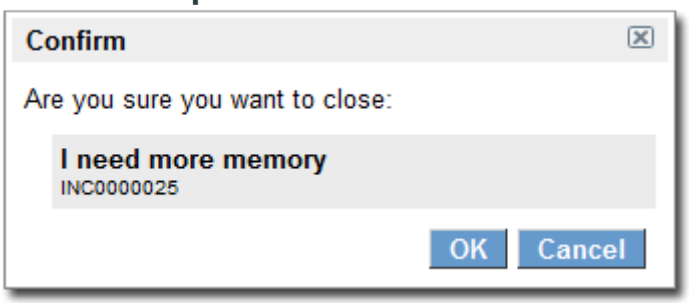
3. Attempting to move a closed incident anywhere displays the expected error message.

Timeline Example Error Moving



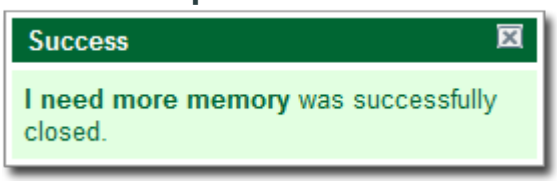
4. Moving the incident: *I need more memory* displays the following confirmation box.

Timeline Example Confirm Close



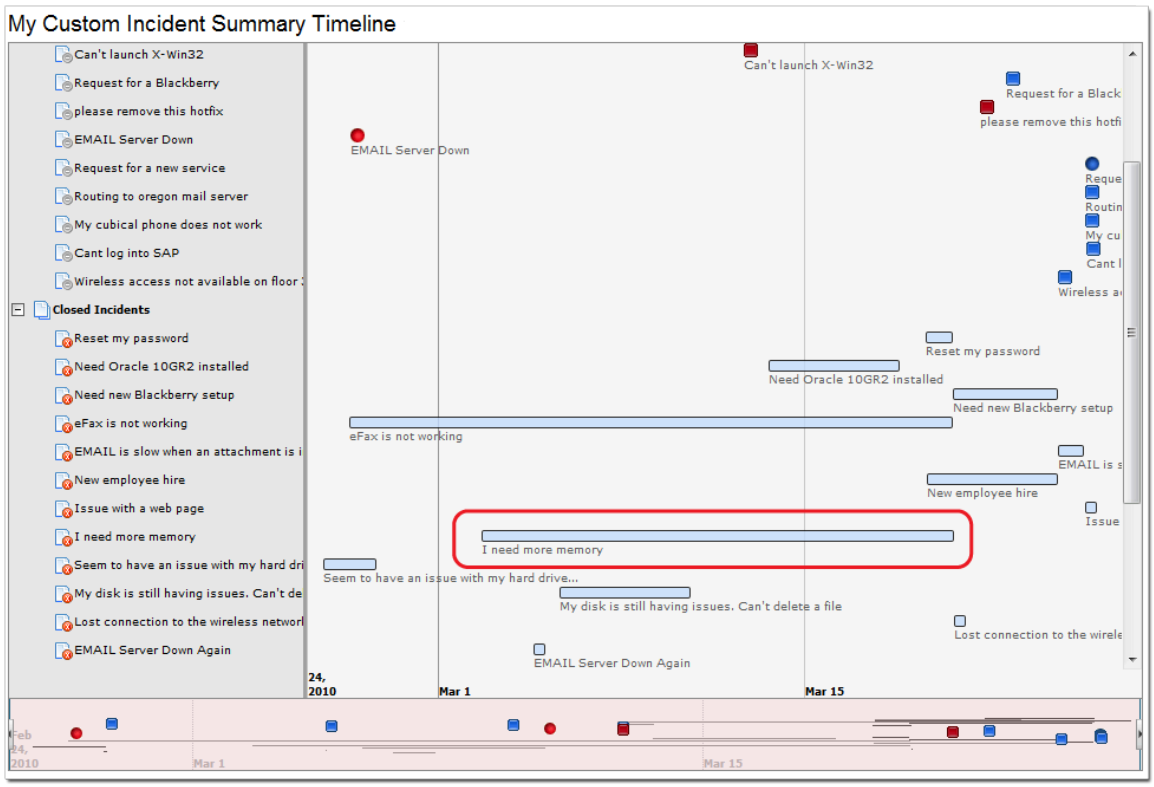
5. Clicking the **Cancel** button closes the overlay. Clicking the **OK** button actually updates the incident_state of the record and then displays the following success box.

Timeline Example Close Success



6. After clicking **OK**, it is clear the incident is now listed in the **Closed Incidents** group.

Timeline Example Incident Updated



XMLDocument script object

A JavaScript object wrapper for parsing and extracting XML data from an XML document (String).

Use this Javascript class to instantiate an object from an XML string, usually a return value from a Web Service invocation, or the XML payload of ECC Queue. Using the XMLDocument object in a Javascript business rule lets you query values from the XML elements and attributes directly.

Constructor

The constructor of a script object creates a new instance of the object to be used.

```
var xmlString = "<test>" +
                "  <one>" +
                "    <two att=\"xxx\">abcd1234</two>" +
                "    <three boo=\"yah\"
 att=\"yyy\">1234abcd</three>" +
                "    <two>another</two>" +
                "  </one>" +
                "  <number>1234</number>" +
                "</test>";

var xmldoc = new XMLDocument(xmlString);
```

To perform XML parsing of an XML string that is name space qualified, specify "true" for the second argument for the XMLDocument constructor. The following is an example of parsing and XML string that contains name space qualification of its elements.

```
var xmlString = "<bk:book xmlns:bk='urn:loc.gov:books'
 xmlns:isbn='urn:ISBN:0-395-36341-6'>" +
                  "<bk:title>Cheaper by the Dozen</bk:title>" +
                  "<isbn:number>1568491379</isbn:number>" +
                "</bk:book>";

var xmldoc = new XMLDocument(xmlString, true); // XML document
 is name space aware
```

Locating nodes and elements

Now that we have the XMLDocument object, we can call the following APIs to locate nodes and elements of the XML document, as well as extract text from it directly. The query syntax for locating nodes and attributes is based on XPath.

Note:
XML parsing with this object is not namespace aware, this means that if you are locating a node name with namespace in it eg. "<names:myelement> ...", the XPath search string would be "//myelement"

The following are examples of locating a node by its XPath and getting the text value out of the resulting node.

```
var str = xmldoc.getNodeText("//two"); // returns the first
 occurrence of the node
// str == "abcd1234"

str = xmldoc.getNodeText("//three");
// str == "1234abcd"

str = xmldoc.getNodeText("/test/one/*");
// str == "abcd1234"

str = xmldoc.getNodeInt("//number");
// str == 1234
```

The following examples locates the node by XPath and uses the API on node and element to get attributes and value.

```
var node = xmldoc.getNode("/test/one/two");
// node.getNodeName() == "two"
// node.getNodeType() == "1" // 1 == ELEMENT_NODE
// node.getLastChild().getNodeType() == "3" // 3 == TEXT_NODE
// node.getLastChild().getNodeValue() == "abcd1234"
```

Or you can use the following convenience functions to get the node name and type

```
str = xmldoc.getNodeName("//three");
// str == "three"

str = xmldoc.getNodeType("//three");
// str == "1"
```

You can also get a list of nodes that you can iterate or access directly by position

```
var nodelist = xmldoc.getNodes("//one/*"); // two, three, and
 two
// nodelist.getLength() == "3"
// nodelist.item(0).getNodeName() == "two"
// nodelist.item(1).getNodeName() == "three"
```

The following is an example of parsing an XML string that is qualifed with name spaces.

```
var xmlString = "<bk:book xmlns:bk='urn:loc.gov:books'
 xmlns:isbn='urn:ISBN:0-395-36341-6'>" +
                 "<bk:title>Cheaper by the Dozen</bk:title>" +
                 "<isbn:number>1568491379</isbn:number>" +
               "</bk:book>";

var xmldoc = new XMLDocument(xmlString, true);
var str = xmldoc.getNodeText("//bk:title"); // returns the first
 occurence of the node
gs.log(str);

str = xmldoc.getNodeText("/bk:book/*");
gs.log(str);

str = xmldoc.getNodeInt("//isbn:number");
gs.log(str);
```

Getting attribute values

An attribute is just an extension of a node and so it has all the same APIs.

The following example shows how to query for a node to get its attribute by position

```
node = xmldoc.getNode("//two");
// node.getAttributes().item(0).getNodeValue() == "xxx"

str = xmldoc.getAttribute("//two", "att");
// str == "xxx"
```

XPath also has a query syntax for locating the attribute node directly, for example

```
str = xmldoc.getNodeText("//*[@att=\"yyy\"]");
// str == "1234abcd"

str = xmldoc.getNode("//@boo").getNodeValue();
```

```
// str == "yah"
```

Creating new elements

An XML element can be created at any level of the XML document once it has been created. Use the *setCurrent* function to position where the new element will be created as a child element, and use the *createElement* function to create the element.

```
var xmlString = "<test>" +
                "  <one>" +
                "    <two att=\"xxx\">abcd1234</two>" +
                "    <three boo=\"yah\"
 att=\"yyy\">1234abcd</three>" +
                "    <two>another</two>" +
                "  </one>" +
                "  <number>1234</number>" +
                "</test>";

var xmldoc = new XMLDocument(xmlString);
xmldoc.createElement("new", "test"); // creates the new element
 at the document element level if setCurrent is never called
xmldoc.createElement("new2"); // calling without a value will
 create a new element by itself

var el = xmldoc.createElement("new3");
xmldoc.setCurrent(el); // this is now the parent of any new
 elements created subsequently using createElement()
xmldoc.createElement("newChild", "test");
```

The resulting XML document looks like this

```
<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three boo="yah" att="yyy">1234abcd</three>
    <two>another</two>
  </one>
  <number>1234</number>
  <new>test<new>
  <new2/>
  <new3>
      <newChild>test</newChild>
  </new3>
</test>
```

XMLHelper

The XML helper script include makes it easy to parse XML in scripts.

Use the following methods to export XML to JSON, or JSON to XML.

- The `toObject()` method returns the XML elements as JSON properties. This method works properly whether the supplied parameter is an XML document or an XML string. This method has an optional parameter of the XML input for conversion as an alternative to specifying the XML input in the constructor.
- The `toXMLDoc()` method returns JSON provided as XML elements.

Note:
You must escape ampersand characters (&amp) in your XML or the conversion silently fails.

Example

The following example shows how to convert an XML document into JSON and uses a recursive function to output each member. The recursive function helps indicate how the XML document structure is rendered as JSON.

```
var xmlString = "<company>" + "<employee>" + "<id>10</id>"
 + "<firstname>Tom</firstname>" + "<lastname>Cruise</lastname>"
 + "<test>test1</test>"
 + "<test>test3</test>" + "</employee>" + "<employee>" +
 "<id>20</id>" + "<firstname>Paul</firstname>"
 + "<lastname>Enderson</lastname>" + "<test>test6</test>" +
 "<test>test5</test>" + "</employee>" + "<employee>"
 + "<id>30</id>" + "<firstname>Paul</firstname>" +
 "<lastname>Bush</lastname>" + "<test>test2</test>"
 + "<test>test4</test>" + "</employee>" + "</company>";
var helper = new XMLHelper(xmlString);
var obj = helper.toObject();
logObj(obj, "*");

function logObj(obj, sep) {
  for (x in obj) {
    if (typeof obj[x] != "function") {
      gs.log(sep + x + ":: " + obj[x]);
    }
    logObj(obj[x], sep + "*");
  }
}
```

Output:

```
*** Script: *employee:: [object Object],[object Object],[object
 Object]
*** Script: **2:: [object Object]
*** Script: ***id:: 30
*** Script: ***test:: test2,test4
*** Script: ****0:: test2
*** Script: ****1:: test4
*** Script: ***firstname:: Paul
*** Script: ***lastname:: Bush
*** Script: **0:: [object Object]
*** Script: ***id:: 10
*** Script: ***test:: test1,test3
*** Script: ****0:: test1
*** Script: ****1:: test3
*** Script: ***firstname:: Tom
*** Script: ***lastname:: Cruise
*** Script: **1:: [object Object]
*** Script: ***id:: 20
*** Script: ***test:: test6,test5
*** Script: ****0:: test6
*** Script: ****1:: test5
*** Script: ***firstname:: Paul
*** Script: ***lastname:: Enderson
```

JavaScript engine on the platform

The JavaScript engine that evaluates server-side scripts supports the ECMAScript 2021 (ES12) standard.

No plugins or properties are required to install the new JavaScript engine.

The benefits of the new engine are:

- Modern library code and scripting features, such as optional chaining, destructuring, const and let declarations, and arrow functions
- Follows standard ECMAScript 2021 behavior

What to know

The JavaScript engine provides an improved environment for developing scripts.

- Beginning with the Xanadu release, when you create new scripts, ECMAScript 2021 (ES12) mode is turned on by default regardless of the JavaScript mode configured for the application.
- Beginning with the Tokyo release, new and existing scoped applications can run in ECMAScript 2021 (ES12) mode.
- Compatibility mode supports the legacy modifications to the legacy JavaScript engine.
- Legacy code continues to work.

You configure the mode that the JavaScript engine uses in the design and runtime settings for applications. Available modes are ECMAScript 2021 (ES12), ES5 Standards, and Compatibility.

Related topics

Update a custom application record

JavaScript modes

JavaScript mode is a design and runtime setting for custom applications and scripts. To support existing server-side scripts and new scripts developed to the ECMAScript 2021 standard, the JavaScript engine has three modes: ECMAScript 2021 (ES12), ES5 Standards, and Compatibility.

The JavaScript mode controls which JavaScript features you have access to in an application or script. The default mode for new scoped applications is ECMAScript 2021 (ES12) and for new global applications, it's ES5 Standards. You can also turn on ECMAScript 2021 (ES12) mode for individual scripts in applications that use ES5 Standards or Compatibility mode.

For more information about features supported by the ECMAScript 2021 (ES12) and ES5 Standards modes, see JavaScript engine feature support.

ECMAScript 2021 (ES12) mode

ECMAScript 2021 (ES12) mode is the default mode when you create new scoped applications. When you create new scripts, ECMAScript 2021 (ES12) mode is turned on by default regardless of the JavaScript mode configured for the application. This mode doesn't preserve the legacy behaviors in the pre-Tokyo JavaScript engine or work with global scripts.

ECMAScript 2021 (ES12) mode supports a subset of ECMAScript 2021 (ES12) and ECMAScript 2022 (ES13) syntax and features, including the following features:

- Default function parameters
- Rest parameters
- For-of loops

- Template literals
- Destructuring
  - Declarations
  - Assignment
  - Parameters
- Const declaration
- Let declaration
- Arrow functions
- Class declarations
- Map set
- Optional chaining operator `(?.)`

To learn about specific ECMAScript 2021 (ES12) features, see the Let's Learn ECMAScript 2021 videos on the ServiceNow Dev Program YouTube channel.

ES5 Standards mode

ES5 Standards mode is the default mode for global applications and is an option for scoped applications. This mode doesn't preserve the legacy behaviors in the pre-Helsinki JavaScript engine.

ES5 standards mode supports ECMAScript5 syntax and features, including the following features:

- The "use strict" declaration
- Control over extensibility of objects
- Get and set properties on objects (accessors)
- Control over writability, configurability, and enumerability of object properties
- New Array and Date methods
- Native JSON support
- Support for modern third-party libraries such as lodash.js and moment.js

Compatibility mode

Compatibility mode is used for all scripts developed prior to the addition of ES5 Standards mode. Compatibility mode has some differences from the previous JavaScript engine.

JSON support changes:

- `JSON.stringify()` and `JSON.parse()` are implemented using the ES5 Native JSON object.
- The new `JSON().encode()` and new `JSON().decode()` are still supported, but should only be used when the legacy behavior is required.

The use of third-party JavaScript libraries isn't supported in Compatibility mode.

Considerations for switching JavaScript modes

Switching the JavaScript mode for an application or script might change the behavior of existing scripts. Review some examples of behavior changes before switching JavaScript modes or to troubleshoot any issues that you experience after switching.

For more information about each JavaScript mode, see JavaScript modes and JavaScript engine feature support.

This table highlights how JavaScript behavior has evolved from the lenient and error-prone pre-ES5 environment, to the stricter and more predictable ES5, and lastly the more feature-rich environment of ES12 (ECMAScript 2021).

Behavioral differences in JavaScript modes

Feature	Compatibility Mode	ES5 Standards Mode	ECMAScript 2021 (ES12)
Arguments object	The `arguments` object exists, but there's no `strict mode`, so modifications reflect on `arguments`. Prints: `*** Script: [object Arguments]` `*** Script: [object Arguments]` `*** Script: [object Arguments]` `*** Script: 123`	In strict mode, the arguments object doesn't reflect parameter modifications and throws an error. Prints: `sn_es5: 123` `sn_es5: undefined` `sn_es5: [object Arguments]` `sn_es5: 123`	The same as ES5.
Boolean overrides	Primitive Booleans (`true`, `false`) can be overridden, causing unexpected behavior.	Primitive Booleans are more protected, though still can be overridden when assigned to variables.	The same as ES5, but strict mode helps prevent some assignments. The conditional expression should be written in this form: `(cond_expr instanceof Boolean ? cond_expr.valueOf() : cond_expr).`
Exception for syntax errors	Syntax errors throw exceptions at runtime. Error handling is inconsistent. Example: `Javascript compiler`	More consistent syntax error handling, especially in strict mode. Example: `Evaluator: com.glide.scr`	The same as ES5, but with more robust handling and clearer error messages in updated engines. Example:

Behavioral differences in JavaScript modes (continued)

Feature	Compatibility Mode	ES5 Standards Mode	ECMAScript 2021 (ES12)
	`exception: unterminated string literal (null.null.script; line 1) in: var b = '`	`ipt.RhinoEcmaError: unterminated string literal script : Line(1) column(9) ==> 1: var b = '`	`SyntaxError: Unterminated string constant at line 1 ==> 1: var b = '`
Increment and decrement	Allowed on variables but could behave unexpectedly with complex expressions. Prints: `*** Script: c: 1 *** Script: gr.related_incidents: 1 *** Script: 2 *** Script: 3`	Improved clarity, but still allowed on variables (`var`, `let`, `const`). Prints: `sn_es5: c: 0 sn_es5: gr.related_incidents: 1 sn_es5: 1 sn_es5: 2`	The same as ES5, with stricter rules in some contexts (for example, `const`).
Line continuations	Allowed with a backslash (\) but discouraged due to readability issues. In this example, all three functions are called. `var expr = doFoo(); // do foo doBar(); // do bar finish(); // all done eval(expr);`	Same as Compatibility mode; no change in handling line continuations. In the previous example, ES5 only calls the first function and treats everything after the first comment including the newline as comment until the expression end.	The same as ES5, but template literals provide a more readable alternative.
Missing semicolons	Automatic semicolon insertion (ASI) often led to unexpected behavior.	Throws a syntax error when a semicolon is missing.	The same as ES5. Updated practices encourage explicit semicolons.
Non-existent functions	Calling a non-existent function throws a `ReferenceError`.	Throws a `TypeError` if a non-function is called.	Throws an EcmaError when a non-existent function is called

Behavioral differences in JavaScript modes (continued)

Feature	Compatibility Mode	ES5 Standards Mode	ECMAScript 2021 (ES12)
			or a property is referenced.
Non-existent properties	Accessing a non-existent property returns `undefined`; no error thrown.	Same as pre-ES5.	The same as Compatibility mode and ES5 Standards mode.
Numeric literals	Basic decimal and hexadecimal literals.	Introduced stricter parsing rules and better handling of numeric literals.	Added binary (`0b`), octal (`0o`), and BigInt literals (`123n`).
Reserved keyword as property	Using reserved keywords isn't possible.	Reserved keywords can be used as property names without error, for example, `obj.for`. Prints the object when returned.	The same as ES5.
Treat let and yield as keywords	`let` and `yield` aren't keywords and can be used as identifiers only.	`let` is introduced as a keyword. `yield` is reserved in strict mode.	Both are keywords. Using them as identifiers throws syntax errors.

Set the JavaScript mode for an application

Configure which ECMAScript features can be used in an application by selecting the JavaScript mode.

Before you begin

Role required: delegated developer role granting full access or admin

About this task

The JavaScript mode is a design and runtime setting for custom applications and scripts. To support existing server-side scripts and new scripts developed to the ECMAScript 2021 standard, the JavaScript engine has three modes: ECMAScript 2021 (ES12), ES5 Standards, and Compatibility. For more information about each mode, see JavaScript modes.

For applications that use ES5 Standards or Compatibility mode, you can also turn on ECMAScript 2021 (ES12) for individual scripts in the application. For more information, see Turn on ECMAScript 2021 (ES12) mode for a script.

Note:
Switching the JavaScript mode to ECMAScript 2021 (ES12) for an existing application might change the behavior of its scripts. For more information, see Considerations for switching JavaScript modes.

Procedure

1. In the application navigator, enter `sys_app.list`.
2. Select an application.
3. In the **JavaScript Mode** field, select the mode to use.

- ECMAScript 2021 (ES12): Supports a subset of ECMAScript 2021 (ES12) and ECMAScript 2022 (ES13) syntax and features.
- ES5 Standards Mode: Supports ECMAScript5 syntax and features.
- Compatibility Mode: Used for all scripts developed prior to the addition of ES5 Standards mode.

4. Select **Update**.

Related topics

Legacy - Update a custom application record

Turn on ECMAScript 2021 (ES12) mode for a script

Use the latest JavaScript features supported with ECMAScript 2021 (ES12) mode in server-side scripts in applications that use ES5 Standards mode or Compatibility mode.

Before you begin

Role required: delegated developer role or admin

About this task

Turning on ECMAScript 2021 (ES12) mode for individual scripts is an option for scripts in global or scoped applications configured to use ES5 Standards mode or Compatibility mode. All scripts in applications with the JavaScript mode set to ECMAScript 2021 (ES12) use ECMAScript 2021 (ES12). Switching the JavaScript mode to ECMAScript 2021 (ES12) for an existing script might change the behavior of the script. For more information, see Considerations for switching JavaScript modes.

Note:
Global applications can use ECMAScript 2021 (ES12) mode for individual scripts but not across the application.

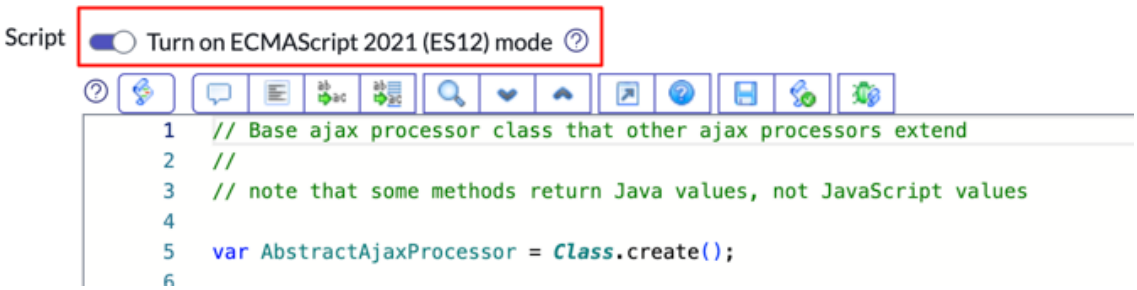
Procedure

1. Navigate to a form with a script field.

   Example
   For example, to open a script include form, navigate to **All** > **System Definition** > **Script Includes** or enter `sys_script_include.list` in the navigation filter.

2. Select an existing script or **New**.
3. Select **Turn on ECMAScript 2021 (ES12) mode**.

   Example



Note:
This option is read only for scripts in applications with the JavaScript mode set to ECMAScript 2021 (ES12), which automatically use ECMAScript 2021 (ES12).

4. Select **Submit** or **Update** to save your changes.

JavaScript engine feature support

Compare ECMAScript features between the ECMAScript 2021 (ES12) and ES5 Standards JavaScript modes in Xanadu. Both modes support a subset of ECMAScript features.

For more information about these features, see the ECMAScript language specifications (ECMA-262) on the Ecma International website.

Support definitions

Supported

The feature has been tested and validated.

Not Supported

The feature has not been validated in the current release.

Disallowed

The feature does not align with the ServiceNow AI Platform programming model or poses a security or performance risk. Disallowed features result in an error.

ECMAScript 2022 (ES13) features

Important:
Prior to deploying code to production, you should test scripts using supported ECMAScript 2022 (ES13) features thoroughly due to the newly added and partial support of features across this ECMAScript version.

Instance class fields

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
public instance class fields	Supported	Not Supported
private instance class fields basic support	Not Supported	Not Supported
private instance class fields initializers	Not Supported	Not Supported
optional private instance class fields access	Not Supported	Not Supported
optional deep private instance class fields access	Not Supported	Not Supported
computed instance class fields	Supported	Not Supported

Static class fields

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
public static class fields	Supported	Not Supported
static class fields use [[Define]]	Supported	Not Supported

Static class fields (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
private static class fields	Supported	Not Supported
computed static class fields	Supported	Not Supported

Private class methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
private instance methods	Not Supported	Not Supported
private static methods	Supported	Not Supported
private accessor properties	Not Supported	Not Supported
private static accessor properties	Supported	Not Supported

.at() method on the built-in indexables

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Array.prototype.at()	Not Supported	Not Supported
String.prototype.at()	Supported	Not Supported
%TypedArray%.prototype.at()	Disallowed	Disallowed

Object.hasOwn

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Basic functionality	Supported	Not Supported
ToObject called before ToPropertyKey	Supported	Not Supported

Error.cause property

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Error has cause	Supported	Not Supported
Error.prototype lacks cause	Supported	Not Supported
EvalError has cause	Supported	Not Supported
EvalError.prototype lacks cause	Supported	Not Supported
RangeError has cause	Supported	Not Supported
RangeError.prototype lacks cause	Supported	Not Supported
ReferenceError has cause	Supported	Not Supported
ReferenceError.prototype lacks cause	Supported	Not Supported
SyntaxError has cause	Supported	Not Supported
SyntaxError.prototype lacks cause	Supported	Not Supported
TypeError has cause	Supported	Not Supported

Error.cause property (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
TypeError.prototype lacks cause	Supported	Not Supported
URIError has cause	Supported	Not Supported
URIError.prototype lacks cause	Supported	Not Supported
AggregateError has cause	Supported	Not Supported
AggregateError.prototype lacks cause	Supported	Not Supported

RegExp Match Indices (`hasIndices` / `d` flag)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
constructor supports it	Not Supported	Not Supported
shows up in flags	Not Supported	Not Supported

Ergonomic brand checks for private fields

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Ergonomic brand checks for private fields	Not Supported	Not Supported

Class static initialization blocks

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Class static initialization blocks	Supported	Not Supported

ECMAScript 2021 (ES12) features

Promise.any

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
fulfillment	Disallowed	Disallowed
AggregateError	Disallowed	Disallowed

WeakReferences

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
WeakRef minimal support	Disallowed	Disallowed
FinalizationRegistry minimal support	Disallowed	Disallowed

Logical assignment

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
||= basic support	Supported	Not Supported

Logical assignment (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
||= short-circuiting behavior	Supported	Not Supported
||= setter not unnecessarily invoked	Supported	Not Supported
&&= basic support	Supported	Not Supported
&&= short-circuiting behavior	Supported	Not Supported
&&= setter not unnecessarily invoked	Supported	Not Supported
??= basic support	Supported	Not Supported
??= short-circuiting behavior	Supported	Not Supported
??= setter not unnecessarily invoked	Supported	Not Supported

Numeric separators

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
numeric separators	Supported	Not Supported

String.prototype.replaceAll

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
String.prototype.replaceAll	Supported	Supported

ECMAScript 2020 (ES11) features

String.prototype.matchAll

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Supported	Not Supported
throws on non-global regex	Supported	Not Supported

BigInt

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Not Supported	Not Supported
constructor	Not Supported	Not Supported
BigInt.asUintN	Not Supported	Not Supported
BigInt.asIntN	Not Supported	Not Supported
BigInt64Array	Not Supported	Not Supported
BigUint64Array	Not Supported	Not Supported
DataView.prototype.getBigInt64	Not Supported	Not Supported
DataView.prototype.getBigUint64	Not Supported	Not Supported

globalThis

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
"globalThis" global property is global object	Disallowed	Disallowed
"globalThis" global property has correct property descriptor	Disallowed	Disallowed

Optional chaining operator (?.)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
optional property access	Supported	Not Supported
optional bracket access	Supported	Not Supported
optional method call	Supported	Not Supported
optional function call	Supported	Not Supported
spread parameters after optional chaining	Supported	Not Supported

Promise.allSettled

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Promise.allSettled	Disallowed	Disallowed

Nullish coalescing operator (??)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
nullish coalescing operator (??)	Supported	Not Supported

ECMAScript 2019 (ES10) features

Symbol.prototype.description

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic	Supported	Not Supported
empty description	Supported	Not Supported
undefined description	Supported	Not Supported

String trimming

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
String.prototype.trimLeft	Supported	Supported
String.prototype.trimRight	Supported	Supported
String.prototype.trimStart	Supported	Not Supported
String.prototype.trimEnd	Supported	Not Supported

Array.prototype.{flat, flatMap}

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Array.prototype.flat	Supported	Not Supported
Array.prototype.flatMap	Supported	Not Supported
flat and flatMap in Array.prototype[@@unscopables]	Supported	Not Supported

Object.fromEntries

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Object.fromEntries	Supported	Not Supported

Optional catch binding

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic	Disallowed	Disallowed
await	Disallowed	Disallowed
yield	Disallowed	Disallowed

Function.prototype.toString revision

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
functions created with the Function constructor	Disallowed	Disallowed
arrows	Disallowed	Disallowed
[native code]	Disallowed	Disallowed
class expression with implicit constructor	Disallowed	Disallowed
class expression with explicit constructor	Disallowed	Disallowed
unicode escape sequences in identifiers	Disallowed	Disallowed
methods and computed property names	Disallowed	Disallowed

JSON superset

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Line separator can appear in string literals	Disallowed	Disallowed
Paragraph separator can appear in string literals	Disallowed	Disallowed

Well-formed JSON.stringify

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Well-formed JSON.stringify	Disallowed	Disallowed

ECMAScript 2018 (ES9) features

Object rest/spread properties

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
object rest properties	Supported	Not Supported
object spread properties	Supported	Not Supported

Promise.prototype.finally

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic support	Disallowed	Disallowed
don't change resolution value	Disallowed	Disallowed
change rejection value	Disallowed	Disallowed

Asynchronous iterators

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
async generators	Disallowed	Disallowed
for-await-of loops	Disallowed	Disallowed

s (dotAll) flag for regular expressions

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
s (dotAll) flag for regular expressions	Supported	Not Supported

RegExp named capture groups

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
RegExp named capture groups	Supported	Not Supported

RegExp Lookbehind Assertions

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
RegExp Lookbehind Assertions	Not Supported	Not Supported

RegExp Unicode Property Escapes

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
RegExp Unicode Property Escapes	Not Supported	Not Supported

Template literal revision

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
template literal revision	Disallowed	Disallowed

ECMAScript 2017 (ES8) features

Object static methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Object.values	Supported	Not Supported
Object.entries	Supported	Not Supported
Object.getOwnPropertyDescriptors	Supported	Not Supported
Object.getOwnPropertyDescriptors doesn't provide undefined descriptors	Not Supported	Not Supported

String padding

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
String.prototype.padStart	Supported	Not Supported
String.prototype.padEnd	Supported	Not Supported

Trailing commas in function syntax

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
in parameter lists	Supported	Not Supported
in argument lists	Supported	Not Supported

Async functions

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
return	Disallowed	Disallowed
throw	Disallowed	Disallowed
no line break between async and function	Disallowed	Disallowed
no "prototype" property	Disallowed	Disallowed

Async functions (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
await	Disallowed	Disallowed
await, rejection	Disallowed	Disallowed
must await a value	Disallowed	Disallowed
can await non-Promise values	Disallowed	Disallowed
cannot await in parameters	Disallowed	Disallowed
async methods, object literals	Disallowed	Disallowed
async methods, classes	Disallowed	Disallowed
async arrow functions in methods, classes	Disallowed	Disallowed
async arrow functions	Disallowed	Disallowed
correct prototype chain	Disallowed	Disallowed
async function prototype, Symbol.toStringTag	Disallowed	Disallowed
async function constructor	Disallowed	Disallowed

Shared memory and atomics

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
SharedArrayBuffer	Disallowed	Disallowed
SharedArrayBuffer[Symbol.species]	Disallowed	Disallowed
SharedArrayBuffer.prototype.byteLength	Disallowed	Disallowed
SharedArrayBuffer.prototype.slice	Disallowed	Disallowed
SharedArrayBuffer.prototype[Symbol.toStringTag]	Disallowed	Disallowed
Atomics.add	Disallowed	Disallowed
Atomics.and	Disallowed	Disallowed
Atomics.compareExchange	Disallowed	Disallowed
Atomics.exchange	Disallowed	Disallowed
Atomics.wait	Disallowed	Disallowed
Atomics.wake	Disallowed	Disallowed
Atomics.isLockFree	Disallowed	Disallowed
Atomics.load	Disallowed	Disallowed
Atomics.or	Disallowed	Disallowed
Atomics.store	Disallowed	Disallowed
Atomics.sub	Disallowed	Disallowed
Atomics.xor	Disallowed	Disallowed

Object.prototype getter/setter methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
__defineGetter__	Disallowed	Disallowed
__defineGetter__, symbols	Disallowed	Disallowed
__defineGetter__, ToObject(this)	Disallowed	Disallowed
__defineSetter__	Disallowed	Disallowed
__defineSetter__, symbols	Disallowed	Disallowed
__defineSetter__, ToObject(this)	Disallowed	Disallowed
__lookupGetter__	Disallowed	Disallowed
__lookupGetter__, prototype chain	Disallowed	Disallowed
__lookupGetter__, symbols	Disallowed	Disallowed
__lookupGetter__, ToObject(this)	Disallowed	Disallowed
__lookupGetter__, data properties can shadow accessors	Disallowed	Disallowed
__lookupSetter__	Disallowed	Disallowed
__lookupSetter__, prototype chain	Disallowed	Disallowed
__lookupSetter__, symbols	Disallowed	Disallowed
__lookupSetter__, ToObject(this)	Disallowed	Disallowed
__lookupSetter__, data properties can shadow accessors	Disallowed	Disallowed

Proxy internal calls, getter/setter methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
__defineGetter__	Disallowed	Disallowed
__defineSetter__	Disallowed	Disallowed
__lookupGetter__	Disallowed	Disallowed
__lookupSetter__	Disallowed	Disallowed

ECMAScript 2016 (ES7) features

Exponentiation (**) operator

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic support	Supported	Not Supported
assignment	Supported	Not Supported
early syntax error for unary negation without parentheses	Disallowed	Disallowed

Array.prototype.includes

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Array.prototype.includes	Supported	Not Supported
Array.prototype.includes is generic	Not Supported	Not Supported
%TypedArray%.prototype.includes	Disallowed	Disallowed

ECMAScript 2015 (ES6) features

Proper tail calls (tail call optimization)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
direct recursion	Disallowed	Disallowed
mutual recursion	Disallowed	Disallowed

Default function parameters

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Supported	Not Supported
explicit undefined defers to the default	Supported	Not Supported
defaults can refer to previous parameters	Supported	Not Supported
arguments object interaction	Supported	Not Supported
temporal dead zone	Disallowed	Disallowed
separate scope	Supported	Not Supported
new Function() support	Disallowed	Disallowed

Rest parameters

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Supported	Not Supported
function 'length' property	Supported	Not Supported
arguments object interaction	Not Supported	Not Supported
can't be used in setters	Disallowed	Disallowed
new Function() support	Disallowed	Disallowed

Spread syntax for iterable objects

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
with arrays, in function calls	Supported	Not Supported
with arrays, in array literals	Supported	Not Supported
with sparse arrays, in function calls	Supported	Not Supported

Spread syntax for iterable objects (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
with sparse arrays, in array literals	Supported	Not Supported
with strings, in function calls	Supported	Not Supported
with strings, in array literals	Supported	Not Supported
with astral plane strings, in function calls	Supported	Not Supported
with astral plane strings, in array literals	Supported	Not Supported
with generator instances, in calls	Disallowed	Disallowed
with generator instances, in arrays	Disallowed	Disallowed
with generic iterables, in calls	Supported	Not Supported
with generic iterables, in arrays	Supported	Not Supported
with instances of iterables, in calls	Supported	Not Supported
with instances of iterables, in arrays	Supported	Not Supported
spreading non-iterables is a runtime error	Supported	Not Supported

Object literal extensions

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
computed properties	Supported	Not Supported
shorthand properties	Supported	Not Supported
shorthand methods	Supported	Not Supported
string-keyed shorthand methods	Supported	Not Supported
computed shorthand methods	Supported	Not Supported
computed accessors	Supported	Not Supported

For-of loops

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
with arrays	Supported	Not Supported
with sparse arrays	Supported	Not Supported
with strings	Supported	Not Supported
with astral plane strings	Supported	Not Supported
with generator instances	Disallowed	Disallowed
with generic iterables	Supported	Not Supported
with instances of generic iterables	Supported	Not Supported
iterator closing, break	Supported	Not Supported
iterator closing, throw	Supported	Not Supported

Octal and binary literals

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
octal literals	Supported	Not Supported
binary literals	Supported	Not Supported
octal supported by Number()	Not Supported	Not Supported
binary supported by Number()	Not Supported	Not Supported

Template literals

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Supported	Not Supported
toString conversion	Supported	Not Supported
tagged template literals	Supported	Not Supported
passed array is frozen	Supported	Not Supported
line break normalization	Disallowed	Disallowed
TemplateStrings call site caching	Supported	Not Supported
TemplateStrings permanent caching	Supported	Not Supported

RegExp "y" and "u" flags

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
"y" flag	Supported	Not Supported
"y" flag, lastIndex	Supported	Not Supported
"u" flag	Not Supported	Not Supported
"u" flag, non-BMP Unicode characters	Not Supported	Not Supported
"u" flag, Unicode code point escapes	Not Supported	Not Supported
"u" flag, case folding	Not Supported	Not Supported

Destructuring, declarations

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
with arrays	Supported	Not Supported
with sparse arrays	Supported	Not Supported
with strings	Supported	Not Supported
with astral plane strings	Supported	Not Supported
with generator instances	Disallowed	Disallowed
with generic iterables	Supported	Not Supported
with instances of generic iterables	Supported	Not Supported
iterator closing	Supported	Not Supported

Destructuring, declarations (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
trailing commas in iterable patterns	Supported	Not Supported
with objects	Supported	Not Supported
object destructuring with primitives	Supported	Not Supported
trailing commas in object patterns	Supported	Not Supported
throws on null and undefined	Supported	Not Supported
computed properties	Supported	Not Supported
multiples in a single var statement	Supported	Not Supported
nested	Supported	Not Supported
in for-in loop heads	Supported	Not Supported
in for-of loop heads	Supported	Not Supported
in catch heads	Supported	Not Supported
rest	Supported	Not Supported
defaults	Supported	Not Supported
defaults, let temporal dead zone	Disallowed	Disallowed

Destructuring, assignment

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
with arrays	Supported	Not Supported
with sparse arrays	Supported	Not Supported
with strings	Supported	Not Supported
with astral plane strings	Supported	Not Supported
with generator instances	Disallowed	Disallowed
with generic iterables	Supported	Not Supported
with instances of generic iterables	Supported	Not Supported
iterator closing	Supported	Not Supported
iterable destructuring expression	Supported	Not Supported
chained iterable destructuring	Supported	Not Supported
trailing commas in iterable patterns	Supported	Not Supported
with objects	Supported	Not Supported
object destructuring with primitives	Supported	Not Supported
trailing commas in object patterns	Supported	Not Supported
object destructuring expression	Supported	Not Supported

Destructuring, assignment (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
parenthesized left-hand-side is a syntax error	Disallowed	Disallowed
chained object destructuring	Supported	Not Supported
throws on null and undefined	Supported	Not Supported
computed properties	Supported	Not Supported
nested	Supported	Not Supported
rest	Supported	Not Supported
nested rest	Supported	Not Supported
empty patterns	Supported	Not Supported
defaults	Supported	Not Supported

Destructuring, parameters

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
with arrays	Supported	Not Supported
with sparse arrays	Supported	Not Supported
with strings	Supported	Not Supported
with astral plane strings	Supported	Not Supported
with generator instances	Disallowed	Disallowed
with generic iterables	Supported	Not Supported
with instances of generic iterables	Supported	Not Supported
iterator closing	Supported	Not Supported
trailing commas in iterable patterns	Supported	Not Supported
with objects	Supported	Not Supported
object destructuring with primitives	Supported	Not Supported
trailing commas in object patterns	Supported	Not Supported
throws on null and undefined	Supported	Not Supported
computed properties	Supported	Not Supported
nested	Supported	Not Supported
'arguments' interaction	Supported	Not Supported
new Function() support	Disallowed	Disallowed
in parameters, function 'length' property	Supported	Not Supported
rest	Supported	Not Supported
empty patterns	Supported	Not Supported

Destructuring, parameters (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
defaults	Supported	Not Supported
defaults, separate scope	Supported	Not Supported
defaults, new Function() support	Disallowed	Disallowed
aliased defaults, arrow function	Supported	Not Supported
shorthand defaults, arrow function	Supported	Not Supported
duplicate identifier	Disallowed	Disallowed

Unicode code point escapes

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
in strings	Supported	Not Supported
in identifiers	Not Supported	Not Supported
in property key definitions	Not Supported	Not Supported
in property key accesses	Not Supported	Not Supported

New.target

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
in constructors	Not Supported	Not Supported
assignment is an early error	Disallowed	Disallowed

Const

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic support	Supported	Supported
is block-scoped	Supported	Not Supported
scope shadow resolution	Supported	Not Supported
cannot be in statements	Disallowed	Disallowed
redefining a const is an error	Disallowed	Disallowed
for loop statement scope	Supported	Not Supported
for-in loop iteration scope	Supported	Not Supported
for-of loop iteration scope	Supported	Not Supported
temporal dead zone	Not Supported	Not Supported
basic support (strict mode)	Supported	Supported
is block-scoped (strict mode)	Supported	Not Supported
scope shadow resolution (strict mode)	Supported	Not Supported
cannot be in statements (strict mode)	Disallowed	Disallowed

Const (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
redefining a const (strict mode)	Disallowed	Disallowed
for loop statement scope (strict mode)	Supported	Not Supported
for-in loop iteration scope (strict mode)	Supported	Not Supported
for-of loop iteration scope (strict mode)	Supported	Not Supported
temporal dead zone (strict mode)	Not Supported	Not Supported

Let

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic support	Supported	Not Supported
is block-scoped	Supported	Not Supported
scope shadow resolution	Supported	Not Supported
cannot be in statements	Disallowed	Disallowed
for loop statement scope	Supported	Not Supported
temporal dead zone	Not Supported	Not Supported
for/for-in loop iteration scope	Supported	Not Supported
for-in loop binding shadowing parameter	Disallowed	Disallowed
basic support (strict mode)	Supported	Not Supported
is block-scoped (strict mode)	Supported	Not Supported
scope shadow resolution (strict mode)	Supported	Not Supported
cannot be in statements (strict mode)	Disallowed	Disallowed
for loop statement scope (strict mode)	Supported	Not Supported
temporal dead zone (strict mode)	Not Supported	Not Supported
for/for-in loop iteration scope (strict mode)	Supported	Not Supported
for-in loop binding shadowing parameter (strict mode)	Disallowed	Disallowed

Block-level function declaration

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
block-level function declaration	Supported	Not Supported

Arrow functions

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
0 parameters	Supported	Not Supported

Arrow functions (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
1 parameter, no brackets	Supported	Not Supported
multiple parameters	Supported	Not Supported
lexical "this" binding	Supported	Not Supported
"this" unchanged by call or apply	Supported	Not Supported
can't be bound, can be curried	Supported	Not Supported
lexical "arguments" binding	Supported	Not Supported
no line break between parameters and `=>`	Disallowed	Disallowed
correct precedence	Disallowed	Disallowed
no "prototype" property	Not Supported	Not Supported
lexical "super" binding in constructors	Supported	Not Supported
lexical "super" binding in methods	Supported	Not Supported
lexical "new.target" binding	Not Supported	Not Supported

Class

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
class statement	Supported	Not Supported
is block-scoped	Supported	Not Supported
class expression	Supported	Not Supported
anonymous class	Supported	Not Supported
constructor	Supported	Not Supported
prototype methods	Supported	Not Supported
string-keyed methods	Supported	Not Supported
computed prototype methods	Supported	Not Supported
optional semicolons	Supported	Not Supported
static methods	Supported	Not Supported
computed static methods	Supported	Not Supported
accessor properties	Supported	Not Supported
computed accessor properties	Supported	Not Supported
static accessor properties	Supported	Not Supported
computed static accessor properties	Supported	Not Supported
class name is lexically scoped	Supported	Not Supported
computed names, temporal dead zone	Not Supported	Not Supported
methods aren't enumerable	Supported	Not Supported
implicit strict mode	Not Supported	Not Supported

Class (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
constructor requires new	Supported	Not Supported
extends	Supported	Not Supported
extends expressions	Supported	Not Supported
extends null	Supported	Not Supported
new.target	Supported	Not Supported

Super

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
statement in constructors	Supported	Not Supported
expression in constructors	Supported	Not Supported
in methods, property access	Supported	Not Supported
in methods, method calls	Supported	Not Supported
method calls use correct "this" binding	Supported	Not Supported
constructor calls use correct "new.target" binding	Supported	Not Supported
is statically bound	Supported	Not Supported
super() invokes the correct constructor	Supported	Not Supported

Generators

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Disallowed	Disallowed
generator function expressions	Disallowed	Disallowed
correct "this" binding	Disallowed	Disallowed
can't use "this" with new	Disallowed	Disallowed
sending	Disallowed	Disallowed
%GeneratorPrototype%	Disallowed	Disallowed
%GeneratorPrototype% prototype chain	Disallowed	Disallowed
%GeneratorPrototype%.constructor	Disallowed	Disallowed
%GeneratorPrototype%.throw	Disallowed	Disallowed
%GeneratorPrototype%.return	Disallowed	Disallowed
yield operator precedence	Disallowed	Disallowed
yield *, arrays	Disallowed	Disallowed
yield *, sparse arrays	Disallowed	Disallowed

Generators (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
yield *, strings	Disallowed	Disallowed
yield *, astral plane strings	Disallowed	Disallowed
yield *, generator instances	Disallowed	Disallowed
yield *, generic iterables	Disallowed	Disallowed
yield *, instances of iterables	Disallowed	Disallowed
yield * on non-iterables is a runtime error	Disallowed	Disallowed
yield *, iterator closing	Disallowed	Disallowed
yield *, iterator closing via throw()	Disallowed	Disallowed
shorthand generator methods	Disallowed	Disallowed
string-keyed shorthand generator methods	Disallowed	Disallowed
computed shorthand generators	Disallowed	Disallowed
shorthand generator methods, classes	Disallowed	Disallowed
computed shorthand generators, classes	Disallowed	Disallowed
shorthand generators can't be constructors	Disallowed	Disallowed

Typed arrays

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Int8Array	Disallowed	Disallowed
Uint8Array	Disallowed	Disallowed
Uint8ClampedArray	Disallowed	Disallowed
Int16Array	Disallowed	Disallowed
Uint16Array	Disallowed	Disallowed
Int32Array	Disallowed	Disallowed
Uint32Array	Disallowed	Disallowed
Float32Array	Disallowed	Disallowed
Float64Array	Disallowed	Disallowed
DataView (Int8)	Disallowed	Disallowed
DataView (Uint8)	Disallowed	Disallowed
DataView (Int16)	Disallowed	Disallowed
DataView (Uint16)	Disallowed	Disallowed
DataView (Int32)	Disallowed	Disallowed
DataView (Uint32)	Disallowed	Disallowed
DataView (Float32)	Disallowed	Disallowed
DataView (Float64)	Disallowed	Disallowed

Typed arrays (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
ArrayBuffer[Symbol.species]	Disallowed	Disallowed
constructors require new	Disallowed	Disallowed
constructors accept generic iterables	Disallowed	Disallowed
correct prototype chains	Disallowed	Disallowed
%TypedArray%.from	Disallowed	Disallowed
%TypedArray%.of	Disallowed	Disallowed
%TypedArray%.prototype.subarray	Disallowed	Disallowed
%TypedArray%.prototype.join	Disallowed	Disallowed
%TypedArray%.prototype.indexOf	Disallowed	Disallowed
%TypedArray%.prototype.lastIndexOf	Disallowed	Disallowed
%TypedArray%.prototype.slice	Disallowed	Disallowed
%TypedArray%.prototype.every	Disallowed	Disallowed
%TypedArray%.prototype.filter	Disallowed	Disallowed
%TypedArray%.prototype.forEach	Disallowed	Disallowed
%TypedArray%.prototype.map	Disallowed	Disallowed
%TypedArray%.prototype.reduce	Disallowed	Disallowed
%TypedArray%.prototype.reduceRight	Disallowed	Disallowed
%TypedArray%.prototype.reverse	Disallowed	Disallowed
%TypedArray%.prototype.some	Disallowed	Disallowed
%TypedArray%.prototype.sort	Disallowed	Disallowed
%TypedArray%.prototype.copyWithin	Disallowed	Disallowed
%TypedArray%.prototype.find	Disallowed	Disallowed
%TypedArray%.prototype.findIndex	Disallowed	Disallowed
%TypedArray%.prototype.fill	Disallowed	Disallowed
%TypedArray%.prototype.keys	Disallowed	Disallowed
%TypedArray%.prototype.values	Disallowed	Disallowed
%TypedArray%.prototype.entries	Disallowed	Disallowed
%TypedArray%.prototype[Symbol.iterator]	Disallowed	Disallowed
%TypedArray%[Symbol.species]	Disallowed	Disallowed

Map

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Supported	Not Supported
constructor arguments	Supported	Not Supported

Map (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
constructor requires new	Supported	Not Supported
constructor accepts null	Supported	Not Supported
constructor invokes set	Supported	Not Supported
iterator closing	Supported	Not Supported
Map.prototype.set returns this	Supported	Not Supported
-0 key converts to +0	Supported	Not Supported
Map.prototype.size	Supported	Not Supported
Map.prototype.delete	Supported	Not Supported
Map.prototype.clear	Supported	Not Supported
Map.prototype.forEach	Supported	Not Supported
Map.prototype.keys	Supported	Not Supported
Map.prototype.values	Supported	Not Supported
Map.prototype.entries	Supported	Not Supported
Map.prototype[Symbol.iterator]	Supported	Not Supported
Map.prototype isn't an instance	Supported	Not Supported
Map iterator prototype chain	Supported	Not Supported
Map[Symbol.species]	Supported	Not Supported

Set

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Supported	Not Supported
constructor arguments	Supported	Not Supported
constructor requires new	Supported	Not Supported
constructor accepts null	Supported	Not Supported
constructor invokes add	Supported	Not Supported
iterator closing	Supported	Not Supported
Set.prototype.add returns this	Supported	Not Supported
-0 key converts to +0	Supported	Not Supported
Set.prototype.size	Supported	Not Supported
Set.prototype.delete	Supported	Not Supported
Set.prototype.clear	Supported	Not Supported
Set.prototype.forEach	Supported	Not Supported
Set.prototype.keys	Supported	Not Supported
Set.prototype.values	Supported	Not Supported

Set (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Set.prototype.entries	Supported	Not Supported
Set.prototype[Symbol.iterator]	Supported	Not Supported
Set.prototype isn't an instance	Supported	Not Supported
Set iterator prototype chain	Supported	Not Supported
Set[Symbol.species]	Supported	Not Supported

WeakMap

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Disallowed	Disallowed
constructor arguments	Disallowed	Disallowed
constructor requires new	Disallowed	Disallowed
constructor accepts null	Disallowed	Disallowed
constructor invokes set	Disallowed	Disallowed
frozen objects as keys	Disallowed	Disallowed
iterator closing	Disallowed	Disallowed
WeakMap.prototype.set returns this	Disallowed	Disallowed
WeakMap.prototype.delete	Disallowed	Disallowed
no WeakMap.prototype.clear method	Disallowed	Disallowed
.has, .get and .delete methods accept primitives	Disallowed	Disallowed
WeakMap.prototype isn't an instance	Disallowed	Disallowed

WeakSet

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Disallowed	Disallowed
constructor arguments	Disallowed	Disallowed
constructor requires new	Disallowed	Disallowed
constructor accepts null	Disallowed	Disallowed
constructor invokes add	Disallowed	Disallowed
iterator closing	Disallowed	Disallowed
WeakSet.prototype.add returns this	Disallowed	Disallowed
WeakSet.prototype.delete	Disallowed	Disallowed
no WeakSet.prototype.clear method	Disallowed	Disallowed

WeakSet (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
.has and .delete methods accept primitives	Disallowed	Disallowed
WeakSet.prototype isn't an instance	Disallowed	Disallowed

Proxy

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
constructor requires new	Disallowed	Disallowed
no "prototype" property	Disallowed	Disallowed
"get" handler	Disallowed	Disallowed
"get" handler, instances of proxies	Disallowed	Disallowed
"get" handler invariants	Disallowed	Disallowed
"set" handler	Disallowed	Disallowed
"set" handler, instances of proxies	Disallowed	Disallowed
"set" handler invariants	Disallowed	Disallowed
"has" handler	Disallowed	Disallowed
"has" handler, instances of proxies	Disallowed	Disallowed
"has" handler invariants	Disallowed	Disallowed
"deleteProperty" handler	Disallowed	Disallowed
"deleteProperty" handler invariant	Disallowed	Disallowed
"getOwnPropertyDescriptor" handler	Disallowed	Disallowed
"getOwnPropertyDescriptor" handler invariants	Disallowed	Disallowed
"defineProperty" handler	Disallowed	Disallowed
"defineProperty" handler invariants	Disallowed	Disallowed
"getPrototypeOf" handler	Disallowed	Disallowed
"getPrototypeOf" handler invariant	Disallowed	Disallowed
"setPrototypeOf" handler	Disallowed	Disallowed
"setPrototypeOf" handler invariant	Disallowed	Disallowed
"isExtensible" handler	Disallowed	Disallowed
"isExtensible" handler invariant	Disallowed	Disallowed
"preventExtensions" handler	Disallowed	Disallowed
"preventExtensions" handler invariant	Disallowed	Disallowed
"ownKeys" handler	Disallowed	Disallowed
"ownKeys" handler invariant	Disallowed	Disallowed

Proxy (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
"apply" handler	Disallowed	Disallowed
"apply" handler invariant	Disallowed	Disallowed
"construct" handler	Disallowed	Disallowed
"construct" handler invariants	Disallowed	Disallowed
Proxy.revocable	Disallowed	Disallowed
Array.isArray support	Disallowed	Disallowed
JSON.stringify support	Disallowed	Disallowed

Reflect

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Reflect.get	Disallowed	Disallowed
Reflect.set	Disallowed	Disallowed
Reflect.has	Disallowed	Disallowed
Reflect.deleteProperty	Disallowed	Disallowed
Reflect.getOwnPropertyDescriptor	Disallowed	Disallowed
Reflect.defineProperty	Disallowed	Disallowed
Reflect.getPrototypeOf	Disallowed	Disallowed
Reflect.setPrototypeOf	Disallowed	Disallowed
Reflect.isExtensible	Disallowed	Disallowed
Reflect.preventExtensions	Disallowed	Disallowed
Reflect.ownKeys, string keys	Disallowed	Disallowed
Reflect.ownKeys, symbol keys	Disallowed	Disallowed
Reflect.apply	Disallowed	Disallowed
Reflect.construct	Disallowed	Disallowed
Reflect.construct sets new.target meta-property	Disallowed	Disallowed
Reflect.construct creates instances from third argument	Disallowed	Disallowed
Reflect.construct, Array subclassing	Disallowed	Disallowed
Reflect.construct, RegExp subclassing	Disallowed	Disallowed
Reflect.construct, Function subclassing	Disallowed	Disallowed
Reflect.construct, Promise subclassing	Disallowed	Disallowed

Promise

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Disallowed	Disallowed
constructor requires new	Disallowed	Disallowed
Promise.prototype isn't an instance	Disallowed	Disallowed
Promise.all	Disallowed	Disallowed
Promise.all, generic iterables	Disallowed	Disallowed
Promise.race	Disallowed	Disallowed
Promise.race, generic iterables	Disallowed	Disallowed
Promise[Symbol.species]	Disallowed	Disallowed

Symbol

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Supported	Not Supported
typeof support	Supported	Not Supported
symbol keys are hidden to pre-ES6 code	Supported	Not Supported
Object.defineProperty support	Supported	Not Supported
symbols inherit from Symbol.prototype	Supported	Not Supported
cannot coerce to string or number	Not Supported	Not Supported
can convert with String()	Not Supported	Not Supported
new Symbol() throws	Supported	Not Supported
Object(symbol)	Not Supported	Not Supported
JSON.stringify ignores symbol primitives	Supported	Not Supported
JSON.stringify ignores symbol objects	Not Supported	Not Supported
global symbol registry	Supported	Not Supported

Well-known symbols

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Symbol.hasInstance	Disallowed	Disallowed
Symbol.isConcatSpreadable	Disallowed	Disallowed
Symbol.iterator, existence	Disallowed	Disallowed
Symbol.iterator, arguments object	Disallowed	Disallowed
Symbol.species, existence	Disallowed	Disallowed
Symbol.species, Array.prototype.concat	Disallowed	Disallowed
Symbol.species, Array.prototype.filter	Disallowed	Disallowed

Well-known symbols (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Symbol.species, Array.prototype.map	Disallowed	Disallowed
Symbol.species, Array.prototype.slice	Disallowed	Disallowed
Symbol.species, Array.prototype.splice	Disallowed	Disallowed
Symbol.species, RegExp.prototype[Symbol.split]	Disallowed	Disallowed
Symbol.species, Promise.prototype.then	Disallowed	Disallowed
Symbol.replace	Disallowed	Disallowed
Symbol.search	Disallowed	Disallowed
Symbol.split	Disallowed	Disallowed
Symbol.match	Disallowed	Disallowed
Symbol.match, RegExp constructor	Disallowed	Disallowed
Symbol.match, String.prototype.startsWith	Disallowed	Disallowed
Symbol.match, String.prototype.endsWith	Disallowed	Disallowed
Symbol.match, String.prototype.includes	Disallowed	Disallowed
Symbol.toPrimitive	Disallowed	Disallowed
Symbol.toStringTag	Disallowed	Disallowed
Symbol.toStringTag affects existing built-ins	Disallowed	Disallowed
Symbol.toStringTag, new built-ins	Disallowed	Disallowed
Symbol.toStringTag, misc. built-ins	Disallowed	Disallowed
Symbol.unscopables	Disallowed	Disallowed

Object static methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Object.assign	Supported	Not Supported
Object.is	Supported	Not Supported
Object.getOwnPropertySymbols	Supported	Not Supported
Object.setPrototypeOf	Not Supported	Not Supported

Function "name" property

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
function statements	Supported	Supported
function expressions	Supported	Supported
new Function	Not Supported	Not Supported

Function "name" property (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
bound functions	Not Supported	Not Supported
variables (function)	Supported	Not Supported
object methods (function)	Supported	Not Supported
accessor properties	Not Supported	Not Supported
shorthand methods	Supported	Not Supported
shorthand methods (no lexical binding)	Supported	Not Supported
symbol-keyed methods	Not Supported	Not Supported
class statements	Supported	Not Supported
class expressions	Supported	Not Supported
variables (class)	Supported	Not Supported
object methods (class)	Not Supported	Not Supported
class prototype methods	Supported	Not Supported
class static methods	Supported	Not Supported
isn't writable, is configurable	Not Supported	Not Supported

String static methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
String.raw	Supported	Not Supported
String.fromCodePoint	Supported	Not Supported

String.prototype methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
String.prototype.codePointAt	Supported	Supported
String.prototype.normalize	Supported	Supported
String.prototype.repeat	Supported	Supported
String.prototype.startsWith	Supported	Supported
String.prototype.startsWith throws on RegExp	Not Supported	Not Supported
String.prototype.endsWith	Supported	Supported
String.prototype.endsWith throws on RegExp	Not Supported	Not Supported
String.prototype.includes	Supported	Supported
String.prototype[Symbol.iterator]	Supported	Not Supported
String iterator prototype chain	Supported	Not Supported

RegExp.prototype properties

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
RegExp.prototype.flags	Supported	Not Supported
RegExp.prototype[Symbol.match]	Not Supported	Not Supported
RegExp.prototype[Symbol.replace]	Supported	Not Supported
RegExp.prototype[Symbol.split]	Supported	Not Supported
RegExp.prototype[Symbol.search]	Not Supported	Not Supported
RegExp[Symbol.species]	Supported	Not Supported

Array static methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Array.from, array-like objects	Supported	Not Supported
Array.from, generator instances	Disallowed	Disallowed
Array.from, generic iterables	Supported	Not Supported
Array.from, instances of generic iterables	Supported	Not Supported
Array.from map function, array-like objects	Supported	Not Supported
Array.from map function, generator instances	Disallowed	Disallowed
Array.from map function, generic iterables	Supported	Not Supported
Array.from map function, instances of iterables	Supported	Not Supported
Array.from, iterator closing	Supported	Not Supported
Array.of	Supported	Not Supported
Array[Symbol.species]	Supported	Not Supported

Array.prototype methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Array.prototype.copyWithin	Supported	Not Supported
Array.prototype.find	Supported	Not Supported
Array.prototype.findIndex	Supported	Not Supported
Array.prototype.fill	Supported	Not Supported
Array.prototype.keys	Supported	Not Supported
Array.prototype.values	Supported	Not Supported
Array.prototype.entries	Supported	Not Supported
Array.prototype[Symbol.iterator]	Supported	Not Supported
Array iterator prototype chain	Supported	Not Supported

Array.prototype methods (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Array.prototype[Symbol.unscopables]	Supported	Not Supported

Number properties

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Number.isFinite	Supported	Not Supported
Number.isInteger	Supported	Not Supported
Number.isSafeInteger	Supported	Not Supported
Number.isNaN	Supported	Not Supported
Number.parseFloat	Disallowed	Disallowed
Number.parseInt	Disallowed	Disallowed
Number.EPSILON	Supported	Not Supported
Number.MIN_SAFE_INTEGER	Supported	Not Supported
Number.MAX_SAFE_INTEGER	Supported	Not Supported

Math methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Math.clz32	Supported	Not Supported
Math.imul	Supported	Not Supported
Math.sign	Supported	Not Supported
Math.log10	Supported	Not Supported
Math.log2	Supported	Not Supported
Math.log1p	Supported	Not Supported
Math.expm1	Supported	Not Supported
Math.cosh	Supported	Not Supported
Math.sinh	Supported	Not Supported
Math.tanh	Supported	Not Supported
Math.acosh	Supported	Not Supported
Math.asinh	Supported	Not Supported
Math.atanh	Supported	Not Supported
Math.trunc	Supported	Not Supported
Math.fround	Supported	Not Supported
Math.cbrt	Supported	Not Supported
Math.hypot	Supported	Not Supported

Date.prototype[Symbol.toPrimitive]

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Date.prototype[Symbol.toPrimitive]	Supported	Not Supported

Array is subclassable

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
length property (accessing)	Disallowed	Disallowed
length property (setting)	Disallowed	Disallowed
correct prototype chain	Disallowed	Disallowed
Array.isArray support	Disallowed	Disallowed
Array.prototype.concat	Disallowed	Disallowed
Array.prototype.filter	Disallowed	Disallowed
Array.prototype.map	Disallowed	Disallowed
Array.prototype.slice	Disallowed	Disallowed
Array.prototype.splice	Disallowed	Disallowed
Array.from	Disallowed	Disallowed
Array.of	Disallowed	Disallowed

RegExp is subclassable

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Disallowed	Disallowed
correct prototype chain	Disallowed	Disallowed
RegExp.prototype.exec	Disallowed	Disallowed
RegExp.prototype.test	Disallowed	Disallowed

Function is subclassable

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
can be called	Disallowed	Disallowed
correct prototype chain	Disallowed	Disallowed
can be used with "new"	Disallowed	Disallowed
Function.prototype.call	Disallowed	Disallowed
Function.prototype.apply	Disallowed	Disallowed
Function.prototype.bind	Disallowed	Disallowed

Promise is subclassable

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Disallowed	Disallowed
correct prototype chain	Disallowed	Disallowed
Promise.all	Disallowed	Disallowed
Promise.race	Disallowed	Disallowed

Miscellaneous subclassables

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Boolean is subclassable	Disallowed	Disallowed
Number is subclassable	Disallowed	Disallowed
String is subclassable	Disallowed	Disallowed
Error is subclassable	Disallowed	Disallowed
Map is subclassable	Disallowed	Disallowed
Set is subclassable	Disallowed	Disallowed

Prototype of bound functions

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functions	Disallowed	Disallowed
generator functions	Disallowed	Disallowed
arrow functions	Disallowed	Disallowed
classes	Disallowed	Disallowed
subclasses	Disallowed	Disallowed

Proxy, internal 'get' calls

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
ToPrimitive	Disallowed	Disallowed
CreateListFromArrayLike	Disallowed	Disallowed
instanceof operator	Supported	Disallowed
HasBinding	Disallowed	Disallowed
CreateDynamicFunction	Disallowed	Disallowed
ClassDefinitionEvaluation	Disallowed	Disallowed
IteratorComplete, IteratorValue	Disallowed	Disallowed
ToPropertyDescriptor	Disallowed	Disallowed
Object.assign	Disallowed	Disallowed
Object.defineProperties	Disallowed	Disallowed

Proxy, internal 'get' calls (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Function.prototype.bind	Disallowed	Disallowed
Error.prototype.toString	Disallowed	Disallowed
String.raw	Disallowed	Disallowed
RegExp constructor	Disallowed	Disallowed
RegExp.prototype.flags	Disallowed	Disallowed
RegExp.prototype.test	Disallowed	Disallowed
RegExp.prototype.toString	Disallowed	Disallowed
RegExp.prototype[Symbol.match]	Disallowed	Disallowed
RegExp.prototype[Symbol.replace]	Disallowed	Disallowed
RegExp.prototype[Symbol.search]	Disallowed	Disallowed
RegExp.prototype[Symbol.split]	Disallowed	Disallowed
Array.from	Disallowed	Disallowed
Array.prototype.concat	Disallowed	Disallowed
Array.prototype iteration methods	Disallowed	Disallowed
Array.prototype.pop	Disallowed	Disallowed
Array.prototype.reverse	Disallowed	Disallowed
Array.prototype.shift	Disallowed	Disallowed
Array.prototype.splice	Disallowed	Disallowed
Array.prototype.toString	Disallowed	Disallowed
JSON.stringify	Disallowed	Disallowed
Promise resolve functions	Disallowed	Disallowed
String.prototype.match	Disallowed	Disallowed
String.prototype.replace	Disallowed	Disallowed
String.prototype.search	Disallowed	Disallowed
String.prototype.split	Disallowed	Disallowed
Date.prototype.toJSON	Disallowed	Disallowed

Proxy, internal 'set' calls

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Object.assign	Disallowed	Disallowed
Array.from	Disallowed	Disallowed
Array.of	Disallowed	Disallowed
Array.prototype.copyWithin	Disallowed	Disallowed
Array.prototype.fill	Disallowed	Disallowed

Proxy, internal 'set' calls (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Array.prototype.pop	Disallowed	Disallowed
Array.prototype.push	Disallowed	Disallowed
Array.prototype.reverse	Disallowed	Disallowed
Array.prototype.shift	Disallowed	Disallowed
Array.prototype.splice	Disallowed	Disallowed
Array.prototype.unshift	Disallowed	Disallowed

Proxy, internal 'defineProperty' calls

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
[[Set]]	Disallowed	Disallowed
SetIntegrityLevel	Disallowed	Disallowed

Proxy, internal 'deleteProperty' calls

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Array.prototype.copyWithin	Disallowed	Disallowed
Array.prototype.pop	Disallowed	Disallowed
Array.prototype.reverse	Disallowed	Disallowed
Array.prototype.shift	Disallowed	Disallowed
Array.prototype.splice	Disallowed	Disallowed
Array.prototype.unshift	Disallowed	Disallowed

Proxy, internal 'getOwnPropertyDescriptor' calls

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
[[Set]]	Disallowed	Disallowed
Object.assign	Disallowed	Disallowed
Object.prototype.hasOwnProperty	Disallowed	Disallowed
Function.prototype.bind	Disallowed	Disallowed

Proxy, internal 'ownKeys' calls

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
SetIntegrityLevel	Disallowed	Disallowed
TestIntegrityLevel	Disallowed	Disallowed
SerializeJSONObject	Disallowed	Disallowed

Object static methods accept primitives

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Object.getPrototypeOf	Disallowed	Disallowed
Object.getOwnPropertyDescriptor	Disallowed	Disallowed
Object.getOwnPropertyNames	Disallowed	Disallowed
Object.seal	Disallowed	Disallowed
Object.freeze	Disallowed	Disallowed
Object.preventExtensions	Disallowed	Disallowed
Object.isSealed	Disallowed	Disallowed
Object.isFrozen	Disallowed	Disallowed
Object.isExtensible	Disallowed	Disallowed
Object.keys	Disallowed	Disallowed

Own property order

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Object.keys	Disallowed	Disallowed
Object.getOwnPropertyNames	Disallowed	Disallowed
Object.assign	Disallowed	Disallowed
JSON.stringify	Disallowed	Disallowed
JSON.parse	Disallowed	Disallowed
Reflect.ownKeys, string key order	Disallowed	Disallowed
Reflect.ownKeys, symbol key order	Disallowed	Disallowed

Updated identifier syntax

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
var â¸¯;	Disallowed	Disallowed
var ð ‹€;	Disallowed	Disallowed
no escaped reserved words as identifiers	Disallowed	Disallowed

Non-strict function semantics

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
hoisted block-level function declaration	Disallowed	Disallowed
labeled function statements	Disallowed	Disallowed
function statements in if-statement clauses	Disallowed	Disallowed

__proto__ in object literals

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic support	Disallowed	Disallowed
multiple __proto__ is an error	Disallowed	Disallowed
not a computed property	Disallowed	Disallowed
not a shorthand property	Disallowed	Disallowed
not a shorthand method	Disallowed	Disallowed

Object.prototype.__proto__

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
get prototype	Disallowed	Disallowed
set prototype	Disallowed	Disallowed
absent from Object.create(null)	Disallowed	Disallowed
present in hasOwnProperty()	Disallowed	Disallowed
correct property descriptor	Disallowed	Disallowed
present in Object.getOwnPropertyNames()	Disallowed	Disallowed

String.prototype HTML methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
existence	Disallowed	Disallowed
tags' names are lowercase	Disallowed	Disallowed
quotes in arguments are escaped	Disallowed	Disallowed

RegExp.prototype.compile

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
basic functionality	Disallowed	Disallowed
returns this	Disallowed	Disallowed

RegExp syntax extensions

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
hyphens in character sets	Disallowed	Disallowed
invalid character escapes	Disallowed	Disallowed
invalid control-character escapes	Disallowed	Disallowed
invalid Unicode escapes	Disallowed	Disallowed
invalid hexadecimal escapes	Disallowed	Disallowed

RegExp syntax extensions (continued)

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
incomplete patterns and quantifiers	Disallowed	Disallowed
octal escape sequences	Disallowed	Disallowed
invalid backreferences become octal escapes	Disallowed	Disallowed

ECMAScript 2009 (ES5) features

Object/array literal extensions

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Getter accessors	Supported	Supported
Setter accessors	Supported	Supported
Trailing commas in object literals	Supported	Supported
Trailing commas in array literals	Supported	Supported
Reserved words as property names	Supported	Supported

Object static methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Object.create	Supported	Supported
Object.defineProperty	Supported	Supported
Object.defineProperties	Supported	Supported
Object.getPrototypeOf	Supported	Supported
Object.keys	Supported	Supported
Object.seal	Supported	Supported
Object.freeze	Supported	Supported
Object.preventExtensions	Supported	Supported
Object.isSealed	Supported	Supported
Object.isFrozen	Supported	Supported
Object.isExtensible	Supported	Supported
Object.getOwnPropertyDescriptor	Supported	Supported
Object.getOwnPropertyNames	Supported	Supported

Array methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Array.isArray	Supported	Supported
Array.prototype.indexOf	Supported	Supported
Array.prototype.lastIndexOf	Supported	Supported
Array.prototype.every	Supported	Supported
Array.prototype.some	Supported	Supported
Array.prototype.forEach	Supported	Supported
Array.prototype.map	Supported	Supported
Array.prototype.filter	Supported	Supported
Array.prototype.reduce	Supported	Supported
Array.prototype.reduceRight	Supported	Supported
Array.prototype.sort: compareFn must be function or undefined	Not Supported	Not Supported
Array.prototype.sort: compareFn may be explicit undefined	Supported	Supported

String properties and methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Property access on strings	Supported	Supported
String.prototype.split	Supported	Not Supported
String.prototype.trim	Supported	Supported

Date methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Date.prototype.toISOString	Supported	Supported
Date.now	Supported	Supported
Date.prototype.toJSON	Not Supported	Not Supported

Immutable globals

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
undefined	Supported	Supported
NaN	Supported	Supported
Infinity	Supported	Supported

Number methods

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Number.prototype.toExponential rounds properly	Supported	Supported
Number.prototype.toExponential throws on Â±Infinity fractionDigits	Supported	Supported
Number.prototype.toExponential does not throw on edge cases	Supported	Supported

Strict mode

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
reserved words	Disallowed	Disallowed
"this" is undefined in functions	Disallowed	Disallowed
"this" is not coerced to object in primitive methods	Disallowed	Disallowed
"this" is not coerced to object in primitive accessors	Disallowed	Disallowed
legacy octal is a SyntaxError	Disallowed	Disallowed
assignment to unresolvable identifiers is a ReferenceError	Disallowed	Disallowed
assignment to eval or arguments is a SyntaxError	Disallowed	Disallowed
assignment to non-writable properties is a TypeError	Disallowed	Disallowed
eval or arguments bindings is a SyntaxError	Disallowed	Disallowed
arguments.caller removed or is a TypeError	Disallowed	Disallowed
arguments.callee is a TypeError	Disallowed	Disallowed
(function(){}).caller and (function(){}).arguments is a TypeError	Disallowed	Disallowed
arguments is unmapped	Disallowed	Disallowed
eval() can't create bindings	Disallowed	Disallowed
deleting bindings is a SyntaxError	Disallowed	Disallowed
deleting non-configurable properties is a TypeError	Disallowed	Disallowed
"with" is a SyntaxError	Disallowed	Disallowed
repeated parameter names is a SyntaxError	Disallowed	Disallowed
function expressions with matching name and argument are valid	Disallowed	Disallowed

Function.prototype.bind

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
Function.prototype.bind	Supported	Supported

JSON

Feature	ECMAScript 2021 (ES12) mode	ES5 Standards mode
JSON	Supported	Supported

JavaScript API Context-sensitive help

The syntax editor can display context-sensitive API information.

JavaScript API Context-sensitive help includes the ability to:

- List script elements that are valid at the cursor's location. The system displays suggestions in a pop-up window.
- Add a selected script element at the cursor's location. If the cursor is within or adjacent to a partial entry, the system completes the entry with the selected script element.
- View API documentation for a selected suggestion.
- View the expected parameters and format of the current script element.

If the cursor is adjacent to a text string, the system searches for script elements that start with this text string. For example, while the cursor is within or adjacent to the string `GlideR`, the system displays script elements such as:

- *`GlideRecord`*
- *`GlideRecordSecure`*

Context-sensitive suggestions are based on script type. For example, when working on a business rule, only suggestions from the server API and for objects such as *`current`* and *`previous`* display. When working on a client script, the system only displays suggestions from the client API.

Client-side scripting

Run JavaScript on the client (web browser) when client-based events occur, such as when a form loads, after form submission, or when a field changes value.

The client-side Glide API provides classes and methods that you can use in client scripts.

Before you begin

Understand the limitations of your scripting environment. For example, client scripts run on forms in the Service Portal or Mobile environments can only include certain APIs. For more information, see Mobile client GlideForm (g form) scripting and migration.

Client-side scripting design and processing

Well-designed client scripts can reduce the amount of time it takes users to complete a form.

Proper client-side processing depends on the form loading first. Making record updates prior to form load can produce unexpected results that bypass client-side processing.

If you create client scripts to control field values on a form, you must use another method to control these field values in a list. You can:

- Disable list editing for the table.
- Create appropriate business rules or access controls for list editing.
- Create data policies.
- Create a separate onCellEdit client script.

Restrict list editing

If you create UI policies or client scripts for fields on a form, you must use another method to ensure that data in those fields is similarly controlled in a list.

With the exception of onCellEdit client scripts, UI policies and client scripts apply to forms only. Use the following methods to restrict list editing when using client scripts:

- Disable list editing for the table.
- Create appropriate business rules or access controls for list editing.
- Create data policies.
- Create a separate onCellEdit client script.

Minimize server lookups

Use client data as much as possible to eliminate the need for time-consuming server lookups.

Client scripting uses either data available on the client or data retrieved from the server. The top ways to get information from the server are g_scratchpad and asynchronous GlideAjax lookup.

The primary difference between these methods is that g_scratchpad is sent once when a form is loaded (information is pushed from the server to the client), whereas GlideAjax is dynamically triggered when the client requests information from the server.

Note:
GlideRecord and g_form.getReference() are also available for retrieving server information. However, these methods are no longer recommended due to their performance impact. Both methods retrieve all fields in the requested GlideRecord when most cases only require one field.

Example: Retrieve server data using g_scratchpad

The g_scratchpad object passes information from the server to the client, such as when the client requires information not available on the form.

For example, if you have a client script that needs to access the field u_retrieve, and the field is not on the form, the data is not available to the client script. A typical solution to this situation is to place the field on the form and then always hide it with a client script or UI policy. While this solution may be faster to configure, it is slower to execute.

If you know what information the client needs from the server before the form is loaded, a display business rule can create g_scratchpad properties to hold this information. Theg_scratchpad is sent to the client when the form is requested, making it available to all client-side scripting methods. This is a very efficient means of sending information from the server to the client. However, you can only load data this way when the form is loaded. The business rule cannot be triggered dynamically. In those cases, use an asynchronous GlideAjax call.

For example, assume you open an incident and need to pass this information to the client:

- The value of the system property css.base.color
- Whether or not the current record has attachments
- The name of the caller's manager

A display business rule sends this information to the client using the following script:

```
g_scratchpad.css = gs.getProperty('css.base.color');
g_scratchpad.hasAttachments = current.hasAttachments();
g_scratchpad.managerName =
 current.caller_id.manager.getDisplayValue();
```

To access scratchpad data using a client script:

```
// Check if the form has attachments
if (g_scratchpad.hasAttachments)
    // do something interesting here
else
    alert('You need to attach a form signed by ' +
 g_scratchpad.managerName);
```

Example: Retrieve server data using asynchronous GlideAjax

Asynchronous GlideAjax allows you to dynamically request information from the server.

This script compares the support group of the CI and the assignment group of the incident by name:

```
//Alert if the assignment groups name matches the support group
function onChange(control, oldValue, newValue, isLoading) {

    if (isLoading)
        return;

    var ga = new GlideAjax('ciCheck');

    ga.addParam('sysparm_name', 'getCiSupportGroup');
    ga.addParam('sysparm_ci', g_form.getValue('cmdb_ci'));
    ga.addParam('sysparm_ag',
 g_form.getValue('assignment_group'));
    ga.getXML(doAlert); // Always try to use asynchronous
 (getXML) calls rather than synchronous (getXMLWait)
}

// Callback function to process the response returned from the
 server
function doAlert(response) {

    var answer =
 response.responseXML.documentElement.getAttribute("answer");

    alert(answer);
}
```

This script relies on the accompanying script include:

```
var ciCheck = Class.create();

ciCheck.prototype = Object.extendsObject(AbstractAjaxProcessor,
 {

    getCiSupportGroup: function() {

        var retVal = ''; // Return value
        var ciID   = this.getParameter('sysparm_ci');
        var agID   = this.getParameter('sysparm_ag');
        var ciRec  = new GlideRecord('cmdb_ci');

        // If we can read the record, check if the sys_ids match
        if (ciRec.get(ciID)) {
```

```
            if (ciRec.getValue('support_group') == agID)
                retVal = 'CI support group and assignment group
 match';
            else
                retVal = 'CI support group and assignment group
 do not match';

            // Can't read the CI, then they don't match
        } else {
            retVal = 'CI support group and assignment group do
 not match';
        }

        return retVal;
    }

});
```

Use the setValue() displayValue parameter for reference fields

When using setValue() on a reference field, include the displayValue parameter to avoid additional server calls.

When using *setValue()* on a reference field, be sure to include the reference field display value as the 3rd parameter. If you set the value without the displayValue, the instance does a synchronous call to retrieve the display value for the record you specified. This extra round trip to the server can impact performance.

This example demonstrates the incorrect way to call setValue:

```
var id = '5137153cc611227c000bbd1bd8cd2005';

g_form.setValue('assigned_to', id); // Client needs to go back
 to the server to
                                    // fetch the name that goes
 with this ID
```

Instead, include the display value as an optional parameter in *setValue()*:

```
var id = '5137153cc611227c000bbd1bd8cd2005';
var name = 'Fred Luddy';

g_form.setValue('assigned_to', id, name); // No server call
 required
```

Use UI policy instead of a client script

When possible, consider using a UI policy instead of a client script.

UI policies provide these benefits over client scripts:

- UI policies have an **Order** field to allow full control over the order in which client-side operations take place.
- UI policies do not require scripting to make a field mandatory, read-only, or visible.

Note:
UI policies apply after client scripts.

Validating the input by using a client script

An excellent use for a client script is to validate the input from the user.

This validation improves the user experience because the user finds out if there are data issues before submitting the information.

An example of validation is to verify that the **Impact** field value is valid with the **Priority** field value. In this example, Low impact is not allowed with High priority.

```
if (g_form.getValue('impact') == '3' &&
 g_form.getValue('priority') == '1')
   g_form.showFieldMsg('impact', getMessage('Low impact now
 allowed with High priority'), 'error');
```

Set client script order

Control the order of execution for your client scripts using the Order field. To avoid having two or more client scripts run concurrently and then conflict, you can add an order for the scripts to run in.

Before you begin

Role required: admin

About this task

Adding an order to the client script creates a processing sequence, ordered from lowest to highest number. If two scripts conflict, the client script with the lower number executes first.

Procedure

1. Navigate to **All** > **System Definition** > **Client Script** and open an existing client script or click **New**.
2. Configuring the form layout to include the **Order** field.
3. Add a number to the order field based on what order you want it to run in relation to other client scripts.
   Choose a lower number for the script you want to execute first.

Avoid DOM manipulation

Avoid Document Object Model (DOM) manipulation if possible. It can cause a maintainability issue when browsers are updated.

Instead, use the GlideForm API or consider a different approach for the solution. In general, when using DOM manipulation methods, you have to reference an element in the DOM by ID or using a CSS selector. When referencing out-of-box DOM elements, there is a risk that the element ID or placement within the DOM could change, causing the code to stop working and/or generate errors. Use forethought, caution, and have a full understanding of the risk you are incurring. Review these objects and reduce the use of DOM manipulation methods as much as possible.

Avoid global client scripts

A global client script is any client script where the selected Table is Global. Global client scripts have no table restrictions, therefore they will load on every page in the system introducing browser load delay in the process.

There is no benefit to loading this kind of scripts on every page.

As an alternative, and for a more modular and scalable approach, consider moving client scripts to a base table (such as Task[task] or Configuration Item[cmdb_ci]) that can be derived for all the

child/extending tables. This eliminates the system loading the scripts on every form in the UI - such as home pages or Service Catalog where they are rarely (if ever) needed.

Enclose code in functions

Enclose the code in a client script inside a function.

Client scripts without a function cause issues with variable scope. When code is not enclosed in a function, variables and other objects are available and shared to all other client-side scripts. If you are using the same variable names, it is possible that they could collide. This can lead to unexpected consequences that are difficult to troubleshoot.

Consider this example:

```
var state = "6";

function onSubmit() {

   if(g_form.getValue('incident_state') == state) {
     alert("This incident is Resolved");
   }
}
```

Because the *state* variable is not enclosed in a function, all client-side scripts, have access to it. Other scripts may also use the common variable name *state*. The duplicate names can conflict and lead to unexpected results. These issues are difficult to isolate and resolve. To avoid this issue, ensure that all code is wrapped in a function:

```
function onSubmit() {

   var state = "6";

   if(g_form.getValue('incident_state') == state) {
     alert("This incident is Resolved");
   }
}
```

This solution is much safer because the scope of the variable *state* is limited to the onSubmit() function. Therefore, the *state* variable does not conflict with *state* variables in other client-side scripts.

Run only necessary scripts

To avoid running time-consuming scripts unnecessarily, make sure that client scripts perform only necessary tasks.

The following examples demonstrate improvements to the initial code sample. Each example demonstrates a particular enhancement to the script to improve performance and avoid unnecessary calls.

Remember that client scripts have no **Condition** field. This means that onLoad() and onChange() scripts run in their entirety every time the appropriate form is loaded. This example is an inefficient onChange() client script set to run when the **Configuration item** field changes.

```
//Set Assignment Group to CI's support group if assignment group
 is empty
function onChange(control, oldValue, newValue, isLoading) {

    var ciSupportGroup =
 g_form.getReference('cmdb_ci').support_group;
```

```
    if (ciSupportGroup != '' &&
 g_form.getValue('assignment_group) != '')
        g_form.setValue('assignment_group',
 ciRec.support_group.sys_id);
}
```

This example improves upon the first by replacing the getReference() or GlideRecord lookup with an asynchronous GlideAjax call.

```
//Set Assignment Group to support group if assignment group is
 empty
function onChange(control, oldValue, newValue, isLoading) {

    var ga = new GlideAjax('ciCheck');

    ga.addParam('sysparm_name', 'getSupportGroup');
    ga.addParam('sysparm_ci', g_form.getValue('cmdb_ci'));
    ga.getXML(setAssignmentGroup);
}

function setAssignmentGroup(response) {

    var answer =
 response.responseXML.documentElement.getAttribute("answer");

    g_form.setValue('assignment_group', answer);
}
```

The *isLoading* flag is the simplest way to prevent unnecessary code from taking up browser time in onChange scripts. The *isLoading* flag should be used at the beginning of any script that is not required to run when the form is loading. There is no need to run this script on a form load because the logic would have already run when the field was last changed. Adding the *isLoading* check to the script prevents it from doing a cmdb_ci lookup on every form load.

The *isTemplate* flag indicates that a template is loading.

```
//Set Assignment Group to CI's support group if assignment group
 is empty
function onChange(control, oldValue, newValue, isLoading,
 isTemplate) {

    if (isLoading)
        return;

    var ga = new GlideAjax('ciCheck');

    ga.addParam('sysparm_name', 'getSupportGroup');
    ga.addParam('sysparm_ci', g_form.getValue('cmdb_ci'));
    ga.getXML(setAssignmentGroup);
}

function setAssignmentGroup(response) {

    var answer =
 response.responseXML.documentElement.getAttribute("answer");

    g_form.setValue('assignment_group', answer);
```

```
}
```

If the onChange script should run during loading, use the following convention:

```
function onChange(control, oldValue, newValue, isLoading,
 isTemplate) {

    if (isLoading) {}; // run during loading

    // rest of script here

}
```

The newValue check tells this script to continue only if there is a valid value in the relevant field. This prevents the script from running when the field value is removed or blanked out. This also ensures that there will always be a valid value available when the rest of the script runs.

```
//Set Assignment Group to CI's support group if assignment group
 is empty
function onChange(control, oldValue, newValue, isLoading,
 isTemplate) {

    if (isLoading)
        return;

    if (newValue) {
        var ga = new GlideAjax('ciCheck');

        ga.addParam('sysparm_name', 'getSupportGroup');
        ga.addParam('sysparm_ci', g_form.getValue('cmdb_ci'));
        ga.getXML(setAssignmentGroup);
    }
}

function setAssignmentGroup(response) {

   var answer =
 response.responseXML.documentElement.getAttribute("answer");

   g_form.setValue('assignment_group', answer);
}
```

To have the script react to a value that changes after the form loads, use the newValue != oldValue check.

Note:
This example does not catch users changing a value and then changing it back to its original value.

```
//Set Assignment Group to CI's support group if assignment group
 is empty
function onChange(control, oldValue, newValue, isLoading,
 isTemplate) {

    if (isLoading)
        return;

    if (newValue) {
```

```
        if (newValue != oldValue) {
            var ga = new GlideAjax('ciCheck');

            ga.addParam('sysparm_name', 'getSupportGroup');
            ga.addParam('sysparm_ci',
 g_form.getValue('cmdb_ci'));
            ga.getXML(setAssignmentGroup);
        }
    }
}

function setAssignmentGroup(response) {

   var answer =
 response.responseXML.documentElement.getAttribute("answer");

   g_form.setValue('assignment_group', answer);
}
```

In this example, the GlideAjax call is buried one level deeper by rearranging the script to check as many things available to the client as possible before running the server calls. The script checks the assignment before executing the GlideAjax call. This prevents the server lookup when the **assignment_group** field is already set.

```
//Set Assignment Group to CI's support group if assignment group
 is empty
function onChange(control, oldValue, newValue, isLoading,
 isTemplate) {

    if (isLoading)
       return;

    if (newValue) {
        if (newValue != oldValue) {
            if (g_form.getValue('assignment_group') == '') {
                var ga = new GlideAjax('ciCheck');

                ga.addParam('sysparm_name', 'getSupportGroup');
                ga.addParam('sysparm_ci',
 g_form.getValue('cmdb_ci'));
                ga.getXML(setAssignmentGroup);
            }
        }
    }
}

function setAssignmentGroup(response) {

   var answer =
 response.responseXML.documentElement.getAttribute("answer");

   g_form.setValue('assignment_group', answer);
}
```

Client scripts

Client scripts allow the system to run JavaScript on the client (web browser) when client-based events occur, such as when a form loads, after form submission, or when a field changes value.

Introduction to client scripts, script types, APIs, and good practices

Use client scripts to configure forms, form fields, and field values while the user is using the form. Client scripts can:

- make fields hidden or visible
- make fields read only or writable
- make fields optional or mandatory based on the user's role
- set the value in one field based on the value in other fields
- modify the options in a choice list based on a user's role
- display messages based on a value in a field

Warning:

Client scripts are intended to optimize the user experience on a form. Client scripts are not meant to protect unwanted access to data.

To prevent unwanted access to data, ensure that sensitive fields are hidden or read-only through ACLs or data policies.

For more information, see Access Control List Rules or Data policy.

Where client scripts run

With the exception of `onCellEdit()` client scripts, client scripts only apply to forms and search pages. If you create a client script to control field values on a form, you must use one of these other methods to control field values when on a list.

- Create an access control to restrict who can edit field values.
- Create a business rule to validate content.
- Create a data policy to validate content.
- Create an `onCellEdit()` client script to validate content.
- Disable list editing for the table.

Note:
Client scripts are not supported on ServiceNow mobile applications.

Client script form

Field	Description
Name	Name of the client script.
Table	Table to which the client script applies.
UI Type	Target user interface to which the client script applies.
Type	`onLoad()` — runs when the system first renders the form and before users can enter data. Typically, `onLoad()` client scripts perform client-side-manipulation of the current form or set default record values.

Field	Description
	onSubmit() — runs when a form is submitted. Typically, *onSubmit()* scripts validate things on the form and ensure that the submission makes sense. An *onSubmit()* client script can cancel form submission by returning a value of false. *onChange()* — runs when a particular field value changes on the form. The *onChange()* client script must specify these parameters. • *control*: the DHTML widget whose value changed. **Note:** *control* is not accessible in mobile and service portal. • *oldValue*: the value the widget had when the record was loaded. **Note:** Old values aren't returned for the HTML field type. • *newValue*: the value the widget has after the change. • *isLoading*: identifies whether the change occurs as part of a form load. • *isTemplate*: identifies whether the change occurs as part of a template load. *onCellEdit()* — runs when the list editor changes a cell value. The *onCellEdit()* client script must specify these parameters. • *sysIDs*: an array of the sys_ids for all items being edited. • *table*: the table of the items being edited. • *oldValues*: the old values of the cells being edited. • *newValue*: the new value for the cells being edited. • *callback*: a callback that continues the execution of any other related cell edit scripts. If *true* is passed as a parameter, the other scripts are executed or the change is committed if there are no more scripts. If *false* is passed as a parameter, any further scripts are not executed and the change is not committed.
Field Name	Name of the field to which the script applies. Available only if the script responds to a field value change (onChange or onCellEdit script types).
Application	Application where this client script resides.
Active	Enables the client script when selected. Unselect this field to disable the client script.
Inherited	Indicates whether the client script applies to extended tables.
Global	If true, the client script runs on all views of the table.
View	Only visible when **Global** is unselected. Views on which the client script will run.
Description	Content describing the functionality and purpose of the client script.

Field	Description
Messages	Text string (one per line) available to the client script as localized messages using `getmessage('[message]')`. For additional information, see Translate a client script message.
Script	Contains the client script.
Isolate script	New client scripts are run in strict mode, with direct DOM access disabled. Access to jQuery, prototype, and the window object are also disabled. To disable this on a per-script basis, configure this form and select the **Isolate script** check box. To disable this feature for all new globally-scoped client-side scripts set the system property glide.script.block.client.globals to false.

UI scripts

UI scripts provide a way to package client-side JavaScript into a reusable form, similar to how script includes store server-side JavaScript. Administrators can create UI scripts and run them from client scripts and other client-side script objects and from HTML code.

UI scripts are not supported for mobile.

Global UI scripts

You can create a UI script and designate it as global, which makes the script available on any form in the system. You cannot create a global UI script in a scoped application.

You can mark a UI script as Global to make it available on any form in the system. For example, you can create a UI script that has a function `helloWorld()`, and has the **Global** field checked:

```
function helloWorld() {
  alert('Hi');
}
```

After you create this global UI script, you can call the `helloWorld()` function from any client script or UI policy you write.

Create a UI script

Create a UI script to define reusable client-side JavaScript code.

Procedure

To create UI scripts, navigate to **System UI** > **UI Scripts** and create or edit a record (see table for field descriptions).

UI scripts

Field	Description
Script Name	Name of the UI script. Ensure the name is unique on your system.
API Name	The API name of the UI script, including the scope and script name (for example, x_custom_app.HelloWorld).
Application	Application that contains the UI script.
Active	Indicator of whether the UI script is active. Only active UI scripts can run.

Field	Description
Global	Indicator of whether the script loads on every page in the system. **Note:** Use caution when creating global UI scripts because they can impact performance. You cannot create a global UI script in a scoped application.
Description	Summary of the purpose of the script.
Script	Client-side script to run when called from other scripts.

Run UI scripts

Follow these guidelines when running UI scripts.

Run a UI script from a form

To run a UI script on a form, Create a formatter and add it to a form. In the associated UI macro, include a `g:requires` tag and specify the *name=* parameter as the name of the UI script followed by the .jsdbx extension. Add the formatter on the form view.

This code ensures that the definitions and results of the UI script are immediately available in the browser.

```
<?xml version="1.0" encoding="utf-8" ?>
<j:jelly trim="false" xmlns:j="jelly:core" xmlns:g="glide"
 xmlns:j2="null" xmlns:g2="null">
    <g2:evaluate var="jvar_stamp">
        var now_GR = new GlideRecord('sys_ui_script');
        gr.orderByDesc('sys_updated_on');
        gr.query();
        gr.next();
        gr.getValue('sys_updated_on');
    </g2:evaluate>
    <g:requires name="<UI SCRIPT NAME>.jsdbx"
 params="cache=$[jvar_stamp]" />
</j:jelly>
```

Call a UI script in HTML

To run a UI script from HTML code, use the `<script>` tag and specify the `src=` argument as the API name of the UI script followed by the .jsdbx extension. For example, include the UI script named *CoolClock* with this code:

```
<script language="javascript" src="CoolClock.jsdbx" />
```

Call a UI script from client-side code

Access UI scripts from within client-side code using the *g_ui_scripts* global object. For more information, see GlideUIScripts - Client.

Note:
This class does not support UI scripts with the **Global** field set to true.

Catalog client scripts

Client-side scripts can add dynamic effects and validation to forms. Scripts can apply to service catalog items or variable sets, allowing administrators to use the same functionality that is available on other forms.

You can use client side scripts to:

- Get or set variable values.
- Hide or display variables.
- Make variables mandatory or not.
- Validate form submission.
- Add something to the cart.
- Order something immediately.

Catalog client script considerations

When you create catalog client scripts, be aware of the following considerations.

- Catalog client scripts run when a user orders an item from the service catalog. Catalog client scripts can also run when variables or variable sets for a catalog item are displayed when a user requests that item.
- For a variable to be accessible using a catalog client script, it must have a variable name. Variables without names do not appear in the list of available variables.
- When using standard client scripts on a Requested Item or Catalog Task form, make a note of fields with the same name as variables. If a table field and a variable of the same name are both present on a form, the table field is matched when it is accessed using a script. If this happens, specifically address the variable by naming it `variables.variable name`. For example: `g_form.setValue('variables.replacement', 'false');`
- If you are using record producers to pass variables from the service catalog to other types of records, these variables are made visible in those records with a variable editor, such as the Change Variable Editor UI formatter on Change request forms. You can manipulate these variables using standard client script methods, such as `setDisplay`, `setMandatory`,`setValue`, and `getValue`.
- Catalog client scripts can be used for catalog items included in a wizard.
- You can use the `g_form.refreshSlushbucket(fieldName)` API to update a list collector variable.

Catalog client script differences

Catalog client scripts are very similar to standard client scripts, with a few important differences.

- Instead of selecting a table such as Incident for the script, select a catalog item or variable set. As your system may have a large number of catalog items, you should select a catalog item or variable set using a reference field instead of the choice list that the standard Client Script form uses.
- When using an *onChange()* catalog client script, it is linked to a particular variable instead of a field. The system automatically populates the variable name selection list with any named variables from the catalog item or variable set selected.

Create a catalog client script

Follow this procedure to create a catalog client script.

Procedure

1. Navigate to **All** > **Service Catalog** > **Catalog Administration** > **Catalog Client Scripts**.
   A list of current custom catalog client scripts appears.
2. Click **New**.
3. Fill in the fields, as appropriate (see table).

Field	Description
Name	Enter a unique name for the catalog client script.
Applies to	Select the item type this client script applies to: ○ **A Catalog Item**: enables the Catalog item field. ○ **A Variable Set**: enables the Variable set field.
Active	Select the check box to enable the client script. Clear the check box to disable the script.
UI Type	Whether to apply this to desktop, mobile, or both.
Script	Enter the client script that should run on the service catalog item.
Type	Select when the script should run, such as **onLoad** or **onSubmit**.
Catalog item or Variable set	Select a catalog item or variable set from the list. The field name and options available depend on the selection in the **Applies to** field.
Applies on a Catalog Item view	Select the check box to apply the catalog client script to catalog items displayed within the order screen on the service catalog. Available in the requester view.
Applies on Requested Items	Select the check box to apply the catalog client script on a Requested Item form, after the item is requested. Available in the fulfiller view. See VEditor .
Applies on Catalog Tasks	Select the check box to apply the catalog client script when a Catalog Task form for the item is being displayed. Available in the fulfiller view. See VEditor .
Applies on the Target Record	Select the check box to support the catalog UI policy on a record created for task-extended tables via record producers. See Default variable editor .

4. Click **Submit**.

Catalog client script examples

Examples of client scripts to perform common actions.

Example: Get the value of a variable

Use the following syntax to obtain the value of a catalog variable. Note that the variable must have a name. Replace `variable_name` with the name of the variable.

```
g_form.getValue('variable_name');
```

Example: Restrict the number of characters a user can type in a variable

This is an example of a script that runs when the variable is displayed, rather than when the item is ordered.

```
function onLoad(){
  var sd = g_form.getControl('short_description');
  sd.maxLength=80;
}
```

Mobile client GlideForm (g form) scripting and migration

Client scripting for mobile is identical to scripting for the web, with some exceptions. All new scripts must conform to certain guidelines. The following items are affected on the mobile platform: client scripts, UI policies, navigator modules, and UI actions.

Client scripts

For new or existing scripts to be valid for mobile, they must conform to the following requirements:

- Use the new mobile methods in place of `g_form.getControl()`.
- Do not use deprecated methods.
- Do not reference unsupported browser objects.
- Do not make synchronous JavaScript, `GlideAjax`, and `GlideRecord` calls.
- Do not call methods that are not available for mobile.
- Enable scripts to run on the mobile UI.

Requirements	
Use the new mobile methods	Several new methods are available for modifying form fields instead of directly manipulating the HTML. These methods replace previous usages of `g_form.getControl()`, which is deprecated for the mobile platform. In your existing scripts, ensure that the new methods are used in place of methods that are not valid on the mobile platform. For information on these new methods, refer to Mobile GlideForm() API.
Do not use deprecated methods	The following methods have been deprecated for the mobile platform because direct access to HTML elements is not allowed: • `g_form.getControl()` • `g_form.getFormElement()` • `g_form.getElement()` To ensure that existing scripts are compatible, remove all calls to deprecated methods from your code. For new scripts, do not use deprecated methods if you want the script to be valid for mobile. For `g_form.getControl()`, some of the functionality previously included with this method has been extracted to individual methods. Instead of `g_form.getControl()`, use the new methods described on the developer site.

Requirements (continued)

Do not reference unsupported browser objects	The following browser objects are not supported in mobile scripts: • Window • jQuery or Prototype ($, $j, or $$) • Document Make sure that new scripts do not use these objects, and remove any usage of these objects from your existing scripts. Use *GlideForm (g_form)* instead, which provides methods such as *setLabel()*, *addDecoration()*, and *hasField()* for accomplishing the same tasks.
Do not make synchronous JavaScript calls	The mobile platform does not allow synchronous JavaScript calls. The *g_form.getReference()* method must now have the callback parameter defined. For example: `g_form.getReference(fieldName, callback)` Be sure that all *g_form.getReference()* calls include the callback parameter. For example, the following script does not include a callback and is incompatible with the mobile platform: `var userName = g_form.getReference('assigned_to').user_name; g_form.setValue('u_assigned_user_name', userName);` The following script has been updated to include the callback and is compatible with the mobile platform: `g_form.getReference('assigned_to', function(now_GR) { g_form.setValue('u_assigned_user_name', gr.user_name); });`
Do not make synchronous Ajax calls	The mobile platform does not allow synchronous *GlideAjax* calls. Any use of *getXMLWait()* in a *GlideAjax* call will not work on the mobile platform. Be sure that all *GlideAjax* calls are asynchronous. For more on synchronous versus asynchronous *GlideAjax* calls and *getXMLWait()*, see AJAX. For information on the available *GlideAjax* methods, refer to the GlideAjax API.
Do not make synchronous *GlideRecord* calls	The mobile platform does not allow synchronous GlideRecord calls. Make sure that any existing *GlideRecord* calls include a callback. For example, the following script does not include a callback and is incompatible with the mobile platform: `var now_GR = new GlideRecord('incident'); gr.addQuery('number', g_form.getValue('related_incident')); gr.query(); gr.next();`

Requirements (continued)

```
g_form.setValue('u_related_incident_description',
gr.short_description);
```

The following script has been updated to include the callback, and is compatible with the mobile platform:

```
var now_GR = new GlideRecord('incident');
gr.addQuery('number',
g_form.getValue('related_incident'));
gr.query(function(now_GR) {
    gr.next();

g_form.setValue('u_related_incident_description',
gr.short_description);
});
```

Do not use methods unavailable on the mobile platform	Due to the limitations and reduced functionality that is imposed by the mobile platform, the following methods are not deprecated but are not available on the mobile platform. If these run on the mobile platform, no action occurs: • `showRelatedList ()` • `hideRelatedList ()` • `showRelatedLists ()` • `hideRelatedLists()` • `flash()` • `getSections()` • `enableAttachments()` • `disableAttachments()` • `setReadonly()` (Note that `setReadOnly()` is available) • `getParameter()`
Enable scripts for mobile	Scripts must be enabled for the mobile platform.

Note:
Focusing an element on a mobile form is not supported.

UI policies

Use the **Run scripts in UI type** field to determine whether scripts run on the mobile platform, the desktop, or both. Update existing policies so that they apply to either the mobile platform or both. For new scripts, also ensure that the mobile option or both is selected.

Navigator modules

For existing code, modules must be transferred to either the sys_ui_application or sys_ui_module tables to be available on the mobile platform. When developing new code, be sure that all modules are created in the sys_ui_application or sys_ui_module tables.

UI actions

UI actions must be transferred to the sys_ui_ng_action table to appear on the mobile platform. UI action scripts that do not use deprecated methods do not require changes to the script itself. For new UI actions, be sure that they are created in the sys_ui_ng_action table.

AJAX

AJAX (asynchronous JavaScript and XML) is a group of interrelated, client-side development techniques used to create asynchronous Web applications.

AJAX enables web applications to send and retrieve information to and from a server in the background, without impacting the user experience with the displayed web page.

GlideAjax

The `GlideAjax` class allows the execution of server-side code from the client. `GlideAjax` calls pass parameters to the script includes, and, using naming conventions, allows the use of these parameters.

Note:
This functionality requires a knowledge of JavaScript.

Using `GlideAjax`:

- Initialize `GlideAjax` with the name of the script include that you want to use.
- When creating the script include, you must set the name field to be exactly the same as the class name.
- When creating the script include, you must select the **Client callable** check box.
- Specify the parameter `sysparm_name`. `GlideAjax` uses `sysparm_name` to find which function to use.
- Any extra parameters may be passed in, all of which must begin with `sysparm_`. Avoid using predefined parameter names:
  - `sysparm_name`
  - `sysparm_function`
  - `sysparm_value`
  - `sysparm_type`
- Code is then executed with the `getXML()` or `getXMLWait()` functions.

For additional information, refer to the GlideAjax API.

Examples of asynchronous GlideAjax

There are two parts to the asynchronous `GlideAjax` script: client-side and server-side code.

Hello World: Returning a value from the server

Example: Client side

This code runs on the client (the web browser). Create a client script as normal. This sends the parameters to server, which then does the processing. So that the client does not wait for the result, a callback function is used to return the result, passed to the `getXML()` function. (In this case it is called `HelloWorldParse`.)

The `getXMLWait()` function does not need a separate callback function, but this will block the client. If the client-server communication takes a long time (for example on slow networks), the

application will seem unresponsive and slow. An example of *getXMLWait()* is in the following section.

```
var ga = new GlideAjax('HelloWorld');
ga.addParam('sysparm_name', 'helloWorld');
ga.addParam('sysparm_user_name', "Bob");
ga.getXML(HelloWorldParse);

function HelloWorldParse(response) {
  var answer =
 response.responseXML.documentElement.getAttribute("answer");
  alert(answer); }
```

Example: Server side

The server-side code for the above function. Do not create a business rule, but instead navigate to **System Definition > Script Include** and create a new script. Paste in the code below.

Note:
You must set the name of the script include to `HelloWorld`.

- The *sys_script_include* code must extend the *AbstractAjaxProcessor* class and be Glide AJAX enabled.
- Function names starting with "_" are considered private and are not callable from the client.
- Avoid overriding methods of *AbstractAjaxProcessor*, including `initialize`. While it is possible to invoke methods of your superclass object which you have overridden, it is complicated and best avoided altogether.

```
var HelloWorld = Class.create();
HelloWorld.prototype =
 Object.extendsObject(AbstractAjaxProcessor, {
   helloWorld:function() { return "Hello " +
 this.getParameter('sysparm_user_name') + "!"; } ,
   _privateFunction: function() { // this function is not client
 callable
    }
 });
```

This results in an alert box that says 'Hello Bob!' when you visit the form.

Returning multiple values

Since the response is an XML document we are not limited to returning a single `answer` value. Here is a more complex example returning multiple XML nodes and attributes.

Example: AJAX processor script include

```
/*
 * MyFavoritesAjax script include Description - sample AJAX
 processor returning multiple value pairs
 */
var MyFavoritesAjax = Class.create();
MyFavoritesAjax.prototype =
 Object.extendsObject(AbstractAjaxProcessor, {

  /*
```

```
  * method available to client scripts call using:
  * var gajax = new GlideAjax("MyFavoritesAjax");
  * gajax.addParam("sysparm_name","getFavorites");
  */
 getFavorites: function() { // build new response xml element
 for result
        var result = this.newItem("result");
   result.setAttribute("message","returning all favorites");

   //add some favorite nodes with name and value attributes
        this._addFavorite("color","blue");
        this._addFavorite("beer","lager");
        this._addFavorite("pet","dog");
      },
  // all items are returned to the client through the inherited
 methods of AbstractAjaxProcessor
  _addFavorite: function(name, value) {
         var favs = this.newItem("favorite");
    favs.setAttribute("name",name);
    favs.setAttribute("value",value); },

 type:"MyFavoritesAjax"

 });
```

Example: Client script

```
// new GlideAjax object referencing name of AJAX script include
var ga = new GlideAjax("MyFavoritesAjax");
// add name parameter to define which function we want to call
// method name in script include will be getFavorites
ga.addParam("sysparm_name","getFavorites");

// submit request to server, call ajaxResponse function with
 server response

ga.getXML(ajaxResponse);

function ajaxResponse(serverResponse) {
  // get result element and attributes
  var result =
 serverResponse.responseXML.getElementsByTagName("result");
  var message = result[0].getAttribute("message");

  //check for message attribute and alert user
  if(message) alert(message);

  //build output to display on client for testing
  var output = "";

  // get favorite elements
  var favorites =
 serverResponse.responseXML.getElementsByTagName("favorite");
  for(var i = 0; i < favorites.length; i ++) {
    var name = favorites[i].getAttribute("name");
    var value = favorites[i].getAttribute("value");
    output += name + " = " + value + "\n "; }
```

```
  alert(output); }
```

Example: XML response

```
<xml sysparm_max= "15" sysparm_name="getFavorites"
 sysparm_processor="MyFavoritesAjax">
  <result message = "returning all favorites"></result>
  <favorite name = "color" value = "blue"></favorite>
  <favorite name = "beer" value = "lager"></favorite>
  <favorite name = "pet" value = "dog"></favorite>
</xml>
```

Examples of synchronous GlideAjax

Use synchronous when your script cannot continue without the *GlideAjax* response. This stops the session until the response is received.

If your use case demands that no further processing can occur until the *GlideAjax* response has been received, you can use *getXMLWait()*. However, because this will slow down your code and lock the user session until the response is received, it is generally recommended that you use *getXML()* with a callback function.

Note:
Do not use *AJAXEvaluateSynchronously*.

The *getXMLWait()* method is not available in scoped applications.

This code results in a client-side alert that displays `The Server Says Hello Bob!`.

The client code.

```
var ga = new GlideAjax('HelloWorld') ;
ga.addParam('sysparm_name','helloWorld');
ga.addParam('sysparm_user_name',"Bob");
ga.getXMLWait();
alert(ga.getAnswer());
```

The server-side script include code.

```
var HelloWorld = Class.create();
HelloWorld.prototype =
 Object.extendsObject(AbstractAjaxProcessor, {
helloWorld: function()
 { return "The Server Says Hello " +
 this.getParameter('sysparm_user_name') + "!"; } } );
```

AJAXClientHelper

Provides helper functions for Ajax clients to retrieve a value from an Ajax client.

Where to use

Use this script include wherever your need to retrieve a value from an Ajax client.

Method summary

Method summary	Description
`getDisplay()`	Gets the display value from the choice list.

Method detail

API method	Description	Input parameters	Output returns
`getDisplay()`	Gets the display value from the choice list.	none	The display value.

Example: Curl

```
// getDisplay
(function(table, sysId) {
    var ga = new GlideAjax('AjaxClientHelper');
    ga.addParam("sysparm_name", "getDisplay");
    ga.addParam('sysparm_table', 'incident');
    ga.addParam('sysparm_value',
 "0598b22b877313003c1c8467a7cb0b71");
    ga.getXMLWait();
    return ga.getAnswer();
}("incident", "0598b22b877313003c1c8467a7cb0b71")); // Returns
 'INC0010001'
```

Useful scripts

Server-side and client scripts that provide useful functionality not included in the core system.

Business rule use cases

Use cases for business rules include aborting a database action and restricting record access.

Abort a database action

You can use a before business rule script to cancel or abort the current database action using the `current.setAbortAction(true)` method.

If the before business rule is executed during an insert action, and a condition in the script calls `current.setAbortAction(true)`, the new record stored in `current` is not created in the database.

Restricting record access

You can use a query business rule that executes before the database query to prevent users from accessing certain records.

Warning:
The customization described here was developed for use in specific instances, and is not supported by Now Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

Consider the following example from a default business rule that limits access to incident records.

Default business rule limits access to incident records

Name	Table	When
incident query	Incident	before, query

Example: Restricting record access

In the following example, users are restricted from accessing incident records unless they have the itil role and are listed in the *Caller* or *Opened by* field. When self-service users open a list of incidents, they can only see the incidents they submitted.

```
if (!gs.hasRole("itil")&& gs.isInteractive()) {
  var u = gs.getUserID();
  var qc = current.addQuery("caller_id",
 u).addOrCondition("opened_by",
 u).addOrCondition("watch_list","CONTAINS", u);
  gs.print("query restricted to user: " + u);}
```

Note:
You can also use access controls to restrict the records that users can see. For information, see Access Control List Rules.

Schedule script for weekdays

Type: Business Rules/Client Scripts.

This script schedules the script for weekdays. Insert any script where it says "Your Script Here."

```
var go ='false';
var now =new Date();

// Correct time zone, which is by default GMT -7
now.setHours(now.getHours()+8);
var day = now.getDay();

// No go on Saturday or Sunday
if(day !=0&& day !=6){

// (your script here)

}
```

Set date field according to current date

This script sets a date field depending on the current day of the week. In this example, if the day is Monday through Wednesday, it sets the date to this coming Monday; otherwise it sets the date field to next Monday.

```
function setCabDate(){
var today = new Date();
var thisDay = today.getDay();

//returns 0 for Sunday, 1 for Monday, through 6 for Saturday.
var thisMon = new GlideDateTime();
thisMon.setDisplayValue(gs.beginningOfThisWeek());
var nextMon = thisMon.getNumericValue();
```

```
nextMon +=(1000*60*60*24*7);

if((thisDay <4)&&(thisDay >0))
  //if today is Mon thru Wed (thisDay = 1, 2, or 3), set cab to
 this coming Monday.

 current.u_req_cab_rev_date.setDateNumericValue(thisMon.getNumer
icValue());
else if((thisDay >=4)||(thisDay ==0))
  //if today is Thurs thru Sun (thisDay = 4, 5, 6, or 0), set
 cab to next Monday.
  current.u_req_cab_rev_date.setDateNumericValue(nextMon);
}
```

To validate the input of all date/time fields, you can use the following in a validation script (**System Definition** > **Validation Scripts**). Because the date/time format is hard coded in this script, it must match your instance's date/time format. If your instance's date/time format changes, you must update your validation script.

Set the validation script's type to *Date/Time*. Then, with this validation script, if a user enters an incorrect format in a date/time field, they receive an error message.

```
function validate(value){
// empty fields are still valid dates
if(!value)
    return true;

// We "should" have the global date format defined always
 defined. But there's always that edge case.
if(typeof g_user_date_time_format !=='undefined')
    return isDate(value, g_user_date_time_format);

// if we don't have that defined, we can always try guessing
return parseDate(value)!==null;}
```

For more information, see Validation script use case - Date and time.

Client-side script use cases

Use cases for client-side scripts include displaying field messages, changing form colors, adding fields, and creating UI routing actions.

Add a field to the service catalog checkout

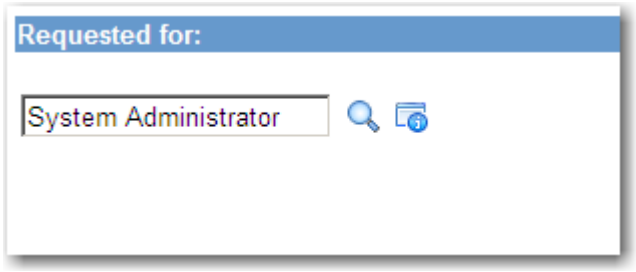
This is an example of adding a **Company** field to the checkout below the **Requested for** field using non-cart layout macros, that is, *glide.sc.use_cart_layouts* is `false`.

Before you begin

Role required: Admin

About this task

Requested for field



This field passes a provided value to the **Company** field of the Service Catalog Request.

This example makes the following assumptions:

- The instance using two-step checkout. If two-step checkout is not enabled, enable it before beginning. For more information, see Service Catalog checkout models.
- This example populates the **Company** field on the Service Catalog Request form. If the field does not appear on the form, configure the form before beginning. For instructions, see Personalize a form.

Procedure

1. Go to **System UI > UI Macros** and select **servicecatalog_cart_template**.
2. Find where there are hidden variables declared and add the following line:

```
<input type="HIDDEN" name="cart_id" id="cart_id"
 value="$[sc_cart.sys_id]" />
```

3. Find the following code, which generates the **Requested For** code:

```
<tr class="header">
<td width = "30%">
  ${gs.getMessage('Requested for')}:
  </td>
  <td width="70%">
    <label for="requestor_location">${gs.getMessage('Deliver
 to')}:</label>
  </td>
</tr>
<tr><td>$[SP]</td>
</tr>
  <tr><td valign="top">
    <j2:if test="$[jvar_can_delta_rf == false]">
      $[sc_cart.requested_for.getDisplayValue()]
    </j2:if>
    <j2:if test="$[jvar_can_delta_rf != false]">
      <g2:catalog_requested_for />
    </j2:if>
  </td>
  <td>
    <textarea id="requestor_location" style="width:100%"
 rows="4"
```

```
      name="requestor_location" wrap="soft"
 onChange="catDeliveryAddress('$[sc_cart.sys_id]',
 'requestor_location');">
      $[sc_cart.delivery_address]
    </textarea>
  </td>
</tr>
<tr>
  <td>$[SP]</td>
</tr>
```

4. Add the following code afterward:

```
<tr class="header">
  <td colspan="2">Company</td>
</tr>
<tr>
  <td>$[SP]</td>
</tr>
<tr>
  <td colspan="2">
    <g2:ui_reference name="core_company" table="core_company"
 onchange="setCartValue()"/>
  </td>
</tr>
<tr>
  <td>$[SP]</td>
</tr>
```

Note:
The `'ui_reference'` macro defines a reference field. There are several macros for different field types. You can see examples of these field types under **System UI -> UI Macros**. These macros start with `'ui_'`. For this example, the reference field created is named **core_company**. For more information, see UI macros.

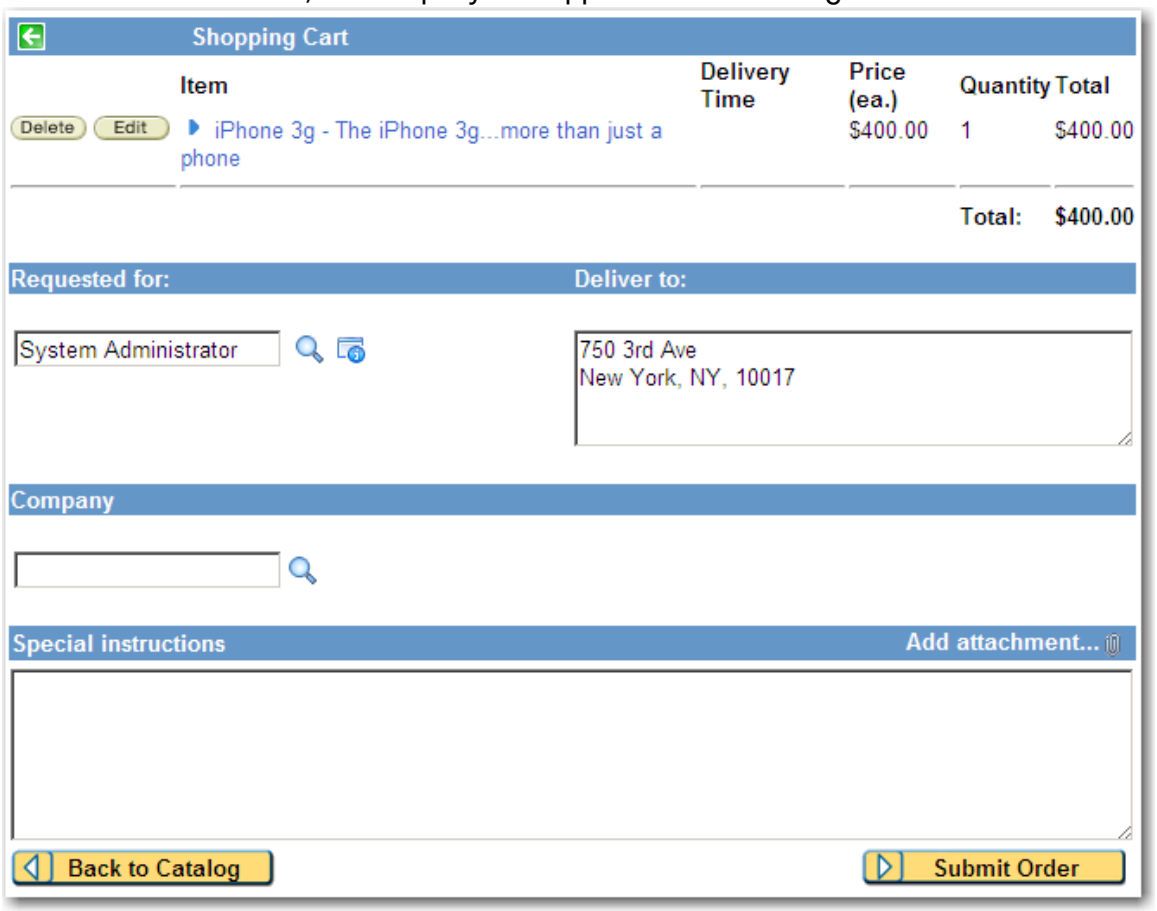
5. Now navigate to **System UI > UI Pages** and select the **servicecatalog_checkout_one** UI Page.
6. Add the followings script to the **Client script** field.

```
function setCartValue() {
  var newField = gel('core_company');
  var myCart = gel('cart_id');
  var cart_item = new GlideRecord('sc_cart_item');
  cart_item.addQuery('cart', myCart.value);
  cart_item.query();
  if(cart_item.next()) {
    cart_item.hints = "<hints><entry
 key='sysparm_processing_hint'
 value='setfield:request.company=" + newField.value +
 "'/></hints>";
    cart_item.update();
  }
}
```

For this example, the reference field was called **core_company**, and the field being populated on the request is **company**. If different fields are used:

- Find this line: `var company = gel('core_company');` and replace **core_company** with the name of the field in the checkout.
- In the line that starts with `'cart_item.hints'` replace `'request.company'` with the name of the field to be populated on the request ticket where `'request'` is the request being generated and `'company'` is the name of the field.

Result
When an item is ordered, the company field appears on the Catalog form:



Add autofill functionality

Add autofill functionality is also called incident template, auto assignments, quick calls, call script, or auto populate.

To auto-fill a **Short Description** field based on the **Subcategory** selected:

1. Create a lookup table.
2. Populate the key field, **Subcategory**.
3. Populate the auto-filled field, **Short Description**.

```
function onChange(control, oldValue, newValue, isLoading) {
  if (isLoading) { return; }
  var newrec = gel('sys_row');
  //Check if new record
  if (newrec.value == -1) {
    var lookup = new GlideRecord('u_short_desc_lookup');
    lookup.addQuery('u_subcategory',
 g_form.getValue('subcategory'));
```

```
    lookup.query();
    var temp; //temp var - reusable
    if (lookup.next()) {
      temp = lookup.u_short_description;
        if (null != temp) { //Set the form value from lookup if
 there is a lookup value
          g_form.setValue('short_description', temp); }
        else {
          g_form.setValue('short_description', "" ); } }
   else {
     //If a lookup record does not exist based on
 lookup.addQuery
     //Then set to UNDEFINED or NULL depending on type
      g_form.setValue('short_description', ""); } }
}
```

You can populate many fields or pull in call script questions into the **Comments** field so call center personnel gather good information to pass on to a technician.

Setting a field to password reset

Lookup table has a record with the following key and auto-fill settings:

- **Subcategory** = *Password*
- **Short Description** = *Password Reset*

Client script settings:

- **Type** = *onChange*
- **Table name** = *incident*
- **Field name** = *Subcategory*

When the user selects the subcategory of *Password* on the Incident form, a client script looks up the matching record and sets the short description equal to *Password Reset*.

Change form color on state change

Changes color of a form field of the form on state change. The script can easily be changed to adjust any property of any object on the page accessible via the HTML DOM.

> **Warning:**
> The customization described here was developed for use in specific instances, and is not supported by Now Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

Name: Change Form Color on State Change

Type: Client script

Description: Changes color of a form field of the form on state change. The script can easily be changed to adjust any property of any object on the page accessible via the HTML DOM.

Script:

```
function onChange(control, oldValue, newValue, isLoading) {
  var elementID = gel("incident.priority");
    switch(newValue) {
```

```
     case "1": elementID.style.backgroundColor = "red"; break;
     case "2": elementID.style.backgroundColor = "tomato";
break;
     case "3": elementID.style.backgroundColor = "orange";
break;
     case "4": elementID.style.backgroundColor = "yellow";
break;
     case "5": elementID.style.backgroundColor = "green";
break;
     default: elementID.style.backgroundColor = "white";
break; } }
```

Create a UI routing action

This solution enables you to create a record with the service desk without knowing whether it is an incident or request item; the service desk can then route the record to the appropriate table.

Before you begin

Role required: **Admin**

Procedure

1. Create a new table that extends the `task` table (for example, New Call).
2. Create a module to create a new New Call record.
3. Create any fields that you want on the New Call table.

   The only fields you need are those fields necessary to determine whether the new call should route to an Incident or a Request Item. Ensure that the form contains any fields that you want to pass to the Incident or Request Item. In this example, the following are created on the form:

   - **Requested for** (reference)
   - **Location** (reference)
   - **Call type** (choice with two values--Incident and Request)
   - **Request Item** (reference to the `sc_cat_item` Item table)
4. Add some UI policies to set a couple of fields to mandatory and hide the **Request item** field based on the **Call type** selection.
5. Remove unnecessary buttons and functionality from the form.
6. Create a new UI Action button.
   This button redirects the user to either an incident or a request. It also creates the incident record and copies values to the incident and the Request Item form.

```
var reqItem = current.u_item;
var requestedFor = current.u_requested_for;
var location = current.location;

if(current.u_incident_request == 'Incident'){
  //Create a new incident record and redirect to the new
 incident
  var rec = new GlideRecord('incident');
  rec.initialize();
  rec.caller_id = requestedFor;
  rec.location = location;
  rec.insert();
  action.setRedirectURL(rec);
```

```
}

if(current.u_incident_request == 'Request'){
  //Build the url and route the user to the request item
  var url = '';
  if(current.u_item.sys_class_name == 'sc_cat_item_guide'){
    url =
 'com.glideapp.servicecatalog_cat_item_guide_view.do?sysparm_in
itial=true&sysparm_guide=' +
      reqItem + '&sysparm_user=' + requestedFor +
 '&sysparm_location=' + location;
  }
  else{
    url =
 'com.glideapp.servicecatalog_cat_item_view.do?sysparm_id=' +
 reqItem + '&sysparm_user=' +
      requestedFor + '&sysparm_location=' + location;
  }
  action.setRedirectURL(url);
}
```

7. The **Route** button in the preceding example passes the **Requested for** and **Location** values in the URL to the Request Item form.
   Ensure that you have variables called *requested_for* and *location* on your item, record producer, or order guide that map these values using the following client script. There is a limit as to how much information you can pass, as the URL has a restricted length. Avoid sending information from long text fields using this method.

```
function onLoad() {
  var url = document.location.toString();
  var userKey = 'sysparm_user=';
  var locKey = 'sysparm_location=';
  var userPosition = url.indexOf(userKey);
  var locPosition = url.indexOf(locKey)
  if (userPosition != -1) {
    var user = url.substr(userPosition+userKey.length, 32);
    g_form.setValue('requested_for', user);
  }
  if (locPosition != -1) {
    var loc = url.substr(locPosition+locKey.length, 32);
    g_form.setValue('location', loc);
  }
}
```

Display field messages

Rather than use JavaScript *alert()*, for a cleaner look, you can display an error on the form itself. The methods *showFieldMsg()* and *hideFieldMsg()* can be used to display a message just below the field itself.

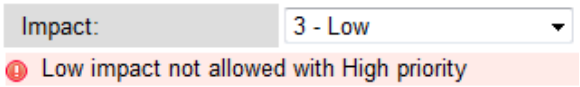
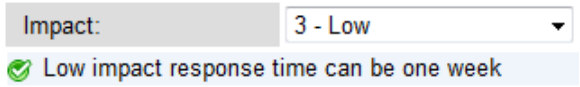
showFieldMsg and *hideFieldMsg* are methods that can be used with the *g_form* object.

These methods are used to change the form view of records (Incident, Problem, and Change forms). These methods may also be available in other client scripts, but must be tested to determine whether they work as expected.

When a field message is displayed on a form on load, the form scrolls to ensure that the field message is visible. Ensuring that users do not miss a field message because it was off the screen.

The global property *glide.ui.scroll_to_message_field* controls automatic message scrolling when the form field is offscreen (scrolls the form to the control or field).

Method Detail

Method Detail	Parameters	Example
showFieldMsg(input, message, type, [scrollform])	• input — name of the field or control • message — message you would like to appear • type — 'info', 'error', or 'warning'; defaults to info if not supplied • scroll form — (optional) Set *scrollForm* to false to prevent scrolling to the field message offscreen	Error Message `g_form.showFieldMsg('impact','Low impact not allowed with High priority','error');`  Informational Message `g_form.showFieldMsg('impact', 'Low impact response time can be one week','info');` `//or this defaults to info type` `//` `g_form.showFieldMsg('impact', 'Low impact response time can be one week');` 
hideFieldMsg(input)	• input — name of the field or control • clearAll — (optional) boolean parameter indicating whether to clear all messages. If true, all messages for the field are cleared. If false or empty, only the first message is removed	Removing a Message `//this will clear the first message printed to the field` `g_form.hideFieldMsg('impact');`

Legacy support

The *showErrorBox()* and *hideErrorBox()* methods are not recommended.

Using client and server code in a UI action

You can use a script to validate input upon a UI Action click on the client side before updating the record on the server side. The user will not have to click the button twice to validate the required fields and update the record.

The script calls the client function for client-side validation, and the UI action completes the task if it passes. The `code that runs without onclick` statement ensures that the server-side function does not run until the client function is no longer running. If successful, the server-side function runs and updates the record.

```
// Client-side onclick function
function resolveIncident() {
  // Set the 'Incident state' and 'State' values to 'Resolved',
 and display mandatory fields
  g_form.setValue('incident_state', 6);
  g_form.setValue('state', 6);
  g_form.setValue('resolved_by', g_user.userID);

  // Call the UI action and skip the 'onclick' function
  gsftSubmit(null, g_form.getFormElement(),
 'resolve_incident'); //MUST call the 'Action name' set in this
 UI Action
}

// Code that runs without 'onclick'
// Ensure call to server-side function with no browser errors
if (typeof window == 'undefined')
  serverResolve();

// Server-side function
function serverResolve() {
  current.incident_state = IncidentState.RESOLVED;
  current.state = IncidentState.RESOLVED;
  current.resolved_by = gs.getUserID();
  current.update();
}
```

gsftSubmit(String control, Object form, String action_name)

Submits a form as if the user had clicked a UI Action after checking for required fields and executing *`onSubmit()`* client scripts. Enables calling client-side code and server-side code and in a single UI action.

Parameters

Name	Type	Description
control	String	Name of a form button on which you want to simulate a user click. Use `null` otherwise.
form	Object	Optional. The form element that contains the submitted input. You can retrieve this value by calling the *`g_form.getFormElement()`* method.
action_name	String	Action name. This value is provided in the record listed in the UI Actions [sys_ui_action] table.

Parameters (continued)

Name	Type	Description
		For example, `'resolve_incident'` is the action name of the **Resolve** button in the Incident [incident] table.

Related topics

GlideForm - getFormElement()

Scoped GlideSystem - eventQueue(String name, Object instance, String parm1, String parm2, String queue)

GlideUser - Client

UI actions

Field script use cases

Common use cases for field customization scripts.

> **Warning:**
> The customization described here was developed for use in specific instances, and is not supported by Now Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

For more information, see Server API reference.

Automatically populate a field

The following example shows how to use a client script to auto-fill a **Short Description** based on the selected **Subcategory**.

In this case, if the table has a record with **Subcategory** = Password and **Short Description** = Password Reset. When the user selects the **Subcategory** of **Password** on the Incident form, a client script looks up the matching record and sets **Short Description** equal to Password Reset.

Client script settings:

- *Type* = onChange
- *Table name* = incident
- *Field name* = Subcategory

```
function onChange(control, oldValue, newValue, isLoading){
    if(isLoading){return;}
    var newrec = gel('sys_row');
    //Check if new record
    if (newrec.value == -1) {
        var lookup = new GlideRecord('u_short_desc_lookup');
        lookup.addQuery('u_subcategory',
 g_form.getValue('subcategory'));
        lookup.query();
        var temp; //temp var - reusable
        if(lookup.next()){
            temp = lookup.u_short_description;
```

```
            if(null != temp) {
                //Set the form value from lookup if there is a
 lookup value
                g_form.setValue('short_description', temp);
            } else {
                g_form.setValue('short_description',"");
                }
         } else {
             //If a lookup record does not exist based on
 lookup.addQuery
             //Then set to UNDEFINED or NULL depending on type
             g_form.setValue('short_description',"");
             }
    }
}
```

Disable HTML tags in descriptions

This code disables HTML tags in **Description** and **Short Description** fields by substituting the tags with non-executing versions.

```
doit();

function doit(){
 var desc = current.description.toString();
 var shdesc = current.short_description.toString();
 if(desc.indexOf('script>')>-1|| shdesc.indexOf('script>')>-1){
   desc = desc.replace(/<script>/g,"(script)");
   current.description =
 desc.replace(/<\/script>/g,"(\/script)");
   shdesc = shdesc.replace(/<script>/g,"(script)");
   current.short_description =
 shdesc.replace(/<\/script>/g,"(\/script)");}
}
```

Eliminate leading and trailing spaces in fields

This example of the script trims trailing and leading spaces in the *FirstName* and *LastName* fields of the sys_user.

```
doit();

function doit(){
  var now_GR =new GlideRecord('sys_user');
  gr.query();
  while(gr.next()){
    if((gr.first_name.toString().length!=
 gr.first_name.toString().trim().length)||(gr.last_name.toString
().length!= gr.last_name.toString().trim().length)){
     gr.first_name= gr.first_name.toString().trim();
     gr.last_name= gr.last_name.toString().trim();
     gr.autoSysFields(false);
     gr.update();}}
}
```

Make a field label flash

The following client script example is for the number field on incident. The label flashes for two seconds.

```
g_form.flash("incident.number","#FFFACD",0);
```

The arguments for the flash method are as follows:

- *tablename.fieldname*
- RGB color or acceptable CSS color like "blue" or "tomato"
- Integer that determines how long the label flashes:
  - 2 for a 1-second flash
  - 0 for a 2-second flash
  - -2 for a 3-second flash
  - -4 for a 4-second flash

Note:
Do not specify this argument if you want the field label colored the specified color.

Make a field label bold

This client script makes the label of a particular field bold. In this case, the field is the **Short Description** on the **Incident Table**.

```
function onLoad(){
  var l = g_form.getLabel('incident.short_description');
  l.style.fontWeight = 'bold';}
```

Make fields read-only

This onLoad client script makes the following fields on the Incident [incident] table read-only:

- **Incident state**
- **Impact**
- **Urgency**
- **Priority**
- **Configuration item**
- **Assigned to**

The script also removes the magnifying glass for the read-only Reference Fields (**Configuration item** and **Assigned to**).

```
function onLoad(){
var incidentState = g_form.getValue('incident_state');
if( incidentState == '6'|| incidentState == '7'){
   g_form.setReadonly('incident_state',true);
   g_form.setReadonly('impact',true);
   g_form.setReadonly('urgency',true);
   g_form.setReadonly('priority',true);
   g_form.setReadonly('cmdb_ci',true);
   g_form.setReadonly('assigned_to',true);}}
```

Set current date/time in field

You can set date and time values in client scripts and script includes.

Client script

You can use the following two lines to set the current date and time in a date/time field. This approach bypasses the problem of getting the value into the proper format and proper time zone.

```
var ajax = new GlideAjax('MyDateTimeAjax');
  ajax.addParam('sysparm_name','nowDateTime');
  ajax.getXML(function(){
    g_form.setValue('put your field name here',
 ajax.getAnswer());});
```

For more information on running server-side scripts with the client, refer to GlideAjax.

System script include

```
// Be sure the Glide AJAX enabled option is checked

var MyDateTimeAjax = Class.create();
MyDateTimeAjax.prototype =
 Object.extendsObject(AbstractAjaxProcessor,{
  nowDateTime:function(){
    return gs.nowDateTime();}});
```

Toggle the timer field by field name

The following client script toggles the timer field based on a particular field name.

```
function toggleTimerByFieldName(fieldName){
 //Step 1: Find the timer object
 //timeObjectName: the timer objects name as it would normally
 be referenced
 //timeObjectHidden: the hidden input node in the field td
 //timeObjectParent: the parent td node containing the field and
 it's constituent nodes
 //timeObjectFields: anchor tag with onclick to stop timer

 var timeObjectName = fieldName;
 var timeObjectHidden = gel(timeObjectName);

 //Step 2: simulate click stop button
 var timeObjectParent;
 var timeObjectFields;

 //verify that we got the correct object
 if(timeObjectHidden.type=="hidden"){

    //Get Parent td node
    timeObjectParent = timeObjectHidden.parentNode;

    //Get input fields
    timeObjectFields =
 timeObjectParent.getElementsByTagName("input");
```

```
    //simulate click of stop button
    var timerTestString ="paused";
    var timerImg;

    //loop through input objects looking for the pause timer
 object
    for(var elIt=0; elIt < timeObjectFields.length; elIt++){
      if(timeObjectFields[elIt].id.match(timerTestString)){
        if(timeObjectFields[elIt].value=="false"){
          timeObjectFields[elIt].value="true";
          timerImg =
 timeObjectParent.getElementsByTagName("img")[0];
          timerImg.src="images/timer_start.gifx";}
      elseif(timeObjectFields[elIt].value=="true"){
          timeObjectFields[elIt].value="false";
          timerImg =
 timeObjectParent.getElementsByTagName("img")[0];
          timerImg.src="images/timer_stop.gifx";}}}}}
```

Modify GlideDateTime field value

The following example uses a server-side script to access a *GlideDateTime* field.

The following server-side script example shows how to modify values using the *GlideDateTime* API. The same concept also applies to the *GlideDate* object.

```
//You first need a GlideDateTime object
//this can be from instantiating a new object "var gdt = new
 GlideDateTime()"
//or getting the object from a GlideDateTime field
//getting the field value (for example: var gdt =
 current.start_date)
//only returns the string value, not the object
//to get the object use var gdt =
 current.start_date.getGlideObject();
//now gdt is a GlideDateTime object
var gdt = current.start_date.getGlideObject();

//All methods can use negative values to subtract intervals

//add 1 hour (60 mins * 60 secs)
gdt.addSecondsLocalTime(3600);

//add 1 day
gdt.addDaysLocalTime(1);

//subtract 1 day
gdt.addDaysLocalTime(-1);

//add 3 weeks
gdt.addWeeksLocalTime(3);

//subtract 6 months.
gdt.addMonthsLocalTime(-6);
```

```
//add 1 year, representing the date and time using the UTC
 timezone instead of the local user's timezone.
gdt.addYearsUTC(1);
```

Approval assignment scripts

This is a searchable version of the useful approval and assignment scripts.

Warning:
The customization described here was developed for use in specific instances, and is not supported by Now Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

For see Viewing my approvals.

Assign a group for ESS requests

The following Assignment Rulescript automatically assigns a group for all ESS Requests.

```
if(current.opened_by.roles==""){

 current.assignment_group.setDisplayValue('Network');
    current.update();}
```

Assign catalog item to group based on delivery plan task

This assignment rule assigns a service catalog item to the database group if it uses a delivery plan that has a catalog task assigned to the desktop group.

```
//Return catalog items that have no group but do have a
 delivery plan assigned
var ri =new GlideRecord("sc_cat_item");
ri.addQuery("group","=",null);
ri.addQuery("delivery_plan","!=",null);
ri.query();
while(ri.next()){
    gs.log("Found an item");
    //Return tasks that point to the same delivery plan
 as the above item
    var dptask =new
 GlideRecord("sc_cat_item_delivery_task");

 dptask.addQuery("delivery_plan","=",ri.delivery_plan);
    dptask.query();while(dptask.next()){
        gs.log("Found a task");var gp =
 dptask.group.getDisplayValue();
        gs.log(gp);//If the task is assigned to
 desktop, assign the item's group to desktop
        if(dptask.group.getDisplayValue()=="Desktop"){
            ri.group.setDisplayValue("Desktop");
            gs.log("updating "+ ri.getDisplayValue());
            ri.update();break;}}}
```

Assign items with one task

This assignment rule automatically assigns any catalog items with only one task associated to a particular group.

```
//Get the catalog item for the current requested item
var scCatItem =new GlideRecord("sc_cat_item");
if(scCatItem.get('sys_id', current.cat_item)){
// If the catalog item already has an assignment
 group or if using workflow we don't need to make an
 assignment
  if(!scCatItem.delivery_plan.nil()&&
 scCatItem.group.nil()){
        var dpTask =new
 GlideRecord("sc_cat_item_delivery_task");

 dpTask.addQuery("delivery_plan","=",scCatItem.delivery
_plan);
        dpTask.query();
        if(dpTask.getRowCount()==1&& dpTask.next()){
        // Check that there is only 1 record in the
 GlideRecord
            dpTask.group;}}}
```

Assignment based on workload

Type: Business Rule.

Description: Populate the assigned to based on the assignment group member who has the least amount of active incidents.

Parameters:

- order: >1000 if you want to execute after assignment rules
- condition: current.assigned_to == '' && current.assignment_group != ''
- when: before, insert/update

```
var assignTo = getLowestUser();
gs.addInfoMessage("assigning to is "+ assignTo);
current.assigned_to= assignTo;

function getLowestUser(){
  var userList =new Array();
  var cg =new GlideRecord('sys_user_grmember');
  cg.addQuery('group', current.assignment_group);
  cg.query();
  while(cg.next()){
    var tech = cg.user.toString();
    var cnt = countTickets(tech);
    gs.addInfoMessage("Tech counts "+ cg.user.name+' '+
 cnt +" "+ tech);
    userList.push({ sys_id: tech,name: cg.user.name,
 count: cnt });}

  for(var i=0; i < userList.length; i++){
    gs.addInfoMessage(userList[i].sys_id+" "+
 userList[i].name+" "+ userList[i].count);}
  userList.sort(function(a, b){
    gs.addInfoMessage("Sorting: "+ a.sys_id+"("+
 a.count+");
    "+ b.sys_id+"("+ b.count+")");
    return a.count- b.count;});
```

```
  if(userList.length<=0)return"";
  return userList[0].sys_id;}

function countTickets(tech){
  var ct =new GlideRecord('incident');
  ct.addQuery('assigned_to',tech);
  ct.addQuery('active',true);
  ct.query();
  return ct.getRowCount();}
```

Run assignment rules when category is changed

Type: Client script.

Table: Incident.

Description: This example is an onChange client script on the category field within Incident. Note: this script used to use synchronous AJAX (asynchronous behavior is specified by the third parameter of the ajaxRequest call). The implementation below uses asynchronous AJAX. The drawback of using the synchronous version is that a network response problem could cause the browser to hang.

```
// Make an AJAX request to the server to get who this
 incident would be
// assigned to given the current values in the record.
 This runs the assignment
// rules thathave been defined in System Policy and
 returns the assigned_to and
// the assignment_group

function onChange(control, oldValue, newValue,
 isLoading){
  if(isLoading){return;
  // No change, do not do anything
  }

   // Construct the URL to ask the server for the
 assignment
  var url
 ="xmlhttp.do?sysparm_processor=AJAXAssignment&sys_targ
et=incident";
  var uv = gel('sys_uniqueValue');
  if(uv){
        url +="&sys_uniqueValue="+ uv.value;}
  // Make the AJAX request to the server and get the
 response
  var serial = g_form.serialize();
  // get all values currently assigned to the incident
  var response = ajaxRequest(url, serial,true,
 responseFunc);}

// This callback function handles the AJAX response.
function responseFunc(response){
  varitem=
 response.responseXML.getElementsByTagName("item")[0];
  // Process the item returned by the server
  if(item){
```

```
 // Get the assigned_to ID and its display value and
put them on the form
   varname=item.getAttribute("name");
   var name_label =item.getAttribute("name_label");
   if(name_label &&name){
     g_form.setValue('assigned_to',name, name_label);}
   else{
     g_form.setValue('assigned_to','','');}
   // Get the assignment_group ID and its display
value and put then on the form
   var group =item.getAttribute("group");
   var group_label =item.getAttribute("group_label");
   if(group_label && group){
     g_form.setValue('assignment_group', group,
group_label);}
   else{
     g_form.setValue('assignment_group','','');}}}
```

Custom approval UI macro

For information on creating a custom approval UI macro, see UI macros.

Scheduling script use cases

A business rule script specifies the actions that the business rule takes. Scripts commonly include predefined global variables to reference items in your system, such as the current record. Global variables are available to all business rules.

Warning:
The customization described here was developed for use in specific instances, and is not supported by Now Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

Calculate duration given a schedule

Type: Before update/insert business rule.

Description: A business duration calculates the open to close duration on an incident based on the particular Creating and using schedules. If there is no schedule specified, the script will simply use the first schedule returned by the query.

Script example:

The example below sets the resolved duration when the incident state moves to resolved.

```
var gr_rec = new GlideRecord('incident');
gr_rec.get('ed92e8d173d023002728660c4cf6a7bc');
if (gr_rec.incident_state == 6) {
var dur = calcDurationSchedule(gr_rec.opened_at,
 gr_rec.sys_updated_on);
}

function calcDurationSchedule(start, end){
  // Get the user
  var usr = new GlideRecord('sys_user');
  usr.get(gs.getUserID());
  // Create schedule - pass in the sys_id of your standard work
 day schedule and pass in the users timezone
```

```
  var sched = new
 GlideSchedule('08fcd0830a0a0b2600079f56b1adb9ae',usr.time_zon
e);
  // Get duration based on schedule/timezone
  return(sched.duration(start.getGlideObject(),
 end.getGlideObject()));
}
```

Check upcoming termination dates

Type: Scheduled script.

Description: This script checks nightly for termination dates on contracts coming up in 90, 50, or 10 days (depending on the contract duration field).

Script example:

```
function contractNoticeDue(){
  var now_GR = new GlideRecord("contract");
  now_GR.addQuery("u_contract_status","Active");
  now_GR.query();
  while(now_GR.next()){
    if((now_GR.u_termination_date<=
 gs.daysAgo(-90))&&(now_GR.u_contract_duration=="Long")){
      now_GR.u_contract_status="In review";}
    elseif((now_GR.u_termination_date<=
 gs.daysAgo(-50))&&(now_GR.u_contract_duration=="Medium")){
      now_GR.u_contract_status="In review";}
    elseif((now_GR.u_termination_date <=
 gs.daysAgo(-10))&&(now_GR.u_contract_duration=="Short")){
      now_GR.u_contract_status="In review";}
    now_GR.update();
  }
}
```

Use scripts in business rules to accomplish common tasks such as:

- Comparing two date fields.
- Parsing XML payloads.
- Aborting a database action in a business rule.

With scripts, you can also:

- Specify the operation that triggers the business rule.
- Use the scratchpad with display business rules to change form values just before a user loads the form.
- Use the OR condition like you would in a condition builder.

You can also utilize the system's scripting functionality available for server-side scripts.

You can use options on the Business Rules form to build conditions, set field values, and display alert messages without needing to write a script.

Server-side script use cases

Use cases for server-side scripts include logging output, getting user objects, and modifying date/time values.

Accessing the workflow scratchpad from business rules

A catalog item has been requested, and the attached workflow contains a run script activity that populates a value in the scratchpad. From a business rule running on the requested item, you want to retrieve or set scratchpad values.

Prerequisites

Role required: admin.

Name: Access Workflow Scratchpad from Business Rules.

Type: Business Rule.

Table: sc_req_item (Requested Item).

Description: A catalog item has been requested, the attached workflow contains a run script activity that populates a value in the scratchpad. From a business rule running on the requested item, you want to retrieve or set scratchpad values.

Parameters: n/a.

Script:

```
//the run script activity sets a value in the scratchpad
workflow.scratchpad.important_msg = "scratch me";

//get the workflow script include helper
var workflow = new Workflow();

//get the requested items workflow context
//this will get all contexts so you will need to get the proper
 one if you have multiple workflows for a record
var context = workflow.getContexts(current);
//make sure we have a valid context
if (context.next()) {
  //get a value from the scratchpad
  var msg = context.scratchpad.important_msg;
  //msg now equals "scratch me", that was set in the run script
 activity

  //add or modify a scratchpad value
  context.scratchpad.status = "completed";
  //we need to save the context record to save the scratchpad
  context.update();
}
```

Assign a catalog item to a group based on a delivery plan task

Assign a service catalog item to the database group if it uses a delivery plan that has a catalog task that is assigned to the desktop group.

Prerequisites

Role required: admin

Warning:
The customization described here was developed for use in specific instances, and is not supported by Now Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

Name: Assign Catalog Item to Group Based on Delivery Plan Task.

Type: Assignment Rule.

Description: This assignment rule assigns a service catalog item to the database group if it uses a delivery plan that has a catalog task assigned to the desktop group.

Script:

```
//Return catalog items that have no group but do have a delivery
 plan assigned var ri  = new GlideRecord ( "sc_cat_item" ) ;
ri.addQuery("group", "=", null);
ri.addQuery("delivery_plan", "!=", null);
ri.query();
while(ri.next()) {
    gs.log("Found an item");
    //Return tasks that point to the same delivery plan as the
 above item
    var dptask = new GlideRecord("sc_cat_item_delivery_task");
    dptask.addQuery("delivery_plan", "=", ri. delivery_plan);
    dptask.query();
    while(dptask.next()) {
        gs.log("Found a task");
        var gp = dptask.group.getDisplayValue();
        gs.log(gp);
        //If the task is assigned to desktop, assign the item's
 group to desktop
        if (dptask.group.getDisplayValue() == "Desktop") {
            ri.group.setDisplayValue("Desktop");
            gs.log("updating " + ri.getDisplayValue());
            ri.update();
            break; } } }
```

Calculating durations

Often you may need to provide users with a way to specify when a task or process is due. Using the DurationCalculator script include, you can calculate the due date using either a simple duration or relative duration.

Typically, setting a due date requires that you calculate work time rather than the total time. Only the part of the day when work is performed is considered when determining the due date. For example, a task is due in 10 hours, but is restricted to a business day schedule. If the work starts at 10am on Monday, it is due on Tuesday at 12pm as calculated below.

```
10am-5pm on Monday (6 hours) + 8am-12pm on Tuesday (4 hours)
```

For information on schedules, which you can use as inputs to DurationCalculator methods, see Creating and using schedules.

This script demonstrates how to use DurationCalculator to compute a due date.

```
/**
* Demonstrate the use of DurationCalculator to compute a due date.
```

```
*
* You must have a start date and a duration. Then you can compute a
* due date using the constraints of a schedule.
*/

gs.include('DurationCalculator');
executeSample();

/**
* Function to house the sample script.
*/function executeSample(){

	// First we need a DurationCalculator object.var dc =new DurationCalculator();

	// ---------- No schedule examples ----------

	// Simple computation of a due date without using a schedule. Seconds// are added to the start date continuously to get to a due date.
	dc.setStartDateTime("5/1/2012");if(!dc.calcDuration(2*24*3600)){// 2 days
	 gs.log("*** Error calculating duration");return;}
	gs.log("calcDuration no schedule: "+ dc.getEndDateTime());// "2012-05-03 00:00:00" two days later

	// Start in the middle of the night (2:00 am) and compute a due date 1 hour in the future// Without a schedule this yields 3:00 am.
	dc.setStartDateTime("5/3/2012 02:00:00");if(!dc.calcDuration(3600)){
		gs.log("*** Error calculating duration");return;}
	gs.log("Middle of night + 1 hour (no schedule): "+ dc.getEndDateTime());// No scheduled start date, just add 1 hour


	// ---------- Add a schedule to the date calculator ----------
	addSchedule(dc);

	// Start in the middle of the night and compute a due date 1 hour in the future.// Since we start at 2:00 am the computation adds the 1 hour from the start// of the day, 8:00am to get to 9:00am
	dc.setStartDateTime("5/3/2012 02:00:00");if(!dc.calcDuration(3600)){//
		gs.log("*** Error calculating duration");return;}
	gs.log("Middle of night + 1 hour (with 8-5 schedule): "+
 dc.getEndDateTime());// 9:00 am

	// Start in the afternoon and add hours beyond quiting time. Our schedule says the work day// ends at 5:00pm, if the duration extends beyond that, we roll over to the next work day.// In this example we are adding 4 hours to 3:00pm which gives us 10:00 am the next day.
	dc.setStartDateTime("5/3/2012 15:00:00");if(!dc.calcDuration(4*3600)){//
		gs.log("*** Error calculating duration");return;}
	gs.log("Afternoon + 4 hour (with 8-5 schedule): "+ dc.getEndDateTime());// 10:00 am.

	// This is a demo of adding 2 hours repeatedly and examine the result. This// is a good way to visualize the result of a due date calculation.
```

```
    dc.setStartDateTime("5/3/2012 15:00:00"); // for(var i=2; i<24;
i+=1){if(!dc.calcDuration(i*3600)){//
            gs.log("*** Error calculating duration");return;}
        gs.log("add "+ i +" hours gives due date: "+
 dc.getEndDateTime());}

    // Setting the timezone causes the schedule to be interpreted in the specified
timezone.// Run the same code as above with different timezone. Note that the 8 to 5 workday
is// offset by the two hours as specified in our timezone.
    dc.setTimeZone("GMT-2");
    dc.setStartDateTime("5/3/2012 15:00:00");for(var i=2; i<24;
 i+=1){if(!dc.calcDuration(i*3600)){//
            gs.log("*** Error calculating duration");return;}
        gs.log("add "+ i +" hours gives due date (GMT-2): "+
 dc.getEndDateTime());}}

/**
* Add a specific schedule to the DurationCalculator object.
*
* @param durationCalculator An instance of DurationCalculator
*/function addSchedule(durationCalculator){// Load the "8-5 weekdays
excluding holidays" schedule into our duration calculator.var scheduleName ="8-5 weekdays
excluding holidays";var grSched =new GlideRecord('cmn_schedule');
    grSched.addQuery('name', scheduleName);
    grSched.query();if(!grSched.next()){
        gs.log('*** Could not find schedule "'+ scheduleName +'"');return;}

 durationCalculator.setSchedule(grSched.getUniqueValue(),"GMT");}
```

Simple duration vs relative duration

How much work is required to complete a task can be expressed as a "relative duration".

Relative duration determines the expected due date and time relative to the starting time. Examples of relative durations include "Next business day by 4pm", or "2 business days by 10:30am".

To calculate a relative duration, the calendar and time zone must be considered to determine what "next business day" means since it is the calendar that defines which days are valid work days and the time zone will affect the result as well. As an example, consider "Next business day by 4pm":

- If it is Monday at 12pm: Next business day by 4pm => Tuesday at 4pm
- If it is Friday at 2pm: Next business day by 4pm => the following Monday at 4pm

Note:
Next business day is often defined by a starting day and time. For example, "next business day at 4pm if before 2pm" indicates that if the current time is after 2pm on a business day, then "Next business day" really means 2 business days since today does not count.

For more information on relative durations, see Define a relative duration.

Calculating a simple duration

This business rule and script example demonstrate how to calculate a simple duration.

```
var dur =new DurationCalculator();
dur.setSchedule(current.schedule);
```

```
dur.setStartDateTime("");

if(current.duration_type==""){

 dur.calcDuration(current.duration.getGlideObject().getNumericVa
lue()/1000);}else{
         dur.calcRelativeDuration(current.duration_type);}

    current.end_date_time= dur.getEndDateTime();
    current.work_seconds= dur.getSeconds();
```

This script demonstrates how to use DurationCalculator to calculate a simple duration.

```
/**
 * Sample script demonstrating use of DurationCalculator to
 compute simple durations
 *
 */

gs.include('DurationCalculator');
executeSample();

/**
 * Function to house the sample script.
 */
function executeSample(){

    // First we need a DurationCalculator object.
    var dc =new DurationCalculator();

    // Compute a simple duration without any schedule. The
 arguments
    // can also be of type GlideDateTime, such as fields from a
 GlideRecord.
    var dur = dc.calcScheduleDuration("5/1/2012","5/2/2012");
    gs.log("calcScheduleDuration no schedule: "+ dur);
    // 86400 seconds (24 hours)

    // The above sample is useful in limited cases. We almost
 always want to
    // use some schedule in a duration computation, let's load a
 schedule.
    addSchedule(dc);

    // Compute a duration using the schedule. The schedule
    // specifies a nine hour work day. The output of this is
 32400 seconds, or
    // a nine hour span.
    dur = dc.calcScheduleDuration("5/23/2012 12:00","5/24/2012
 12:00");
    gs.log("calcScheduleDuration with schedule: "+ dur);
    // 32400 seconds (9 hours)

    // Compute a duration that spans a weekend and holiday. Even
 though this
    // spans three days, it only spans 9 work hours based on the
 schedule.
```

```
    dur = dc.calcScheduleDuration("5/25/2012 12:00","5/29/2012
 12:00");
    gs.log("calcScheduleDuration with schedule spaning holiday:
 "+ dur);
    // 32400 seconds (9 hours)

    // Use the current date time in a calculation. The output of
 this is
    // dependent on when you run it.
    var now =new Date();
    dur = dc.calcScheduleDuration("5/15/2012",new
 GlideDateTime());
    gs.log("calcScheduleDuration with schedule to now: "+ dur);
    // Different on every run.}

/**
 * Add a specific schedule to the DurationCalculator object.
 *
 * @param durationCalculator An instance of DurationCalculator
 */
function addSchedule(durationCalculator){
   //  Load the "8-5 weekdays excluding holidays" schedule into
 our duration calculator.
   var scheduleName ="8-5 weekdays excluding holidays";
   var grSched =new GlideRecord('cmn_schedule');
   grSched.addQuery('name', scheduleName);
   grSched.query();if(!grSched.next()){
        gs.log('*** Could not find schedule "'+ scheduleName
 +'"');
        return;}
    durationCalculator.setSchedule(grSched.getUniqueValue());}
```

Calculating a relative duration

An example of a relative duration calculation script.

This script calculates the relative duration for "Next day at 4pm if after 10am":

```
// Next day at 4pm if before 10am
var days =1;
if(calculator.isAfter(calculator.startDateTime,"10:00:00"))
      days++;

calculator.calcRelativeDueDate(calculator.startDateTime,
 days,"16:00:00");
```

This script demonstrates how to use DurationCalculator to calculate a relative duration.

```
/**
 * Sample use of relative duration calculation.
 *
 */

gs.include('DurationCalculator');
executeSample();

/**
 * Function to house the sample script.
```

```
 */
function executeSample(){

    // First we need a DurationCalculator object. We will also
 use
    // the out-of-box relative duration "2 bus days by 4pm"
    var dc =new DurationCalculator();
    var relDur ="3bf802c20a0a0b52008e2859cd8abcf2";
    // 2 bus days by 4pm if before 10am
    addSchedule(dc);

    // Since our start date is before 10:00am our result is two
 days from
    // now at 4:00pm.
    dc.setStartDateTime("5/1/2012 09:00:00");
    if(!dc.calcRelativeDuration(relDur)){
        gs.log("*** calcRelativeDuration failed");
        return;}
    gs.log("Two days later 4:00pm: "+ dc.getEndDateTime());

    // Since our start date is after 10:00am our result is three
 days from
    // now at 4:00pm.
    dc.setStartDateTime("5/1/2012 11:00:00");
    if(!dc.calcRelativeDuration(relDur)){
        gs.log("*** calcRelativeDuration failed");
        return;}
    gs.log("Three days later 4:00pm: "+ dc.getEndDateTime());}

/**
 * Add a specific schedule to the DurationCalculator object.
 *
 * @param durationCalculator An instance of DurationCalculator
 */
function addSchedule(durationCalculator){
  //  Load the "8-5 weekdays excluding holidays" schedule into
 our duration calculator.
  var scheduleName ="8-5 weekdays excluding holidays";
  var grSched =new GlideRecord('cmn_schedule');
  grSched.addQuery('name', scheduleName);
  grSched.query();
  if(!grSched.next()){
        gs.log('*** Could not find schedule "'+ scheduleName
 +'"');
        return;}

 durationCalculator.setSchedule(grSched.getUniqueValue(),"GMT"
);}
```

How to implement a relative duration

You can implement a relative duration by creating the cmn_relative_duration table and the *DurationCalculator* script include.

Before you begin

Role required: admin

Procedure

1. Create the cmn_relative_duration table.
2. Create the DurationCalculator script include.
3. Create a sample relative duration entry (for example, "Next business day by 4pm").
4. Add the needed fields to SLA tables to support relative durations.
5. Modify duration calculation for SLAs.
6. Modify SLA Percentage timer calculation for SLAs (this must use work_seconds).
7. Add schedule fields to the Workflow: Schedule and Timezone (selected based on the field from workflow table).
8. Add duration support fields to the Workflow Task activity.
9. Implement duration calculation script for the task activity.

The relative duration table and the DurationCalculator methods

The cmn_relative_duration table supports the definition of a due date as either a duration of time or a relative duration.

This table consists of two fields: "name" and "script." The "script" field contains the relative duration calculation script. This script includes the "calculator" variable, which is used to calculate the due date.

The DurationCalculator script include can be used to perform the duration calculations. The following are methods that are available in this script include.

DurationCalculator script include table

Method	Description
setSchedule(String schedID, [String timezone])	Sets the schedule and time zone to be used for calculating the due date.
setStartDateTime(GlideDateTime start)	Sets the start time for the duration calculations. If 'start' is blank, uses current date/time.
calcDuration(int seconds)	Calculates the end date and time. Upon completion the this.endDateTime and this.seconds properties will be set to indicate the results of the calculation.
calcRelativeDuration(String relativeDurationID)	Calculates the duration using the specified relative duration script. Upon completion the this.endDateTime and this.seconds properties will be set to indicate the results of the calculation.
getEndDateTime()	Gets the this.endDateTime property that was set by calcDuration/calcRelativeDuration indicating the end date and time for the duration.
getSeconds()	Gets the this.seconds property that was set by calcDuration/calcRelativeDuration indicating the total number of seconds of work to be performed for the duration.

DurationCalculator script include table (continued)

Method	Description
	Note: This is the total work time, not the total time between start and end times and may be used to determine percentages of the work time
getTotalSeconds()	Gets the this.totalSeconds property that was set by calcDuration/calcRelativeDuration indicating the total number of seconds between the start and end times of the duration.

The following functions are used in relative duration scripts:

Relative duration script functions

Function	Description
boolean isAfter(GlideDateTime dt, String time)	Is 'time' of day after the time of day specified by 'dt'? dt, if blank, uses current date/time. time is in "hh:mm:ss" in 24-hour format.
calcRelativeDueDate(GlideDateTime start, int days, String endTime)	Calculates the due date starting at 'start' and adding 'days' using the schedule and time zone. When we find the day that the work is due on, set the time to 'endTime' of that day. Upon completion, this.endDateTime and this.seconds properties will be set to indicate the results of the calculation. If endTime is blank, use end of the ending work day.

Get a user object

In a business rule or other server script, the `gs.getUser()` method returns a user object. The user object is an internal representation of the currently logged in user and provides information about the user and various utility functions.

About this task

For a list and description of the available scoped methods for the user object, see GlideUser.

Procedure

1. Retrieve the current user.

```
var myUserObject = gs.getUser()
```

2. Use the `getUserByID` method to fetch a different user using the `user_name` field or `sys_id` on the target record.
   For example:

```
var ourUser = gs.getUser();
gs.print(ourUser.getFirstName()); //print the first name of the
 user you are currently logged in as
newUser = ourUser.getUserByID(<user_sys_id>); //fetch a
 different user, using the sys_id of the target user record.
```

```
gs.print(newUser.getFirstName()); //first name of the user you
 fetched above
gs.print(newUser.isMemberOf('Capacity Mgmt'));
```

Log output

GSLog is a script include that simplifies script logging and debugging by implementing levels of log output, selectable by per-caller identified sys_properties values.

Log level

Logs can be at the level of debug, info, notice, warning, err, or crit (after BSD syslog.h and followers). The default logging level is notice, so levels should be chosen accordingly.

Where to use

Use for any server-side script where you want to implement event logging.

For the API reference, see GSLog() .

For more information, see Debugging scripts

Modify a GlideDateTime field value

This example demonstrates how to modify a *GlideDateTime* field value using a server-side script.

Given a *GlideDateTime* field or script object, show a variety of ways to easily modify value. The same concept also applies to the *GlideDate* object.

Note:
The following script is only intended for global applications.

```
//You first need a GlideDateTime object
//this can be from instantiating a new object "var gdt = new
 GlideDateTime()"
//or getting the object from a GlideDateTime field
//getting the field value (for example: var gdt =
 current.start_date) only returns the string value, not the
 object
//to get the object use var gdt =
 current.start_date.getGlideObject();
//now gdt is a GlideDateTime object
var gdt = current.start_date.getGlideObject();

//All methods can use negative values to subtract intervals

//add 1 hour (60 mins * 60 secs)
gdt.addSeconds(3600);

//add 1 day
gdt.addDaysLocalTime(1);

//subtract 1 day
gdt.addDaysLocalTime(-1);

//add 3 weeks
gdt.addWeeksLocalTime(3);
```

```
//subtract 6 months
gdt.addMonthsLocalTime(-6);

//add 1 year, representing the date and time using the UTC
 timezone instead of the local user's timezone.
gdt.addYearsUTC(1);

//set the value of the GlideDateTime object to the current
 session timezone/format
GlideSession.get().setTimeZoneName('US/Eastern');
gdt.setDisplayValue('2018-2-28 00:00:00');
gs.info('In ' + GlideSession.get().getTimeZoneName() + ": " +
 gdt.getDisplayValue());
```

See also:

- GlideDateTime
- GlideDate - Global
- GlideDate - Scoped
- GlideDateTime - Global
- GlideDateTime - Scoped
- GlideTime - Scoped

Using custom queues to process events

You can use custom queues for applications that create a large volume of events or events that take a long time to process. This task shows how to create a custom queue, its monitoring process, and use a script to send events to the queue.

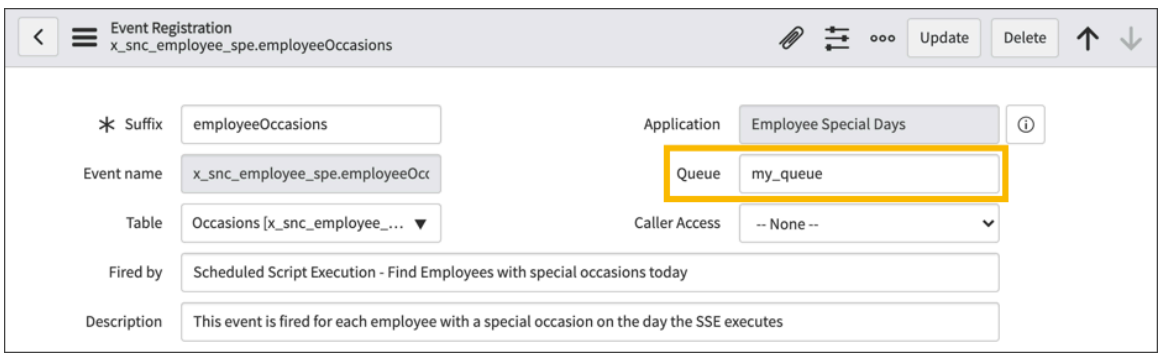
Before you begin

Role required: admin

Note:
This information is for advanced users who understand event processing.

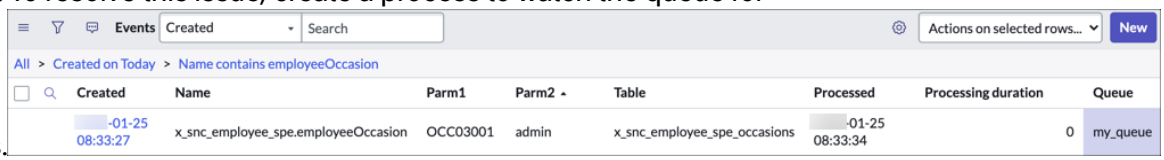
Procedure

1. Navigate to **System Policy** > **Events** > **Registry**.
2. Select the event for which you want to create a custom queue.
   The **Event Registration** form displays.
3. Populate the **Queue** field for the event in the Event Registry.
   Use only lowercase letters, no spaces, and no special characters except underscore (_).

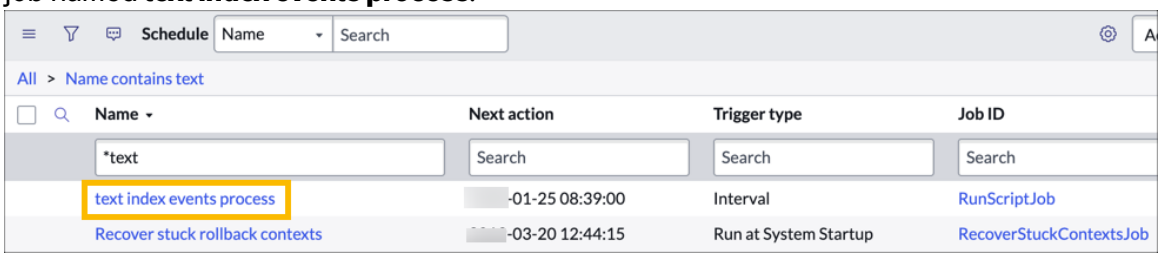


4. Click **Submit**.
   A new event is listed in the Events [sysevent] table.

   In the following example, when the employeeOccasion event is generated, the event is added to my_queue. The events are stuck in the queue. To resolve this issue, create a process to watch the queue for events.



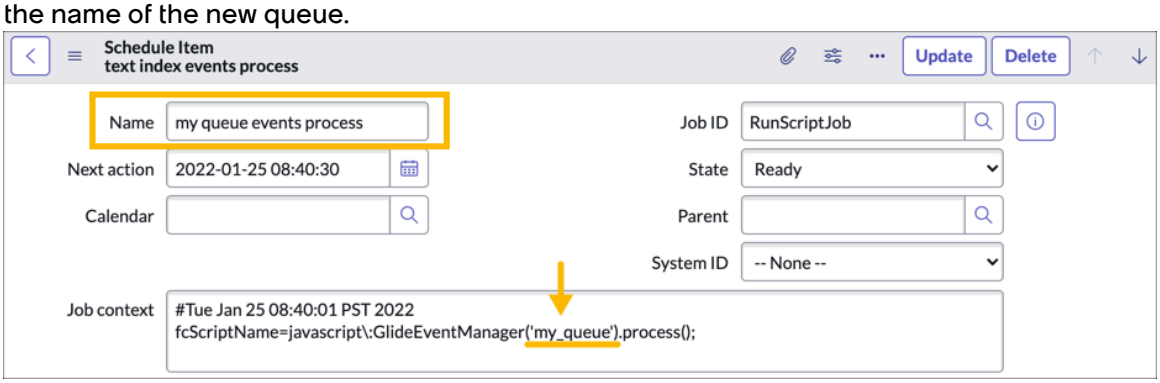
5. Navigate to **System Scheduler** > **Scheduled Jobs** > **Scheduled Jobs** and open the scheduled job named **text index events process**.



6. Click the additional actions menu icon)--> and select **Insert and Stay** to create a copy of **text index events process**.

> **Important:**
> Be sure to copy the job and not overwrite the **text index events process** Scheduled Job.

7. In the copied schedule item, change the value in the **Name** field.
8. In the **Job context** field, replace the value for the `GlideEventManager()` parameter with the name of the new queue.



   The queue monitoring process looks for and processes events in the example **my_queue** event queue.

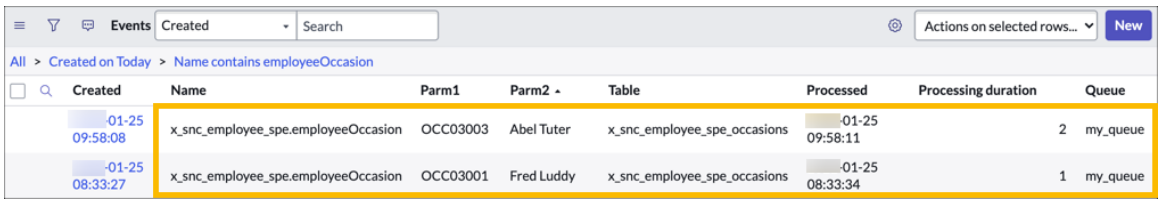
9. Use the `gs.eventQueue()` method's fifth parameter to send events to the custom queue.

   The following code shows how to send an event to the my_queue custom queue.

```
gs.eventQueue('x_60157_employee_spe.employeeOccasion',
 todaysOccasions, todaysOccasions.number,
 todaysOccasions.u_employee.name, 'my_queue');
```

Note:
If an event is in the **Event Registry** and no queue name is provided to `gs.eventQueue`, the queue from the **Event Registry** is still assigned to the event. For example, `gs.eventQueue('x_60157_employee_spe.employeeOccasion')` still associates the event with `my_queue`. If the queue name is provided in the `gs.eventQueue()` call, the queue takes priority.

You can verify that the event called was processed by checking the **Events** [sysevent] table.



Validation script use case - Date and time

To validate the input of all date/time fields, you can use the following in a validation script (**System Definition** > **Validation Scripts**).

Because the date/time format is hard-coded in this script, it must match your instance's date/time format. If your instance's date/time format changes, you must update your validation script.

Set the validation script's type to "`glide_date_time`". Then, with this validation script, if a user enters an incorrect format in a date/time field, they will receive an error message.

```
function validate (value){
 if (!value) {
    return true;
 }

 return (getDateFromFormat(value, 'yyyy-MM-dd HH:mm:ss') != 0);
}
```

For a comprehensive use case, see Restricting record access.

Creating custom UI Pages and UI macros

Use UI pages to create custom pages for an application and UI macros for custom controls or interfaces.

Every UI Page is a Jelly template. Jelly turns XML into executable code. A UI Page works similar to how an index.html file is used in an AngularJS application. Jelly tags in the HTML field of the UI Page form contain AngularJS logic.

Creating UI macros requires knowledge of Jelly script. Review the existing UI macros for examples and suggested approaches. Those who want to build custom interfaces with JavaScript technologies should consider Service Portal as an alternative.

UI pages

UI pages can be used to create and display forms, dialogs, lists, and other UI components.

Use UI pages as widgets on dashboards. To find the UI pages, navigate to **System UI** > **UI Pages**.

This functionality requires a knowledge of HTML or Jelly. You can also create simple AngularJS applications using UI pages.

The UI page form contains the following fields:

UI page

Field	Description
Name	Name used to invoke the page via a URL (must not contain spaces).
Application	Displays the current application scope.
Description	The description of the UI page and what it's used for.
Direct	Select this check box for a direct UI page [sys_ui_page]. A direct UI page doesn't include the common HTML, CSS, and scripts. This setting requires adding custom CSS and JavaScript to use in the page.
HTML	Main component of the page where you define what is rendered when the page is shown. It can contain either static XHTML, dynamically generated content defined as Jelly, or call script includes and UI Macros. **Note:** If `GlideRecord/GlideDBQuery` is used instead of `GlideRecordSecure`, a security recommendation message displays.
Client Script	Include client-side JavaScript that runs in the browser, such as functions called by buttons. It's intended to handle any client-side processing needed, like setting focus to a field or other interactive DHTML features after a page is loaded. Client scripts for the UI page are deployed to the browser within a `<script/>` tag, so the content can similarly be defined within the HTML field. The Client Script field can be used instead to define these scripts concisely to maintain the Jelly and HTML manageability.
Processing Script	Script that runs on the server when the page is submitted. This is useful if your page has a form defined with the `<g:ui_form/>` or `<g:form/>` tags. **Note:** If `GlideRecord/GlideDBQuery` is used instead of `GlideRecordSecure`, a security recommendation message displays.
obsolete-custom-processors	**Note:** This feature is deprecated. While legacy, existing custom processors continue to be supported, creating new custom processors has been deprecated. Instead, use the Scripted REST APIs.
Related lists on the form view:	
Access Controls	View and configure access controls for the UI page. See Use access controls on UI pages for more information.
Versions	Shows all versions of the UI page. Use this list to compare versions or to revert to a previous version.

Access control

A UI page can be secured by creating an ACL with the following parameters:

- **Type**: *ui_page*
- **Operation**: *read*
- **Name**: Name of the UI page to be protected
- **Role**: User role that is allowed to access the record

When saving a new UI page, you are prompted to assign a role for access control.

Note:
An entry with the same name as the UI page is created in the Access Control table.

For details on creating an ACL rule, see Create an ACL rule.

High risk UI pages

UI Pages are considered high risk with any of the following attributes:

- Uses *GlideRecord* or *GlideDBQuery* instead of *GlideRecordSecure*.

  Note:
  This is applicable for *HTML* and *Processing* script fields and not *Client* scripts.

- Does not have a corresponding ACL configured.
- Indicates to be a public UI Page and is entered in the *sys_public* record.
- Shows a message to indicate high risk UI page.
- For instances with *glide.installation.developer* is set to **true**.
- If resource is customized content for a customer instance.

UI page access

Each UI page has a URL computed from the application scope, page name, and the `.do` file extension.

For example, to display the page called glidewindow_example on the demo system, you would navigate to `https://<instance name>.service-now.com/glidewindow_example.do`. If the page was part of a custom application called example_app, you would instead navigate to `https://<instance name>.service-now.com/x_example_app_glidewindow_example.do`.

You can also add additional parameters to a URL that can be accessed within a page's HTML section as jelly variables. That is, appending arguments to the URL as follows: `/my_test_page.do?sysparm_verbose=true` creates jelly variables called *verbose* that can be accessed as follows:

```
<j2:if test="$[!empty(sysparm_verbose)]"> <span>show extra stuff
 </span> </j2:if >
```

A common practical example of this might be retrieving a database record for display. To build a list of a user's roles, pass in a parameter with the user's sys_id. Invoke the following UI page to display a list of roles for that user with Jelly code:

```
role_select.do?sysparm_user=5137153cc611227c000bbd1bd8cd2007
```

```
<j:set var = "jvar_user_id" value = "${sysparm_user}" />

 <g:evaluate> var userRoles = new
GlideRecord('sys_user_has_role');
   userRoles.addQuery('user','${jvar_user_id}');
   userRoles.query();
 </g:evaluate>

 <select id='select_role'>
     <j:while test = "${userRoles.next()}">
         <option value = "${userRoles.sys_id}">
${userRoles.role.name} </option>
     </j:while>
 </select>
```

An exception to be careful of, though, is the reserved variable name sys_id. This variable always contains the ID of the UI page itself, regardless of what is specified in the URL. A common substitute variable name is sysparm_id.

Do not use URL parameters to load client scripts in UI pages. The system no longer evaluates scripts that are passed by URL parameter. If your implementation depends on this behavior, you can add the system property [glide.security.disable_ui_pages_sysparm_client_script] and set it to **false** to temporarily allow the evaluation of URL parameters passing scripts in UI pages.

Use access controls on UI pages

See access controls directly from the UI Page form and add role-based access control when creating or editing a UI Page record.

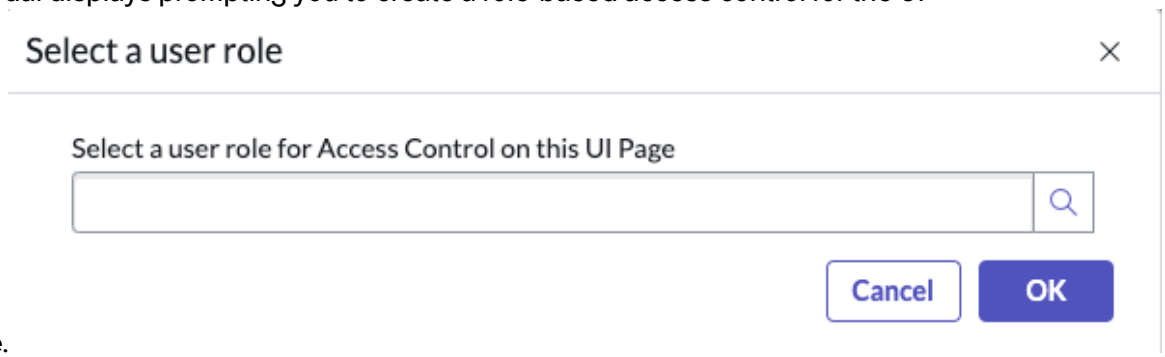
Before you begin

Access controls that have been added to an existing UI page can be accessed and edited by opening the UI page record under related links.

Role required: security_admin and admin

Procedure

1. Elevate to the **security_admin** role.
   For details on role elevation, see Elevate to a privileged role.
2. Navigate to **All** > **System UI** > **UI Pages**.
3. Select **New**.
4. Complete the form.
   See UI pages for additional information for UI field descriptions.
5. Select **Submit** or **Save**.
   A modal displays prompting you to create a role-based access control for the UI page.



6. Select a role.
7. Select **OK** to assign the role.
   You are returned to the UI pages list.

Add, edit, or view access controls for an existing UI page:

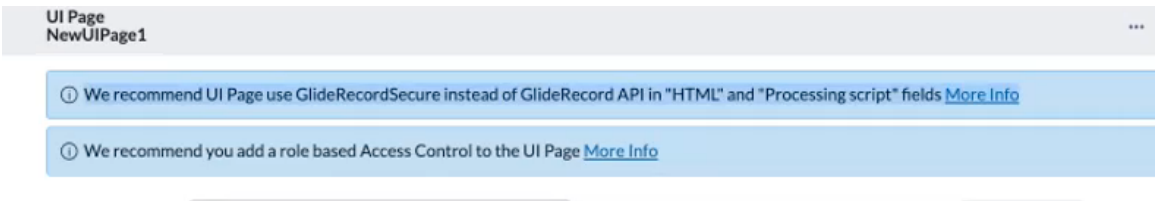
8. Open a UI page from the UI pages table.
9. View the **Access Controls** under the related lists.
10. Select **New** to create a new access control or select an existing entry for editing.
    The Access Control form loads. UI Page details are displayed.
11. Complete the form and assign a role to the UI page.
    For additional information on access control, see Create an ACL rule.
12. Select **Submit** for a new access control or **Update** for edits.

> **Note:**
> There can be multiple access controls per UI page.

Secure UI pages

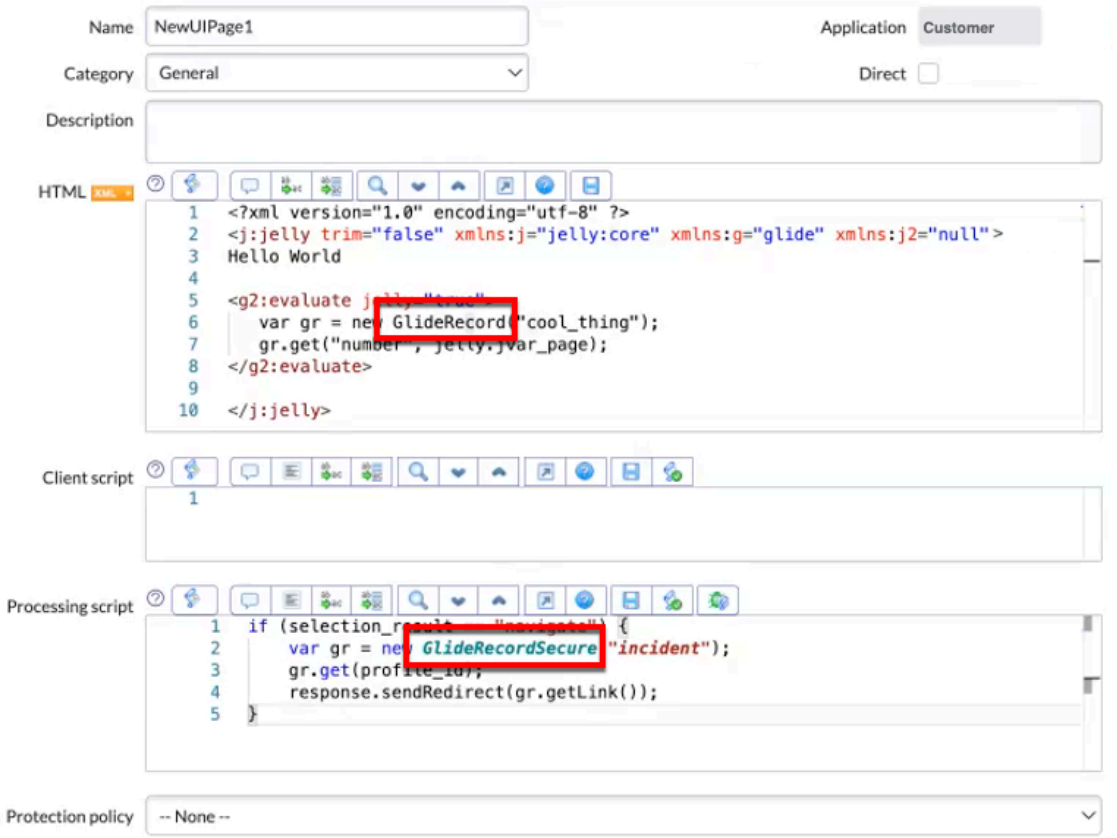
Access controls and related security messages are integrated on high risk UI Pages for increased security.

An informational message displays on high risk UI pages to inform the customer to add a role-based Access Control to the UI page.



The message displays under the following conditions:

- If an ACL (access control list) is missing
- If `GlideRecord/GlideDBQuery` is used instead of `GlideRecordSecure` in either the **HTML** or **Processing script**



sections

- If the UI page is configured as public in `sys_public`

> **Note:**
> Public UI Pages that are public or that use `GlideRecord` don't show a missing ACL warning.

See UI pages for details on high risk UI pages.

Conditions that display the security recommendations message

The message displays under the following conditions:

- All internal instances eclipse/IJ where `glide.installation.developer=true`.
- All customer scoped UI Page resources.
- Any customized UI Page when the `glide.script.ui_page.customer_scoped.security_msgs_enabled` property is set to `true`. (The default value is `true`).

Conditions that don't display the security recommendations message

The message won't display under the following conditions:

- UI Page is public.
- UI Page is in the **Global** scope.
- The *glide.script.ui_page.customer_scoped.security_msgs_enabled* is set to *false*.

Note:
To disable the role-based ACL creation prompt, set the *glide.ui_page.enable_acl_create_ux* property to **false**. The property is set to **true** by default.

UI page process scripts

If your UI page contains a form (uses the *<g:form>* tag), you can submit the form and have the process script run.

The processing script can naturally access fields on the form. For example, if your form contained the *application_sys_id* field:

```
<g:ui_form>
  <p>Click OK to run the processing script.</p>
  <g:dialog_buttons_ok_cancel ok="return true" />
  <input type="hidden" name="application_sys_id"
 value="499836460a0a0b1700003e7ad950b5da" />
</g:ui_form>
```

You can access the field using *application_sys_id*:

```
var application = new GlideRecord('hr_application');
 application.get(application_sys_id);
 application.status = "Rejected";
 application.update();
 var urlOnStack = GlideSession.get().getStack().bottom();
 response.sendRedirect(urlOnStack);
```

Important:
The preceding script is usable only with Global applications.

If you are using the UI page for a dialog, you can also reference the most recent URL on the stack using the code above and then send the response to that location. This is useful if you want to have the dialog's processing script update something and then redisplay the screen that brought up the dialog.

UI macros

UI macros are discrete scripted components administrators can add to the user interface.

UI macros are typically controls that provide inputs or information not provided by existing field types. By default, the system provides UI macros for a variety of user interface elements such as:

- All formatters
- The Service Catalog cart
- The action icons next to fields
- The action icons on forms and lists
- The widgets of a content management system
- The Orchestration activity designer

Administrators can create their own UI macros to provide custom controls or interfaces. Creating UI macros requires knowledge of Jelly script. Review the existing UI macros for examples and suggested approaches. Those who want to build custom interfaces with JavaScript technologies should consider Service Portal as an alternative.

Note:
To view available UI macros, navigate to **All** > **System UI** > **UI Macros**.

UI macro basics

UI macros can be used to build solutions that can't be built using the available catalog variable types.

Accessible UI macros

Several UI macros are available in the UI macros [ui_macros] table. These UI macros have names starting with `ui_` and emulate the behavior of a subset of standard form field types for use in UI pages. For example, the ui_date_time UI macro can act like a glide_date_time field in a UI page, providing a type-able input field with a date/time selector.

A UI macro can be included in a UI page with the <g:> Jelly tag. The following example includes the ui_date_time UI Macro, providing two optional attributes to the UI Macro:

```
<g:ui_date_time name="due_date" value="2023-11-24 08:30:00"
 onchange="checkDateValue()" />
```

The attribute values in the <g:> Jelly tag are provided to the UI macro as jvar-prefixed variables. The UI macro can use the jvar-prefixed variables in its XML.

Jelly tag attributes used as jvar-prefixed variables

Attributes	jvar-prefixed variables
name	jvar_name
value	jvar_value
onchange	jvar_onchange

Note:
These UI macros aren't intended to support all features of their corresponding form field type. In many cases, the macros are only intended for specific ServiceNow application use cases.

The `ui_`-prefixed UI macros include descriptions specifying supported attributes that can (or must) be provided in the UI page's <g:> Jelly tag. To view available UI macros, navigate to **All** > **System UI** > **UI macros**.

/

Custom UI macros

The ui_example UI macro uses three jvar-prefixed variables: jvar_name, jvar_test_attribute and jvar_laptop_type. These variables can be provided to the UI macro as name, test_attribute and laptop_type attributes in a <g:> Jelly tag in a UI page.

Creating the ui_example UI macro

Navigate to **All** > **System UI** > **UI Macros** and select **New**.

Name the macro `ui_example` and add the following script to the **XML** field:

```
<?xml version="1.0" encoding="utf-8" ?>
<j:jelly trim="true" xmlns:j="jelly:core"
 xmlns:g="glide">
    <div>name jvar: ${jvar_name}</div>
    <div>test_attribute jvar:
 ${jvar_test_attribute}</div>
    <div>laptop_type jvar: ${jvar_laptop_type}</div>
</j:jelly>
```

Using the macro in a UI page

Navigate to **All** > **System UI** > **UI Pages** and select **New**. You will be asked to select an administrator role.

Name the UI page and add the following script to the **HTML** field:

```
<?xml version="1.0" encoding="utf-8" ?>
<j:jelly trim="false" xmlns:j="jelly:core"
 xmlns:g="glide" xmlns:j2="null" xmlns:g2="null">
    <div style="text-decoration:underline">Template
 include one:</div>
    <g:ui_example name="Fred Luddy" test_attribute="I
 am an attribute" laptop_type="ThinkPad" />
    <hr/>
    <div style="text-decoration:underline">Template
 include two:</div>
    <g:ui_example name="Pat Casey" test_attribute="I am
 a different attribute" laptop_type="Macbook Pro" />
</j:jelly>
```

Checking output

In the UI page, you can click **Try It** to view the results.

Note:
If you create a macro and it doesn't display as expected in the UI page, clearing the cache might help. To clear the cache of your instance, in your browser, enter `<server_url>/cache.do`.

Related topics

- UI pages
- Jelly tags
- <g:ui_form/>
- <g:ui_input_field />
- <g:ui_checkbox/>

Calling UI macros

Administrators can call UI macros from certain record types associated with the user interface.

Calling UI macros by record type

Record type	Example
Dictionary attribute	Display an icon for a reference field:

Calling UI macros by record type (continued)

Record type	Example
	`ref_contributions=ui_macro_name`
UI page	Display something on a UI page: `<g:macro_invoke macro="ui_macro_name" />`
UI macro	Call a UI macro from another UI macro: `<?xml version= "1.0" encoding="utf-8" ?>` `<j:jelly trim= "false" xmlns:j="jelly:core"` `xmlns:g="glide" xmlns:j2="null" xmlns:g2="null">` `<g:ui_macro_name />` `<g:ui_macro_name_2 />` `</j:jelly>`

UI macro form

Each UI macro record consists of a name and an XML document written in Jelly code.

UI macro fields

Field	Description
Name	A unique and descriptive name for this macro.
Active	Select the check box to render the element as defined. Clear the check box to disable the element without deleting the code. For example, the **email_reply** macro is inactive by default.
Description	Describe the purpose of the macro and parameters passed to it.
XML	Jelly script that defines the macro.

Custom approval UI macro

This section describes how to create a custom approval UI macro.

Warning:
The customization described here was developed for use in specific instances, and is not supported by Now Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

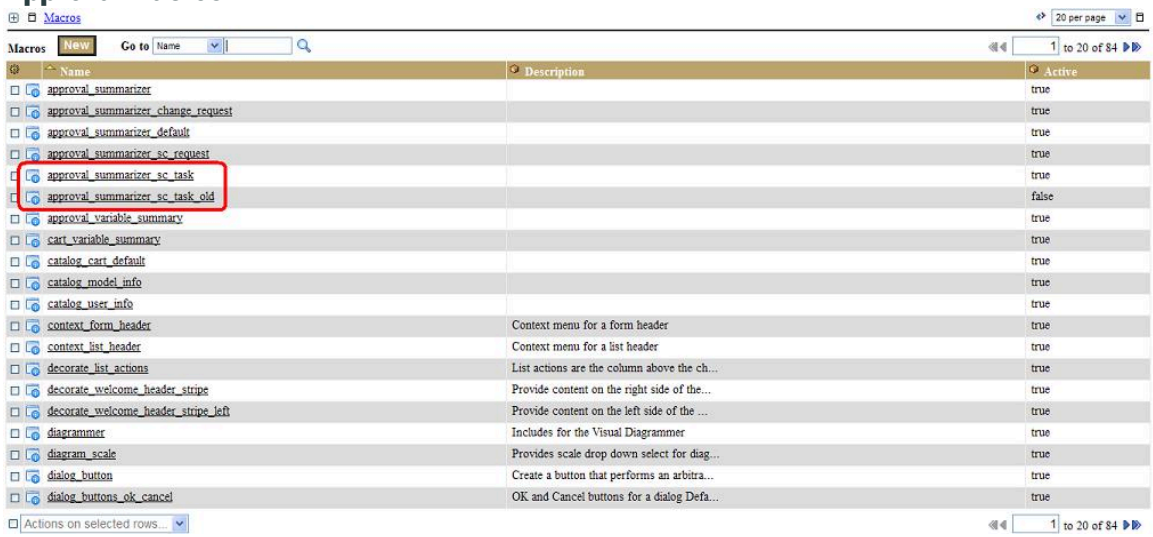
Script Name: Custom Approval UI Macro

Type: UI Macro

Description: Here is an option to get more detail out of the My Approvals view of an Execution Plan. This can be done by creating a new UI macro. Navigate to **System UI** and click **UI Macros**. First, you will need to rename the existing `approval_summarizer_sc_task` to something like `approval_summarizer_sc_task_old` and deactivate it. Then you will need to create a new one using the same name (`approval_summarizer_sc_task`). The name should

basically tell you what the macro does and what it applies to. In this case, we're replacing an existing one so we decided to re-use the existing name.

Approval macros



Then you should copy the xml script at the bottom of this article into the xml code window in the new UI macro. This is great way to give some detail to an approver when you are doing line item approvals via approval tasks within the Service Catalog Execution Plans.

The old way

This is the view you see in My Approvals when using an approval task via the old method.

My Approvals (old way)

Notice there is not much detail telling the approver what they are actually approving. You can see the short description of the task but not much around what the item is.

The new way

This is the view you will see if you use the xml script below in place of the OOB (out-of-box) UI macro.

Using this method you can see details much like the request approval. You have a link into the item ordered, a short description (which contains the ability to expand the variables from the item), price, quantity and the total price. This helps the approver in that it shows more detail. They can now see what they are actually approving.

Script:

```
<?xml version="1.0" encoding="utf-8" ?>
    <j:jelly trim="true" xmlns:j="jelly:core" xmlns:g="glide"
 xmlns:j2="null" xmlns:g2="null">
    <tr>
        <td class="label_left" width="100%">
            <label style="margin-left: 10px">
                ${gs.getMessage('Summary of Item being
 approved')}:
                <input style="visibility: hidden"
 NAME="make_spacing_ok"></input>
            </label>
        </td>
    </tr>
    <g:evaluate var="jvar_ni" expression="
        var sc_task = ${ref}.sysapproval;
        var sc_req_labels = new GlideRecord('sc_req_item');
        sc_req_labels.initialize();
        var sc_req_item = new GlideRecord('sc_req_item');
        sc_req_item.addQuery('request_item',
 sc_task.request_item.sys_id);
        sc_req_item.query();
    " />
    <tr>
        <td width="100%">
            <table width="100%">
                <tr>
                    <td class="label_left" width="150px">
                        <label style="margin-left: 10px">

 ${sc_task.request_item.request.opened_by.sys_meta.label}:
                        </label>
                    </td>
                    <td>

 ${sc_task.request_item.request.opened_by.getDisplayValue()}
                    </td>
                </tr>
                <tr>
```

```
                    <td class="label_left" width="150px">
                        <label style="margin-left: 10px">

${sc_task.request_item.request.requested_for.sys_meta.label}:
                        </label>
                    </td>
                    <td>

${sc_task.request_item.request.requested_for.getDisplayValue()}
                    </td>
                </tr>
                <tr>
                    <td class="label_left" width="150px">
                        <label style="margin-left:
10px">${gs.getMessage('Total')}:</label>
                    </td>
                    <td>
                        <g:currency_format
value="${sc_task.request_item.request.price}" />
                    </td>
                </tr>
            </table>
        </td>
    </tr>
    <j:set var="jvar_line_num" value="0" />
    <tr>
        <td width="100%">
            <table width="100%">
                <tr class="header">
                    <td colspan="2">
                        ${sc_req_labels.number.sys_meta.label}
                    </td>
                    <td>

${sc_req_labels.description.sys_meta.label}
                    </td>
                    <td>
                        ${sc_req_labels.price.sys_meta.label}
                    </td>
                    <td>
                        ${sc_req_labels.quantity.sys_meta.label}
                    </td>
                    <td>
                        ${gs.getMessage('Total')}
                    </td>
                </tr>
                <j:set var="jvar_item_price"
value="${sc_task.request_item.price *
sc_task.request_item.quantity}"/>
                <j:set var="jvar_overall_total"
value="${jvar_overall_total + jvar_item_price}"/>
                <j:set var="jvar_line_color" value="odd"/>
                <j:set var="jvar_line_num"
value="${jvar_line_num + 1}"/>
                <j:if test="${jvar_line_num % 2 == 0}">
                    <j:set var="jvar_line_color" value="even"/>
                </j:if>
```

```
                <g:evaluate var="ni" expression="
                    var smart_description =
sc_task.request_item.cat_item.short_description;
                    if (smart_description == null ||
smart_description == '' || smart_description == 'undefined')
                    smart_description =
sc_task.request_item.cat_item.name;
                "/>
                <tr class="${jvar_line_color}">
                    <td valign="top">
                        <g2:evaluate var="jvar_pop_target"
expression="$[ref].getRecordClassName()" />
                        <a class="linked" target="gsft_main"

href="sc_req_item.do?sys_id=${sc_task.request_item.sys_id}"
                            onmousemove="popListDiv(event,
'sc_req_item',
'${sc_task.request_item.sys_id}','${sysparm_view}')"
                            onmouseout="lockPopup(event)">
                            <img
src="images/icons/hover_icon.gifx" class="clsshort"/>
                        </a>
                    </td>
                    <td valign="top">
                        <a class="linked" target="gsft_main"

href="sc_req_item.do?sys_id=${sc_task.request_item.sys_id}">
                            ${sc_task.request_item.number}
                        </a>
                    </td>
                    <td valign="top" width="50%">
                        <g:call
function="variable_summary_approval.xml"

question_name="${sc_task.request_item.sys_id}"

question_help_tag="${smart_description}"

sc_req_item="${sc_task.request_item.sys_id}"
                            help_class="${jvar_line_color}"/>
                    </td>
                    <td valign="top"> <g:currency_format
value="${sc_task.request_item.price}"/> </td>
                    <td valign="top">
${sc_task.request_item.quantity}</td>
                    <td valign="top"> <g:currency_format
value="${jvar_item_price}"/></td>
                </tr>
            </table>
        </td>
    </tr>
</j:jelly>
```

Jelly tags

Use Jelly to turn XML into HTML.

Watch these introductory videos to learn about using Jelly in the ServiceNow AI Platform.

- Introducing Jelly Scripting - Part 1 (Video)
- Introducing Jelly Scripting - Part 2 (Video)
- Introducing Jelly Scripting - Part 3 (Video)

Jelly Tags

if

- Description: The `if` tag is just what it looks like, an `if` tag. This is like an `if` statement in any programming language, but keep in mind that there is no `elseif` tag and no `else` tag. If you want to create that kind of structure, try the `choose/when/otherwise` syntax.
- Parameters: `test` - The condition to evaluate in order to determine if the block will execute.
- Example:

```
<g:evaluate var="jvar_now_GR" object="true">
   var now_GR = new GlideRecord("incident");
   now_GR.addQuery("active", true);
   now_GR.query();
   now_GR;
</g:evaluate>

<j:if test="${!jvar_now_GR.hasNext()}">
   We did not find any active incidents.
</j:if>
<j:if test="${jvar_now_GR.next()}">
   We found ${jvar_now_GR.getRowCount()} active
 incidents.
</j:if>
```

while

- Description: The `while` tag does a while loop.
- Parameters: `test` - The condition to evaluate in order to determine if the statement will loop through. This should be an expression enclosed in `${}` or `$[]` that evaluates to true or false.
- Example:

```
<g:evaluate var="jvar_now_GR" object="true">
   var now_GR = new GlideRecord("incident");
   now_GR.addQuery("active", true);
   now_GR.query();
   now_GR;
</g:evaluate>

<j:while test="${jvar_now_GR.next()}">
  <a
 href="incident.do?sys_id=${jvar_now_GR.getValue('sys_
id')}">${jvar_now_GR.getValue('number')}</a>
</j:while>
```

set

- Description: The `set` tag sets a variable.
- Parameters:
  - `var` - The variable to set. Often the system prefixes these variables with `jvar_` for consistency.
  - `value` - The value to set `var` to. This is often an expression enclosed in `${}` or `$[]`.
  - `defaultValue` - If the value results to null or empty, this value is put into the `var`.
- Example:

```
<j:set var="jvar_incident_number"
 value="${jvar_now_GR.getValue('number')}"/>
```

set_if

- Description: The `set_if` tag sets a variable based on a test. This tag is similar to the `ternary` operator in other programming languages (`var = <test> ? <if_true> : <if_false>`).
- Parameters:
  - `var` - The variable to set. Often the system prefixes these variables with `jvar_` for consistency.
  - `test` - The condition to evaluate in order to determine if the statement will evaluate the true value or the false value. This should be an expression enclosed in `${}` or `$[]` that evaluates to true or false.
  - `true` - The value to set the variable to if `test` evaluates to `true`. This parameter is optional, so if the field is blank, and if test evaluates to true, the variable is left blank.
  - `false` - The value to set the variable to if `test` evaluates to false. This parameter is optional, so if the field is blank, and if test evaluates to false, the variable will be left blank.

choose

- Description: The `choose` tag starts a choose block of code. This is similar to the `if-elseif-else` kind of syntax in most programming languages. With a `choose` tag, you can use `when` and `otherwise` tags to specify other blocks of code.
- Parameters: None
- Example:

```
<j:choose>
   <j:when test="${jvar_now_GR.getRowCount() ${AMP}lt;
 1}">We found multiple records!</j:when>
   <j:when test="${jvar_now_GR.next()}">We found
 record ${jvar_now_GR.getValue('number')}</j:when>
   <j:otherwise>Sorry, we could not find the record
 you specified.</j:otherwise>
</j:choose>
```

when

- Description: The `when` tag is used within a choose block to indicate a condition. This tag is similar to an `if` or an `elseif` in that it specifies a condition, executes the inner content, and then implies a break at the end to leave the `if-elseif` construct.
- Parameters: `test` - The condition to evaluate in order to determine if the statement will loop through. This should be an expression enclosed in `${}` or `$[]` that evaluates to true or false.
- Example:

```
<j:choose>
   <j:when test="${jvar_now_GR.getRowCount() ${AMP}lt;
 1}">We found multiple records!</j:when>
   <j:when test="${jvar_now_GR.next()}">We found
 record ${jvar_now_GR.getValue('number')}</j:when>
   <j:otherwise>Sorry, we could not find the record
 you specified.</j:otherwise>
</j:choose>
```

otherwise

- Description: The `otherwise` tag is used within a `choose/when/otherwise` block, and is like the `else` or `default` case.
- Parameters: None
- Example:

```
<j:choose>
   <j:when test="${jvar_now_GR.getRowCount() ${AMP}lt;
 1}">We found multiple records!</j:when>
   <j:when test="${jvar_now_GR.next()}">We found
 record ${jvar_now_GR.getValue('number')}</j:when>
   <j:otherwise>Sorry, we could not find the record
 you specified.</j:otherwise>
</j:choose>
```

Glide Tags

evaluate

- Description: The `evaluate` tag evaluates JavaScript code (server side), and makes variables visible to future code. Unlike other tags, the `evaluate` tag evaluates the content that is inside the tag as server side JavaScript.

  The context is the same as that of script includes in the system. Other script includes, global business rules, *`GlideRecord`*, *`GlideSystem`*, and Jelly variables (prefixed with jelly. if the parameter `jelly="true"` is set) are available.

- Parameters:
  - `var` - The name of the variable that will be set to the result of the script.
  - `object` - If set to `true`, the result of the expression is treated as an object instead of a primitive variable (string or integer variable values).
  - `jelly` - If set to `true`, allows Jelly context variables to be referenced in the script.

- `expression` - This is an expression to be executed for the value to put in `var`. The expression can be either of two places. First, it can be an attribute on the `evaluate` tag itself. Otherwise, the content between the beginning tag and ending tag is the expression. The last line of the expression is the actual value passed into `var`.

- Example:

```
<g:evaluate var="jvar_now_GR" object="true">
  var now_GR = new GlideRecord("incident");
  now_GR.addQuery("active", "true");
  now_GR.query();
  now_GR; // this is the variable put into the
 variable jvar_now_GR
</g:evaluate>
```

```
<g:evaluate var="jvar_now_GR" object="true"
 expression="
  var now_GR = new GlideRecord('incident');
  now_GR.addQuery('active', 'true');
  now_GR.query();
  now_GR; // this is the variable put into the
 variable jvar_now_GR" />
```

messages

- Description: The `messages` tag helps with translation. When *`gs.getMessage()`* is called anywhere on a page, there are two possible places where the translation is found. First, the page checks a local cache of translations. Second, the page makes an AJAX call to the server to find the translation. What `g:messages` does is allow pages to cache certain messages.
- Parameters: None
- Example:

```
<g:messages>
  Yes
  No
  Cancel
</g:messages>
```

breakpoint

- Description: When the `breakpoint` tag is called, it prints a list of all the variables in Jelly at the current moment, with their respective values. If a variable is specified, it prints the requested variable and its value. The output is placed in the system log.
- Parameters: `var` - (Optional) The variable to log the value for. If `var` is not specified, then all variables are dumped into the log.
- Example:

```
<g:breakpoint />
```

```
<g:breakpoint var="sysparm_view"/>
```

no_escape

- Description: The system, by default, uses escaped output as a security measure. Output placed inside of `no_escape` tags is not escaped before output. Be careful when using these tags, because if user input is displayed here it can open a security vulnerability on the page.
- Parameters: None
- Example (phase 1) – Disables automatic output escaping of all contained ${} expressions:

```
<g:no_escape>
 ${jvar_raw_html_data}
</g:no_escape>
```

- Example (phase 2) – Use NOESC to disable escaping for the specified string. This implies the expression must evaluate to a string.

```
<g:no_escape>$[NOESC:jvar_expr]</g:no_escape>
```

For information on phase 1 and phase 2 evaluation, refer to the Jelly introduction videos listed at the beginning of this section.

macro_invoke

- Description: The `macro_invoke` tag calls a UI macro that you have specified in the database. You may also call a UI macro by specifying it in the tag name. For example, if you had a UI macro named my_macro, you could call that macro with the tag `<g:my_macro/>`. For information, see UI macros.
- Parameters:
  - `macro` - The name of the UI macro to execute. If your tag name is `g:macro_invoke`, then the macro attribute specifies the name of the macro. If the tag name includes the name of the macro, then there is no need to include a macro attribute.
  - Other attributes - For each attribute you specify, a variable with that name will be available in the context of the UI macro, prefixed with `jvar_`.
- Example:

```
<!-- Will invoke the contents of the UI macro
 named "sample_macro", which will have the variable
 jvar_message available within it-->
<g:macro_invoke macro="sample_macro" message="This is
 a sample macro variable." />
```

```
<!-- Will invoke the contents of the UI macro
 named "sample_macro", which will have the variable
 jvar_message available within it-->
<g:sample_macro message="This is a sample macro
 variable." />
```

if_polaris

- Description: The `if_polaris` tag checks if Next Experience is enabled for the current page. It must include at least one of the child tags `<g:then />` or `g:else />`.
- Parameters: None
- Example:

```
<g:if_polaris>
    <g:then><g:inline
 template="polaris_nav"/></g:then>
    <g:else><g:inline
 template="classic_nav"/></g:else>
</g:if_polaris>

<g:if_polaris>
    <g:then><a href="…">Click here to see what's
 new!</a></g:then>
</g:if_polaris>

<g:if_polaris>
    <g:else>Core UI only code here!</g:else>
</g:if_polaris>
```

then

- Description: The `then` tag is used within an `if_polaris` block to set the page content when Next Experience is enabled.
- Parameters: None
- Example:

```
<g:if_polaris>
    <g:then><g:inline
 template="polaris_nav"/></g:then>
    <g:else><g:inline
 template="classic_nav"/></g:else>
</g:if_polaris>

<g:if_polaris>
    <g:then><a href="…">Click here to see what's
 new!</a></g:then>
</g:if_polaris>

<g:if_polaris>
    <g:else>Core UI only code here!</g:else>
</g:if_polaris>
```

else

- Description: The `else` tag is used within an `if_polaris` block to set the page content when Next Experience isn't enabled.
- Parameters: None
- Example:

```
<g:if_polaris>
    <g:then><g:inline
 template="polaris_nav"/></g:then>
    <g:else><g:inline
 template="classic_nav"/></g:else>
</g:if_polaris>

<g:if_polaris>
```

```
    <g:then><a href="…">Click here to see what's
 new!</a></g:then>
</g:if_polaris>

<g:if_polaris>
    <g:else>Core UI only code here!</g:else>
</g:if_polaris>
```

Jelly escaping types

You use different methods when escaping characters in JavaScript and HTML. JavaScript uses the backslash character, and HTML uses the ampersand character.

Note:
This functionality requires a knowledge of JavaScript, HTML, and Apache Jelly (a Java and XML based scripting and processing engine for turning XML into executable code).

There are two different types of escaping that is required when generating output from Jelly:

- JavaScript
- HTML

The escaping for each of these consists of the following types.

Jelly escaping types

Type	From	To
JavaScript	' (single quote)	\'
	" (double quote)	\"
	CR (carriage return)	(blank)
	NL (newline)	\n ('\' followed by 'n')
HTML	& (ampersand)	&
	< (less than)	<
	> (greater than)	>

You can also escape HTML using the getHTMLValue() function which will enforce all line breaks and escape the characters mentioned above. It can be used as follows:

```
${test.getHTMLValue()}
```

Add escaping to a Jelly replacement

You can handle character escaping in Jelly files. XML escaping behavior can be modified only by users with the security_admin role.

About this task

Note:
This functionality requires a knowledge of JavaScript, HTML, and Apache Jelly (a Java and XML based scripting and processing engine for turning XML into executable code).

Procedure

Add a prefix to the `${expression}` or `$[expression]` indicating the escaping to be performed.

```
${JS:expression}
${HTML:expression}
```

The prefix tells the system to take the result of the expression and escape it before outputting. The escaping may be combined by specifying a comma-separated list of prefixes:

```
${JS,HTML:expression}
```

Extensions to Jelly syntax

Apache's Jelly syntax is used to render forms, lists, UI pages, and many other things rendered in ServiceNow.

With Jelly, logic can be embedded within static content and computed values may be inserted into the static content.

Important:
This functionality requires a knowledge of Apache Jelly (a Java and XML based scripting and processing engine for turning XML into executable code).

This page from Apache has a summary of the standard Jelly tags: http://commons.apache.org/jelly/tags.html

Namespaces

Jelly often includes multiple namespaces when invoking tags.

The "j" namespaces are standard Jelly whereas the "g" namespaces are unique to ServiceNow scripts. For example, the <g:evaluate> tag is supplied by ServiceNow to allow you to compute a value using JavaScript. The standard Jelly tag <j:test> is used to evaluate a condition.

Phases

Usually, there are two phases indicated by namespaces <j> versus <j2> and <g> versus <g2>.

The namespaces without the "2" happen in the first phase of processing and these are cached except when used in a UI page. Those with the "2" are never cached. Care must be taken when selecting whether to use phase 1 or phase 2 for efficiency and correct results.

In addition to the namespaces, the syntax used to insert values into static content differs depending on which phase is to supply the value. A dollar with braces surrounding a value inserts the value in phase 1. For example, `${jvar_ref}` inserts the value `jvar_ref` during phase 1 of the jelly process. A dollar with brackets surrounding a value inserts the value in phase 2. For example, `$[jvar_ref]` inserts the value `jvar_ref` during phase 2. A value surrounded by quotes is treated as a string. For example, `'[jvar_ref]'` inserts the value `jvar_ref` as a string during phase 2.

```
 <script>
if (confirm("$[gs.getMessage('home.delete.confirm') ]"))
```

```
   ...
</script>
```

```
<input type="hidden" id="${jvar_name}" name="${jvar_name}"
 value="${jvar_value}" class="${jvar_class}" />
```

If tests

You can use if statements in Jelly scripts.

Testing whether something is true or not can be done as follows:

```
<j:if test="${jvar_something}">...do something...</j:if>
<j:if test="${!jvar_something}">...do something...</j:if>
```

The reason this statement works, is that, in Jelly, a term like jvar_something is "truthful" in an if tag if:

1. it is Boolean and true
2. it is a String and = "true", "yes", "on", or "1"

Testing whether something exists can be done as follows:

```
<j:if test="${empty(jvar_something)}">...do something...</j:if>
```

The reason this statement works is that the JEXL empty function returns true if its argument is:

1. null
2. an empty string
3. a zero length collection
4. a map with no keys
5. an empty array

Note:
You cannot mix javascript and jvar variables in a JEXL expression. They must be broken into separate expressions.

Set_If

Sets a variable to one of two different values depending on whether a test is true or false.

```
<g2:set_if var="jvar_style"
 test="$[gs.getPreference('table.compact') != 'false']"
   true="margin-top:0px; margin-bottom:0px;"
   false="margin-top:2px; margin-bottom:2px;" />
```

<g:insert> versus <g:inline> versus <g:call>

This page provides a comparative explanation of three tags: <g:insert>, <g:inline>, and <g:call>.

<g:insert>

The <g:insert> tag inserts a Jelly file into your Jelly in a new context. This means you cannot access the variables previously established in your Jelly.

```
<g:insert template="get_target_form_function.xml" />
```

<g:inline>

The <g:inline> tag inserts a Jelly file into your Jelly in the same context. This means that the inserted Jelly can access the variables you previously established and it can change the values of those variables.

```
<g:inline template="element_default.xml" />
```

<g:call>

For better encapsulation, the <g:call> tag may be used. Your function will only have access to the values passed to it. The Jelly context will look the same after a call as before the call. This means you cannot set a global variable here and read it later. This also means you can't mistakenly set a global variable called "jvar_temp" and overwrite a variable that somebody else was relying on.

Passing values, if needed, is done explicitly by including the name of the parameter on the <g:call> line followed by the equal sign followed by the value in quotes:

```
<g:call function="collapsing_image.xml" id="${jvar_section_id}"
 image="$[jvar_cimg]"
   first_section_id="${jvar_first_section_id}"
 image_alt="${jvar_cimg_alt}"/>
```

If values are passed, and you want to have defaults or required parameters, your Jelly referenced in the function must then include a line to declare whether the parameters are required or have a default value:

```
<g:function id="REQUIRED" image="REQUIRED" image_prefix=""
 image_alt="REQUIRED"/>
```

The example above indicates that 3 of the parameter are required and one parameter is option with a blank default value. Note that if you are not passing values or if you do want to have default or required values, you do not need to include the <g:function> line at all. In general, however, you will want to include a <g:function> line.

The value can then be referenced in your template by prepending the "jvar_" prefix to the parameter's name:

```
<img id="img.${jvar_id}" src="images/${jvar_image}"
 alt="${jvar_image_alt}"
   onclick="toggleSectionDisplay('${jvar_id}',
 '${jvar_image_prefix}','${jvar_first_section_id}');"/>
```

For <g:call>, parameters may also be pass implicitly as a list of named variables in an "arguments" parameter:

```
 <g:call function="item_link_default.xml"
 arguments="sysparm_view,ref_parent,jvar_target_text"/>
```

As an alternative to passing variables into the function via separate tag arguments, it is possible to pass a list of variables in a single 'arguments' argument. All variables identified by name (comma separated) in the *`argument`* parameter are re-introduced within the function under the exact same name (e.g. inside the function template, we'd have variables sysparm_view, ref_parent, and jvar_target_text available to us).

The function template may return a value to the calling template using the `return=` attribute. Within the function the `jvar_answer` variable sets the return value.

```
<g:call function="item_body_cell_calc_style.xml"
 arguments="jvar_type" return="jvar_style"/>
```

<g:evaluate>

The <g:evaluate> tag is used to evaluate an expression written in Rhino JavaScript and sometimes to set a variable to the value of the expression.

The last statement in the expression is the value the variable will contain.

```
<g2:evaluate var="jvar_page" jelly="true">
     var page = "";
     var pageTitle = "";
     var pageGR = new GlideRecord("cmn_schedule_page");
     pageGR.addQuery("type", jelly.jvar_type");
     pageGR.query();
     if (pageGR.next()) {
        page = pageGR.getValue("sys_id");
        pageTitle = pageGR.getDisplayValue();
     }
     page;
</g2:evaluate>
```

```
<g2:evaluate var="not_important"
 expression="sc_req_item.popCurrent()"/>
```

object="true"

If you would like to have the evaluate return an object (for example an array), use the argument object="true".

```
<g2:evaluate object="true" var="jvar_items"
 expression="SncRelationships.getCMDBViews()" />
```

jelly="true"

If you would like to access Jelly variables inside an evaluate, include jelly="true" in the evaulate and add "jelly." before the Jelly variable's name. For example, to access the GlideJellyContext:

```
<g2:evaluate var="jvar_row_no" jelly="true">
   var gf = jelly.context.getGlideForm();
   var row = gf.getRowNumber();
   row;
</g2:evaluate>
```

Another example of accessing a jvar using the jelly="true" parameter. The value of `jvar_h` was set previously and we can access it inside the `evaluate`:

```
$[NLBR:jvar_h.getHTMLValue('newvalue')]
<g2:evaluate var="jvar_fix_escaping" jelly="true">
    var auditValue = jelly.jvar_h.getHTMLValue('newvalue');
    gs.log("************ " + auditValue);
</g2:evaluate>
```

copyToPhase2="true"

If you have a need to take the results of an evaluation that occurs in phase 1 and propagate it to phase 2, use copyToPhase2="true". There is some protection for escaping in this use. For example:

```
<g:evaluate var="jvar_has_special_inc" copyToPhase2="true">
   var specialInc = gs.tableExists("special_incident");
   specialInc;
</g:evaluate>
$[jvar_has_special_inc]
```

If you do not need to evaluate something, you can do this more directly. Beware of escaping issues here (double quotes in jvar_rows would cause a problem in the example):

```
<j2:set var="jvar_rows" value="${jvar_rows}"/>
```

<g:breakpoint/>

This tag can be used to display the current Jelly variables and their values in the log.

Be sure to remove this tag before going to production.

<g:ui_form/>

This tag defines a form on the UI page.

For example, if your form contained the `application_sys_id` field, the g:ui_form might benefit from a processing script.

```
<g:ui_form>
    <p>Click OK to run the processing script.</p>
    <g:dialog_buttons_ok_cancel ok="return true" />
    <input type="hidden" name="application_sys_id"
 value="499836460a0a0b1700003e7ad950b5da"/>
 </g:ui_form>
```

For more information, see UI macros.

<g:ui_input_field />

This tag adds a reference to a UI macro that creates an input field on a page that allows users to input information. The `ui_input_field` passes a label, name, value, and size into the UI macro.

Here is an example from a UI page:

```
<g:ui_input_field label="sys_id" name="sysid"
 value="9d385017c611228701d22104cc95c371" size="50"/>
```

For more information, see UI macros.

<g:ui_checkbox/>

This tag puts a user-editable check mark on a page. The name and value are passed into the UI macro.

Here is an example from a table on a UI page:

```
<table>
    <tr>
      <td nowrap="true">
          <label>Time Card Active:</label>
      </td>
      <td>
          <g:ui_checkbox name="timecard_active"
 value="${sysparm_timecard_active}"/>
      </td>
```

```
   </tr>
</table>
```

For more information, see UI macros.

<g:dialog_buttons_ok_cancel/>

This tag puts buttons on the UI page that run a specified processing script if the tag returns true.

If your UI page contains a form (uses the <g:form> tag), you can submit the form and have the Processing Script run. The Processing Script can naturally access fields on the form. For example, if your form contained the application_sys_id field:

```
<g:ui_form>
    <p>Click OK to run the processing script.</p>
    <g:dialog_buttons_ok_cancel ok="return true" />
    <input type="hidden" name="application_sys_id"
 value="499836460a0a0b1700003e7ad950b5da"/>
 </g:ui_form>
```

<g:ui_reference/>

This tag adds a reference to a page that can be referenced by a Processing Script.

The following example creates a reference defined by name, ID, and table parameters in the tag:

```
<g:ui_reference name="QUERY:active=true^roles=itil"
 id="assigned_to" table="sys_user" />
```

Then in the Processing Script, reference the name field like this:

```
newTask.assigned_to =
 request.getParameter("QUERY:active=true^roles=itil");
```

You can specify a reference qualifier, so that the "name" attribute can be unique. The following example creates a reference defined by name, ID, and table parameters in the tag.

Note:
The "columns" attribute only applies to the auto-completer.

```
<g:ui_reference name="parent_id" id="parent_id"
 table="pm_project" query="active=true"
 completer="AJAXTableCompleter"
columns="project_manager;short_description"/>
```

Ampersand

Ampersands in Jelly can cause you grief because Jelly is XML.

Use `${AMP}` to insert an ampersand in Jelly. If you are writing JavaScript that appears in the HTML part of say a UI page or UI macro that is actually going to run on the browser you are better off putting this code in the "client script" field and that way you can avoid escaping issues. However, if you really must put it in the "HTML" field, you will need to do something like this:

```
ta = ta[1].split('$[AMP]');
```

And

Use `${AND}` to insert a JavaScript and in Jelly.

For example:

```
if (d ${AND} e)
   var color = d.value;
```

Alternately, in a Jelly test you would use `&amp`. For example:

```
<j:if test="${jvar_form_name == 'sys_form_template'
 &amp;amp;amp; !RP.isDialog()}">
```

Less than

Similar to ampersands, less than ("<") signs can also cause problems due to Jelly being XML. This can be resolved by reversing your test such that it is not necessary or by using `${AMP}lt;` in place of the less than sign.

```
<g2:evaluate var="jvar_text">
     var days = "";
     var selectedDays = '$[${ref}]';
     for (var i = 1; i ${AMP}lt;= 7; i++) {
        if (selectedDays.indexOf(i.toString()) >= 0) {
           days += gs.getMessage("dow" + i);
           days += " ";
        }
     }
     days;
 </g2:evaluate>
```

Many times you can avoid the "less than" operator all together by just using "not equals" which doesn't have escaping issues. For example:

```
for (var i=0; i != ta.length; i++) {
}
```

Whitespace

Normally, white space is removed by Jelly. To keep it, you must specify that it not be trimmed.

For example, the following keeps the space after the colon.

```
<j2:whitespace trim="false">${gs.getMessage('Did you mean')}:
 </j2:whitespace>
```

Spaces

To encode a non-breaking space ( ), you can use `$[SP]`.

For example:

```
<span id="gsft_domain" style="display: inline">
    ${gs.getMessage('Domain')}:$[SP]
    <span id="domainDD" class="drop_down_element"
 style="text-decoration: none; color: white">
        ${gs.getMessage("Loading...")}
   </span>
</span>
```

Tracing Jelly

ServiceNow has a feature that allows the evaluation of Jelly to be traced.

The trace is sent to the log. This should only be turned on during debugging as this produces a lot of logging. To turn on the trace, set the property glide.ui.template.trace to true. For example, the following script can be executed to do this:

```
GlideProperties.set ( 'glide.ui.template.trace' , true ) ;
```

If you want to see your log entries on your web browser at the bottom of each page, navigate to **System Diagnostics** > **Debug Log**.

Debugging scripts

Debug scripts using session logs and ServiceNow AI Platform debugging tools such as a walk-through script debugger and error messages that display in the UI.

Debugging server-side scripts

Use the Script Debugger and session logs to debug server-side code. For more information, see Script Debugger and Session Log.

You can also use session debug to display error messages related to a server-side script that runs as a result of a client-side change. For more information, see Session debug.

GSLog is a script include that simplifies script logging and debugging by implementing levels of log output, selectable by per-caller identified sys_properties values. For more information, see GSLog API.

Debugging client-side scripts

Use session debug to display debugging messages in the user interface. For more information, see Session debug. Use the session log to view logging information for script includes and custom UIs, such as Agent Workspace.

You can also debug client-side scripts using browser-based developers tools.

Debugging applications and scopes

Use the application debugging options to understand how a script's application scope might affect your application, table, or record. You may need to update cross-scope privileges to troubleshoot scope access issues. See Debugging applications.

Script Debugger and Session Log

The Script Debugger enables users with the script_debugger role to debug server-side JavaScript. Users with the log_debugger role can use the Session Log to view and download required logs.

Users with the script_debugger role can perform these actions using Script Debugger:

- Have a dedicated debug transaction, which applies only to the current session.
- Set and remove breakpoints.
- Pause the current session at a breakpoint.
- Evaluate expressions during runtime.
- Step through code line-by-line.
- Step into and out of function and method calls.
- View the value of local and global variables.
- View the value of private variables from function closures.

- View the call stack.
- View the transaction that the system is processing.
- Turn off the script debugger to resume running paused scripts.

Use the Session Log tab to retrieve the session log for business rules, script includes, and a custom UI such as ServiceNow® Agent Workspace that has a GraphQL component. Users with the log_debugger role can:

- View session logs in a separate tab.
- Download a log.
- View logs for Agent Workspace.
- Specify debug options to view or download only the required logs.

By default, 100 transactions and 10,000 messages appear on the Session Log tab. If the transaction or message count exceeds the default value, the session log is cleared and the next transactions or messages appear. You can configure this transaction and message count using the *glide.debugger.log.transaction.count* and *glide.debugger.log_messages_limit* user preferences respectively. For more information about the *glide.debugger.log.transaction.count* and *glide.debugger.log_messages_limit* user preferences, see User preference settings.

Note:
Enable Session Log as a separate tab with Script Debugger using the *glide.debugger.log.ui* system property.

- The **Page** option displays logs under forms and lists and on the **Session Log** tab.
- The **Session** option displays logs only on the **Session Log** tab.

For more information about the *glide.debugger.log.ui* system property, see Available system properties.

When you execute a statement in the Console, the executed statement is stored in the browser cache. You can use the up arrow key to get the previous statement and down arrow key to get the next statement from the browser cache. The user preference setting, *glide.debugger.console.cached_stmt_limit*, defines the number of statements cached in a browser session. The default statement cache value is 20 and the maximum value is 100. You can configure the statement cache value from user preferences.

Note:
The cached statements are not available when the browser cache is cleared or when you log in from a different browser or a different computer.

The Script Debugger can pause any server-side script that runs in an interactive transaction such as business rules, script includes, script actions, or UI actions that require a response to proceed. If the GlideSystem method `isInteractive()` returns **True** when running the script in context, then the Script Debugger can pause it.

Note:
Some script objects, such as script includes, can be called from multiple contexts. For example:

- when a business rule runs a script include on a form submit that is an interactive transaction waiting on the form data to change before continuing.
- when a scheduled job runs the same script include that is a non-interactive background transaction that can also run other scripts simultaneously.

To debug client-side scripts, you can use browser-based developers tools.

A debugger transaction remains open as long as the user session is valid. If a user logs out or their session times out, the system closes the debugger transaction.

To view debug logs, see Display debugging logs.

Note:
When the Script Debugger is enabled, code is executed in interpreted mode. If parts of the script are set to run in strict mode, the debugger is not able to find the correct objects and the debugger fails. The Script Debugger must run on scripts outside of strict mode.

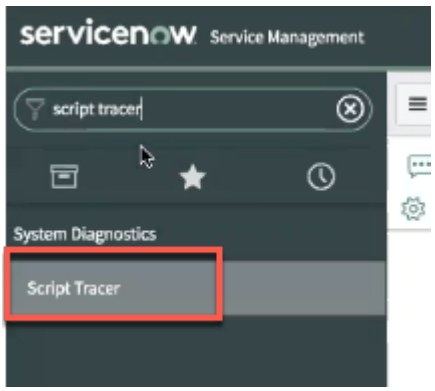
Script Tracer and debugging scripts

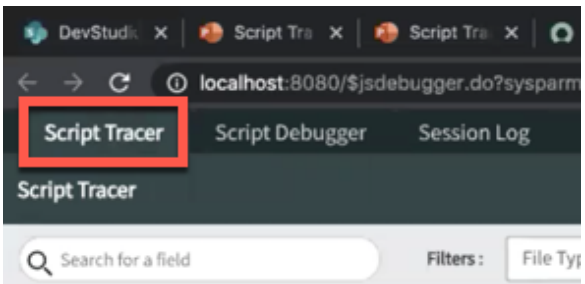
The Script Tracer can help you filter your debugging search to quickly narrow down script problems. You can identify lines of scripts in the Glide record that have undergone change during execution. Finding those specific lines of scripts rather than doing a wide search helps save time and improves productivity.

Overview

Use the Script Tracer to narrow your search so you can debug scripts and business rules more efficiently. You can find the Script Tracer by searching in the left navigation pane.

Note:
To use the Script Tracer, your role must be admin.

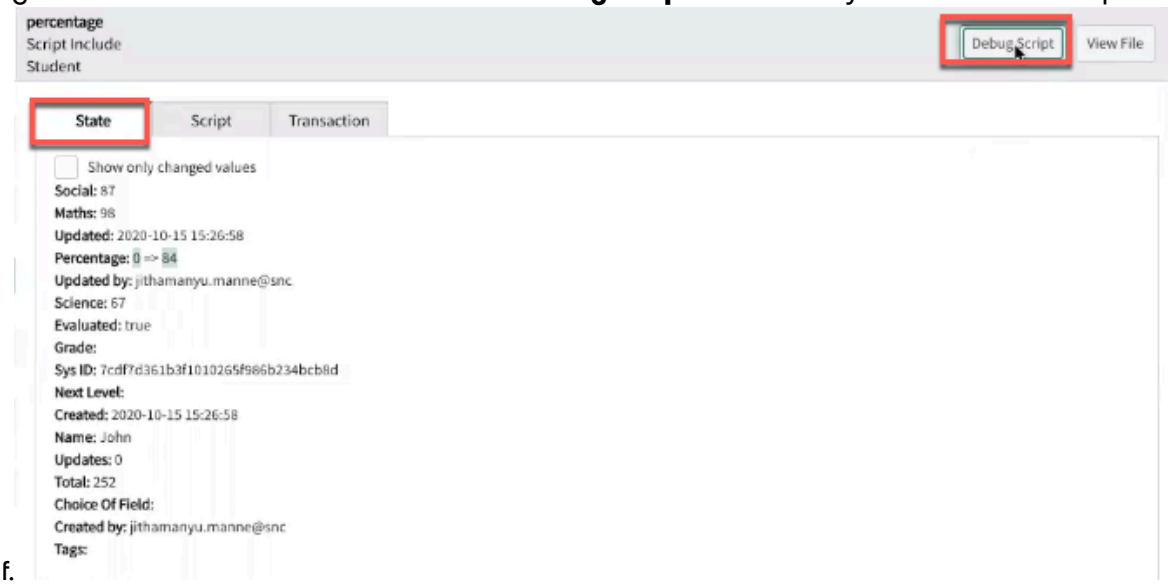




Once you enable Script Tracer and execute a UI transaction, the Tracer searches through all the scripts being executed. The following filters are available:

- **File type**: Search for a specific file type
- **Table**: Look in the specific table for the script being executed

The Script Tracer searches for changes in the script during execution and presents them in a list for you to examine. When you click **Start Tracer**, the Tracer begins searching for changes in the Glide record. You can click the **Debug Script** button at any time to see the script itself.



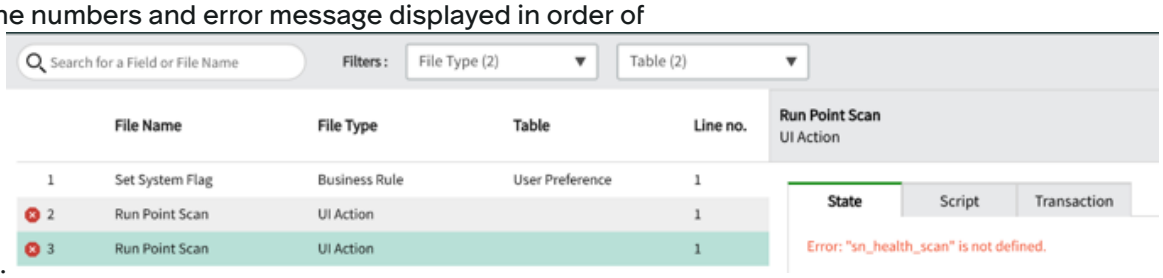
Use the tabs to see specific information from the Tracer.

The **State** tab displays the differences between the old and new scripts.

- By default, the **Show only changed values** check box is enabled, so you can avoid fields that have not changed.
- To view all the fields (changed or not), you can clear that check box.

Note:
If the file is not reflected in the trace statement, it means the changes in the Glide record is not recognized by the system.

If there are any errors, they display at the top of the State tab, with their line numbers and error message displayed in order of occurrence.



- **Script**: Displays the line of changed scripts that the Glide record has undergone during execution. You can view the entire line of script by clicking the **Show Script** button.
- **Transaction**: Shows all transaction records of the trace
- **Debug Script**: Opens the script in Debugger to debug the script
- **View File**: Opens the script in the ServiceNow platform for editing
- **Clear trace**: Clears the trace when you are finished.

Limiting the tracer

You may want to set a limit for your trace so that you don't generate too many returns. By default there will be up to 1,000 lines of script traced. Once this number is reached you must clear the

trace and start tracing again. If you want to change the maximum number of lines for tracing you can configure your limit using the property *glide.debug.trace.trace_line_limit*.

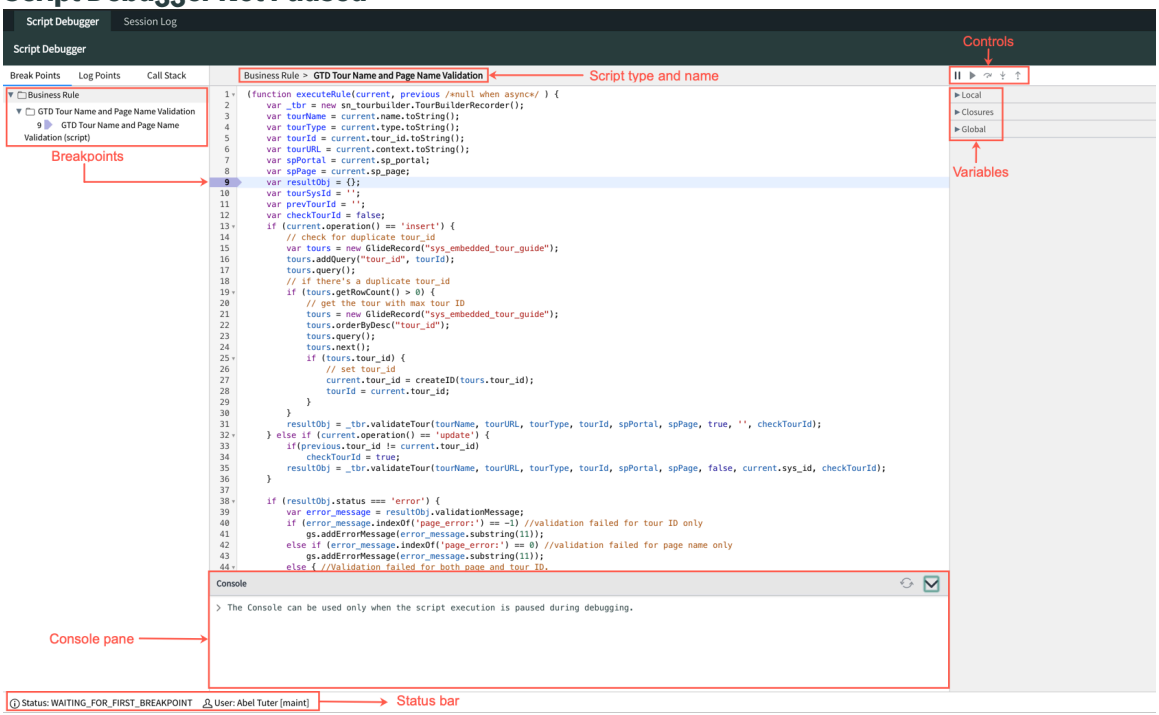
Since each trace you run is new, make sure you're finished reading the results of one trace before clearing it and beginning another one.

To learn more, see Debugging scripts.

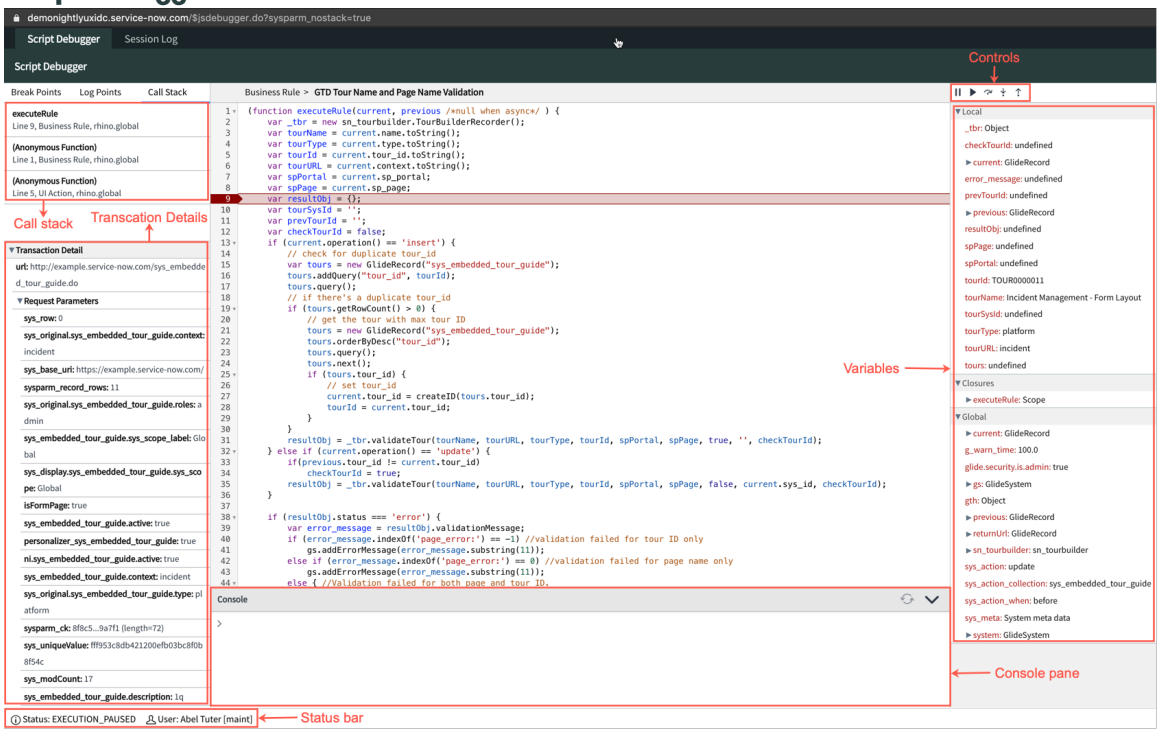
Parts of the user interface

The Script Debugger interface displays information about breakpoints set, the call stack and line number of the currently executing script line, details about variables and transactions, and status of console.

Script Debugger Not Paused



Script Debugger Paused



Parts of the Script Debugger

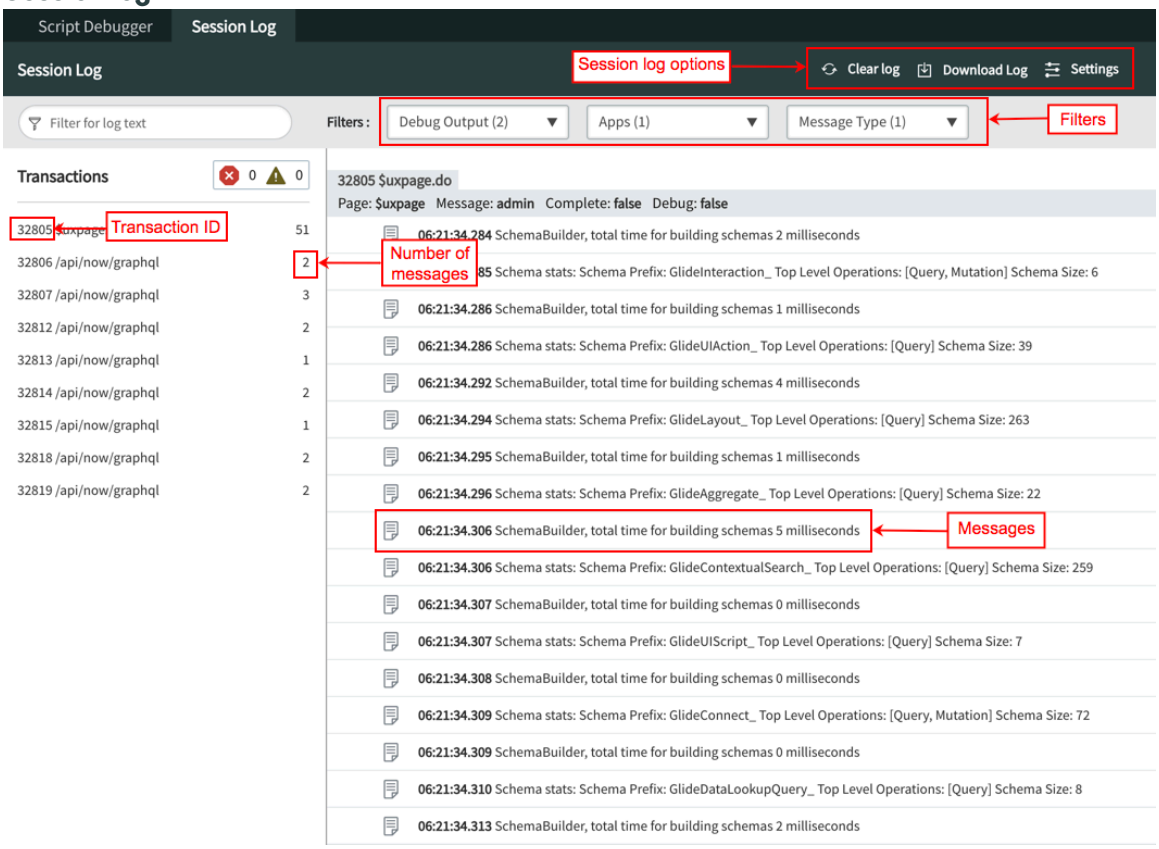
User interface element	Description
Breakpoints	Displays a list of the breakpoints set by script type, script name, and line number. The debugger updates this list as you add and remove breakpoints.
Call stack	Displays a list of script calls that preceded or invoked the current line number. This information is only visible when the debugger pauses on a breakpoint.
Transaction details	Displays information about the current transaction. This information is only visible when the debugger pauses on a breakpoint.
Status	Displays if the debugger is waiting for a breakpoint, paused on a breakpoint, or has encountered an exception.
User	Displays the name of the user who is running the current debugger session.
Coding pane header	Displays the script type and name of the script in the coding pane.
Breakpoint icon	Indicates the line number where the debugger pauses when evaluating the current script.
Pause debugging button	Stops any current debugging session, and disables the Script Debugger for the current user. The Script Debugger doesn't pause on breakpoints for the current user until it's restarted.

Parts of the Script Debugger (continued)

User interface element	Description
Console	Displays a command line interface used for evaluating expressions during runtime. The console is available only when the script execution is paused.
Resume script execution button	Advances from the current breakpoint to the next breakpoint. If there are no other breakpoints, the script runs to completion.
Step over next function call button	Advances past the method that's about to be called, executing the method as a single step.
Step into next function call	Advances to the first line of executed code within a method call. Stepping into a method updates the current position within the call stack. If the user doesn't have read access to the method call, then this control acts like step over instead.
Step out of current function	Exits from current method call and returns to the calling script from the call stack. If the user isn't within a method call, then this control acts like step over instead.
Local	Displays a list of local scope JavaScript variable names and their values. This information is only visible when the debugger pauses on a breakpoint.
Closures	Displays a list of global scope JavaScript variable names and their values set by function closure. This information is only visible when the debugger pauses on a breakpoint.
Global	Displays a list of global scope JavaScript variable names and their values. This information is only visible when the debugger pauses on a breakpoint.

Session Log

Session log



Session Log user interface elements

User interface element	Description
Transactions	Transaction ID. Displays information about the current transaction.
Filter for log text	Field to enter text to filter the logs that contain a specific text.
Debug Output	Option to filter logs based on the dynamically loaded debug output types. For example, **Security Rule**.
Apps	Option to filter logs based on the dynamically loaded apps. For example, **Service Management Integrations**.
Message Type	Option to filter logs based on the dynamically log levels. For example, **Info**.
Clear log	Clears all logs.
Download log	Download the logs in HTML file format.
Settings	Session debug options. For information about the debug options, see Session debug.

Script Debugger step-through and console controls

After the Script Debugger pauses a script, use the step-through controls to move between script lines and move between scripts in the call stack. Use the Console controls to expand console, collapse console, clear console, and rerun expressions.

Step-through controls

Control	Icon	Description
Stop debugging SHIFT+F2		Stops any current debugging session, and disables the Script Debugger for the current user. The Script Debugger does not pause on breakpoints for the current user until it is restarted.
Start debugger - F2		Enables the Script Debugger for the current user. The Script Debugger pauses on breakpoints.
Resume script execution - F9		Advances from the current breakpoint to the next breakpoint. If there are no other breakpoints, the script runs to completion.
Step over next function call - OPTION +F9		Advances to the next evaluated line of script based on current conditions. The Script Debugger skips any lines of code that do not need to run because their conditions are not met. For example, when the condition of an `if` statement is not true, the debugger skips the code block for the condition.
Step into next function call - OPTION +F10		When the Script Debugger pauses on a method call, this control allows the user to advance to the first line of executed code within the method call. Stepping into a method updates the current position within the call stack. If the user does not have read access to the method call, then this control acts like step over instead.
Step out of current function - OPTION +F11		When the Script Debugger pauses within a method call, this control allows the user to exit the current method call and return to the calling script from the call stack. If the user is not within a method call, then this control acts like step over instead.

Console controls

Control	Icon	Description
Open Console		Expands the Console.
Close Console		Collapses the Console.
Clear expressions		Clears all the expressions in the Console.
Re-execute expression		Re-executes the expression which is already executed.

Related topics

Set or remove breakpoints

Evaluate expressions in runtime using Console

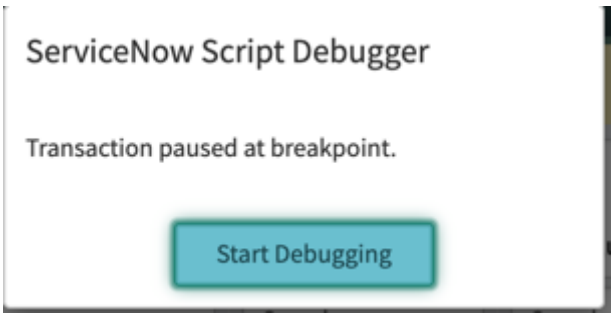
Define, declare, and verify new variables and functions while you debug a script in runtime using Console. The script execution must be paused to use Console.

Before you begin

- Review Limitations with using Console
- Role required: script_debugger, admin

Procedure

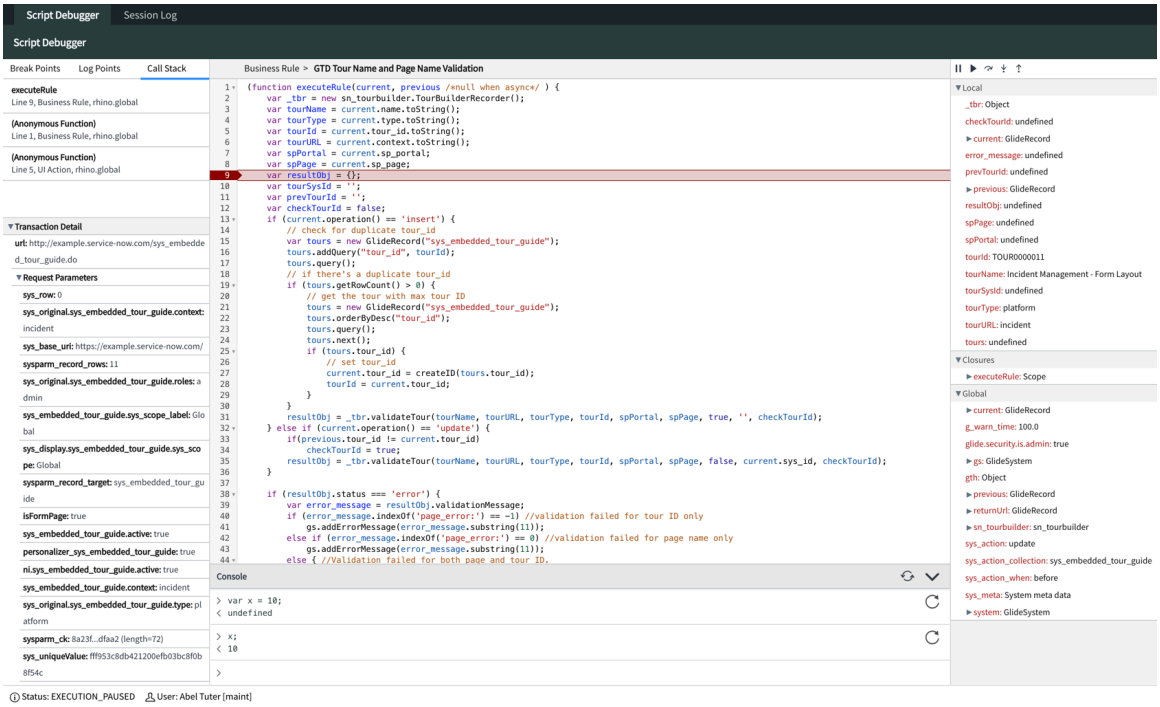
1. Launch the Script Debugger in one of the following ways.
   The Script Debugger modal is displayed.
2. Trigger the script.
   For example, create a record to trigger an insert business rule script. The Script Debugger pauses the script on the first line that contains a breakpoint, and then you see the ServiceNow Script Debugger confirmation window.



3. Click **Start Debugging**.
   The focus shifts to the Script Debugger window and you see the target script that paused at the first breakpoint.

   Note:
   Make sure that the status of Script Debugger is EXECUTION_PAUSED. You can use Console only when the script execution is paused during debugging.

4. Click the expand console (^) to expand the Console pane.
   To start evaluating expressions, enter one or more expressions in the Console and press Enter. For example, enter `var x = 10;` and press Enter. To enter multiple lines of expressions, press Shift + Enter after each line and press Enter after the last expression. To clear all the expressions in the Console, click the clear console icon (⟳). For more information on Console controls, see Script Debugger step-through and console controls.



After a statement is executed, it is stored in the browser cache. You can use the up arrow key to get the previous statement and the down arrow key to get the next statement from the browser cache. You can configure the number of cached statements for a session from user preferences. For more information about user preferences settings, see Script Debugger and Session Log.

Result

After a statement is executed, it is stored in the browser cache. You can use the up arrow key to get the previous statement and down arrow key to get the next statement from the browser cache. You can configure the number of cached statements for a session from user preferences. For more information about user preferences settings, see Script Debugger and Session Log.

Related topics

Limitations with using Console

Limitations with using Console

You need to be aware of a few limitations when you use Console to evaluate expressions while debugging a script in runtime.

- The properties and values of an object don't display in Console. When you try to display an object to Console, only the string value of the object appears.
- Console doesn't support GlideSystem printing methods, such as info() and print().
- You can't use the `this` keyword in Console.
- A script debugger timeout occurs when you evaluate expressions in Console.
- While executing long scripts, if you see the response `Awaiting response from server`, you can't resume debugging or stop debugging using the resume or stop controls.

Launch the Script Debugger

Developers can launch the Script Debugger from the application navigator, Studio, or from the syntax editor.

Before you begin
Role required:

- admin
- script_debugger

Procedure

Select a path based on your starting point.
The system opens the Script Debugger in a new window.

Set or remove breakpoints

Set breakpoints or conditional breakpoints to pause scripts at specific lines, and remove breakpoints when you are done debugging them.

Before you begin
Role required:

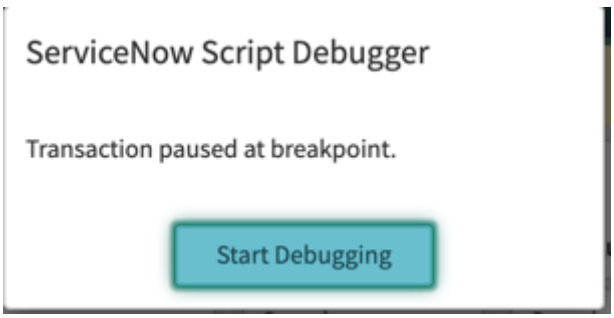
- admin
- script_debugger

About this task
Breakpoints belong to the developer who sets them. Developers must set and remove their own breakpoints.

Note:
At a specific line, you can set either a logpoint or breakpoint but not both.

Procedure

1. Navigate to the server script to debug.
   For example, navigate to **All** > **System Definition** > **Business Rules**.
2. From the Syntax Editor, click the gutter next to a script line.
3. From the Syntax Editor toolbar, click the **Open Script Debugger** icon.
   The system opens a Script Debugger window.
4. Trigger the script.
   For example, create a record to trigger an insert business rule script.
   The Script Debugger pauses the script on the first line containing a breakpoint, and the system displays a confirmation window.



5. Click **Start Debugging**.

The system switches focus to the Script Debugger window and displays the target script paused at the first breakpoint. Console pane is enabled.

6. When debugging is complete, remove breakpoints from the script.

Related topics

Script Debugger step-through and console controls

Set or remove logpoints

Set breakpoints or conditional logpoints to log messages to the console at specific lines, and remove logpoints when you are done debugging them.

Before you begin

- Set the *glide.debug.log_point* system property to `true`. See Available system properties for more information.
- Role required: admin or script_debugger

About this task

Logpoints belong to the developer who sets them. Developers must set and remove their own logpoints.

Note:
At a specific line, you can set either a logpoint or breakpoint but not both.

Procedure

1. Navigate to the server script to debug.
   For example, navigate to **All** > **System Definition** > **Business Rules**.
2. From the Syntax Editor, click the gutter next to a script line.

   Note:
   The script entered for the logpoint must have the same format as that of the script in GSLog and GSInfo script includes.

3. From the Syntax Editor toolbar, click the **Open Script Debugger** icon .
4. On the Script Debugger window, trigger the script.
   For example, create a record to trigger an insert business rule script.

   Note:
   You can also add logpoints in the Script Debugger window.

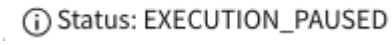
5. In the Script Debugger window, click **Session Log** to view the logpoints.
6. When debugging is complete, remove logpoints from the script.

Script Debugger status

The Script Debugger status determines what debugging actions are available and what information it can display.

The Script Debugger displays its status at the bottom left of the user interface.

Sample Script Debugger status



Possible Script Debugger status values

Status	Occurs when	Description	Actions available
WAITING_FOR_FIRST_BREAKPOINT	The user opens a Script Debugger window or tab.	The Script Debugger is ready to pause script and display debugging information.	Pause script at the first breakpoint in the call stack.
EXECUTION_PAUSED	• The Script Debugger pauses on a breakpoint. • The user steps over, steps into, or steps out to the next line of evaluated code.	The Script Debugger has paused on a line of code, and the user can debug the script. Console is enabled.	• Resume processing until the Script Debugger reaches the next breakpoint. • Step through a script. • Display the call stack. • Display transaction information. • Display variable values. • Evaluate expressions in Console during runtime.
WAITING_FOR_BREAKPOINT	• The user resumes processing until the Script Debugger reaches the next breakpoint. • The user steps through a script until the Script Debugger reaches the next line of code to evaluate or the transaction completes.	The Script Debugger is searching for the next line of code at which to pause. Users will typically never see this status because the Script Debugger changes status after it locates the next breakpoint or script line to evaluate.	• Pause script at the next breakpoint. • Pause script at the next script line requiring evaluation.
OFF	• The user pauses the Script Debugger. • The user closes the Script Debugger window or tab. • The user session ends for any reason. • The administrator resets all Script	The Script Debugger is inactive and does not pause scripts or display debugging information.	• Start the Script Debugger. • Open a Script Debugger window or tab.

Possible Script Debugger status values (continued)

Status	Occurs when	Description	Actions available
	Debugger instances by navigating to the `debugger_reset.do` page.		

Log entries

Every time a debug transaction finishes executing, the system creates a log entry for it with a DEBUGGED prefix. For example:

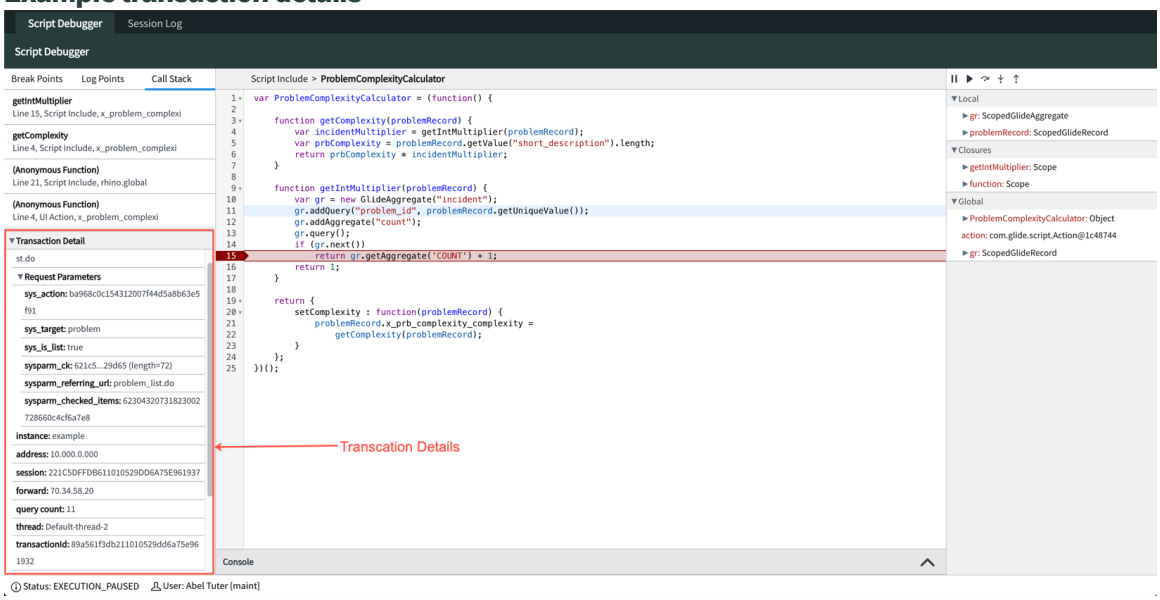
```
2016-08-15 15:57:32 (197) Default-thread-3
 900F5101671122O0C4098C7942415A75 *** End
#39, path: /my-app.do, user: admin, DEBUGGED total transaction
 time: 0:00:11.010,
transaction processing time: 0:00:11.010, network: 0:00:00.000,
 chars: 6,058, uncompressed
chars: 20,731, SQL time: 50 (count: 34), business rule: 0
 (count: 0), phase 1
form length 56,464, largest chunk written: 10,428, request parms
 size: 40, largest input read: 0
```

Transaction details

The Script Debugger displays transaction details for the current paused user session.

Transaction details are available in a dedicated resizeable section underneath the Call Stack on the bottom left of the Script Debugger.

Example transaction details



The Script Debugger only displays transaction details when it pauses on a script. Developers can use transaction details to:

- Inspect the URL of the currently paused transaction.
- Inspect the request parameters for the currently paused transaction.

- Inspect network information about the current transaction.
- Inspect the user and session ID that initiated the debug transaction.

Related topics

Available transaction details

Available transaction details

The Script Debugger provides a standard set of transaction details for developers to debug and troubleshoot scripts.

Available transaction details

Transaction element	Description
url	The URL of the currently paused transaction.
Request parameters	The list of request parameters for this transaction. Each transaction has its own list of request parameters, but record transactions typically include the field values used to insert, update, or delete a record.
instance	The instance name.
address	The IP address of the end-user client system.
session	The user session ID.
forward	The IP address of the load balancer.
query count	The number of database queries the Script Debugger has made.
thread	The name of the thread running the Script Debugger instance.
transactionid	The Sys ID of the current transaction.
token	The Script Debugger token of the currently paused transaction. The system uses this token to identify different Script Debugger instances.
name	The name of the currently paused transaction. You can use this name to identify transactions in the logs.
processor	The name of the processor processing the current transaction, if present.
method	The HTTP request method the currently paused transaction uses.
startTime	The date-time stamp when the Script Debugger instance started.
page	The current table or UI page associated with the transaction.
user	The user who triggered the debug transaction.
nodeid	The Sys ID of the node running the Script Debugger instance.

Related topics

Transaction details

Script Debugger multiple developer support

The Script Debugger allows multiple developers to debug their own transactions without affecting each other.

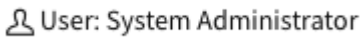
The Script Debugger only allows developers to see and interact with items related to their current debugging session such as:

- Breakpoints
- Call stack
- Console
- Transactions
- Status

The Script Debugger prevents one developer from seeing or modifying another debug session. Administrators, however, can impersonate another user, open the Script Debugger, and debug transactions generated by the impersonated user.

The Script Debugger displays the debug session user at the bottom left of the user interface.

Sample Script Debugger user



Concurrent Script Debugger usage

By default, the system supports debugging `[(The number of semaphores on the instance) / 4]` concurrent transactions. Administrators can specify the number of concurrent transactions the system can debug by setting the *`glide.debugger.config.max_node_concurrency`* system property. The system can debug up to `[(The number of semaphores on the instance) - 2]` concurrent transactions.

Administration of debugging sessions

Debugging sessions can remain actively debugging (in the EXECUTION_PAUSED or WAITING_FOR_BREAKPOINT statuses) until:

- The user pauses the Script Debugger.
- The user closes the Script Debugger.
- The user session ends.

Administrators can view the currently running debugger sessions by navigating to the page `xmlstats.do`.

Administrators can stop all currently running debugging sessions by navigating to the page `debugger_reset.do`. Only users with the admin role can access this page.

Related topics

Script Debugger impersonation support

Script Debugger impersonation support

You can use the Script Debugger while impersonating another user, but only if the impersonated user has the script_debugger role and has read access to the target script.

While impersonating another user, you can:

- See and change breakpoints that belong to the impersonated user.
- View and pause on scripts that the impersonated user has read access to.
- Evaluate expressions in Console on behalf of the impersonated user.

The Script Debugger step-through controls also use the read access of the impersonated user. For example, if the impersonated user does not have read access to a function in the call stack, any **Step into** action instead becomes a **Step over** action.

The impersonated debugging session lasts until:

- You stop impersonating the user.
- You log out or the user session ends.
- You pause the Script Debugger.
- You close the Script Debugger.

Related topics

Script Debugger multiple developer support

Script Debugger Scripts - Background support

The Scripts - Background module does not support setting breakpoints directly in the script field. You can however, set breakpoints in the script objects called or triggered by the Scripts - Background module.

While running arbitrary JavaScript code in the **Scripts - Background** module, the Script Debugger can only pause scripts when you:

- Call a script include containing breakpoints.
- Trigger a business rule containing breakpoints.
- Trigger a script action containing breakpoints.

Domain separation and Script Debugger

Domain separation is supported in Script Debugger. Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Support level: Basic

- Business logic: Ensure that data goes into the proper domain for the application's service provider use cases.
- The application supports domain separation at run time. The domain separation includes separation from the user interface, cache keys, reporting, rollups, and aggregations.
- The owner of the instance must set up the application to function across multiple tenants.

Sample use case: When a service provider (SP) uses chat to respond to a tenant-customer's message, the customer must be able to see the SP's response.

For more information on support levels, see Application support for domain separation.

How domain separation works in Script Debugger

Script Debugger is not a full application but rather, a feature in the Platform suite, meaning it works alongside other features, including domain separation.

Related topics

Domain separation for service providers

Session debug

Enable session debugging to display debugging messages in the user interface.

Troubleshooting slow performance with the Session Debug feature.

You can enable all areas for abundant logging on the bottom of each page load, or you can enable each module one by one, for more specific information about what is happening during this session, and specifically, for the previous transaction. Select session debug options under **System Diagnostics** > **Session Debug**. When enabled, session debugging is active during the user session or until disabled. To view debug logs, see Display debugging logs.

The system provides the following session debugging options.

Session debug options

Debug option	Description
Enable All	Displays all available debugging messages.
Disable All	Stops displaying all debugging messages.
Debug Business Rule	Displays debugging messages for business rules. If there are business rules from multiple applications affecting a table or record, the system displays which application the business rule comes from.
Debug Upgrade	Displays detailed information logged for records processed during the last family-to-family or patch version upgrade session. See Debug Upgrade .
Debug Business Rule (Details)	Displays debugging messages for business rules and any changes made by business rules. If there are business rules from multiple applications affecting a table or record, the system displays which application the business rule comes from.
Debug Log	Displays all log entries.
Debug NLQ	Displays debugging messages for Natural Language Query (NLQ) queries.
Debug Date/Time	Displays Date/Time failures when inputs do not match required formats.
Debug SQL	Displays debugging messages for SQL queries.
Debug SQL (Detailed)	Displays debugging messages for SQL statements and any changes made by SQL statements.
Debug Security	Displays debugging messages for access controls. If there are access controls from multiple applications affecting a table or record, the system displays which application the access controls comes from.
Debug Escalations	Displays debugging messages for SLA and SLO escalations.
Debug AI Search	Displays debugging messages for AI Search.
Debug Metric Statistics	Displays an aggregate view of performance data (slow transactions, scripts, queries, events, and mutexes). These aggregate metrics are sorted by transaction, to help identify items that affect page performance.
Debug Text Search	Displays debugging messages for search result relevance and indexing.

Session debug options (continued)

Debug option	Description
Debug UI Policies	Displays debugging messages for UI policies.
Disable UI Policies Debug	Stops displaying debugging messages for UI policies.
Debug UI Macro	Displays the start and end of the UI Macro in the DOM as HTML comments. The comments consist of table name and UI macro name.
Disable Debug UI Macro	Stops displaying the start and end of the UI Macro in the DOM as HTML comments.
Debug Data Policies	Displays debugging messages for data policies.
Debug Quotas	Displays debugging messages for transaction quotas.
Debug Homepage Render	Displays debugging messages for homepages.
Debug Scopes	Displays debugging messages for entering or exiting application scopes when running script objects.

Display debugging logs

Display session debug logs to help diagnose script and application problems.

Before you begin

Role required: none

Procedure

1. Navigate to **All** > **Session Debug** and select **Enable All**.
2. Under **Session Debug**, select **Debug Log**.
   The Debug log displays.

Debugging applications

Application developers can display debug messages about configuration records to help them troubleshoot issues. The Debug Scopes module provides information about the system switching between custom applications to run server-side scripts.

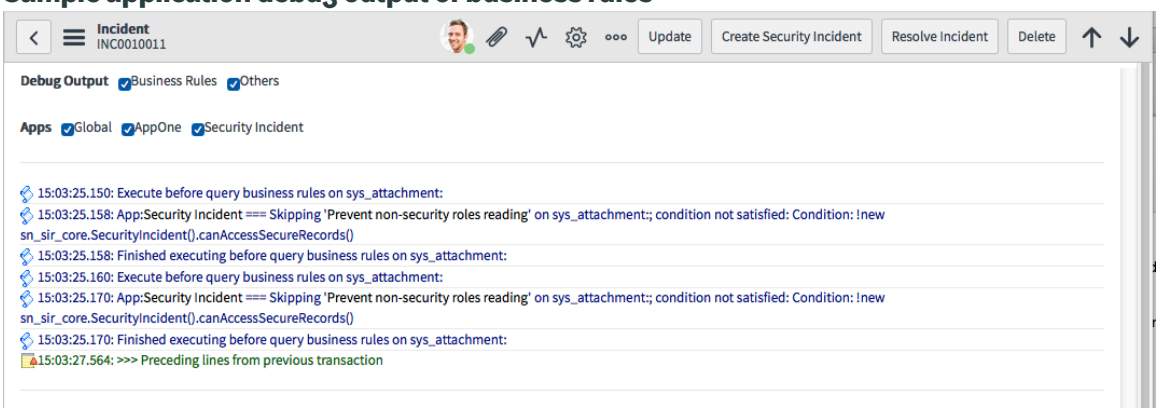
The system offers the following debugging options to help application developers determine how applications affect configuration records.

Application debug options

Debugging option	Description
Debug Business Rule	Use this module to determine which application's business rules are running against tables. The system only displays application information if business rules from different application scopes run on the same table.
Debug Business Rule (Details)	Use this module to determine the results of running business rules against tables. The system only displays application information if business rules from different application scopes run on the same table.
Debug Security	Use this module to determine which application's access controls apply to a given table or record.
Debug Scopes	Use this module to determine the application scope context in which a script runs. Since one script can call another script it is possible to have multiple application scope context changes while running a series of scripts.
Enable Session Debug	Use this related link to enable the generation of log messages for a particular application. Application scripts that use GlideSystem logging methods will generate output to the log at the indicated verbosity level.

When multiple applications contribute to the debug output, the system adds a new section called **Apps** to the display a list of the applications writing to the session log. Clicking on the check box next to the application name hides or displays the application's associated debug messages.

Sample application debug output of business rules

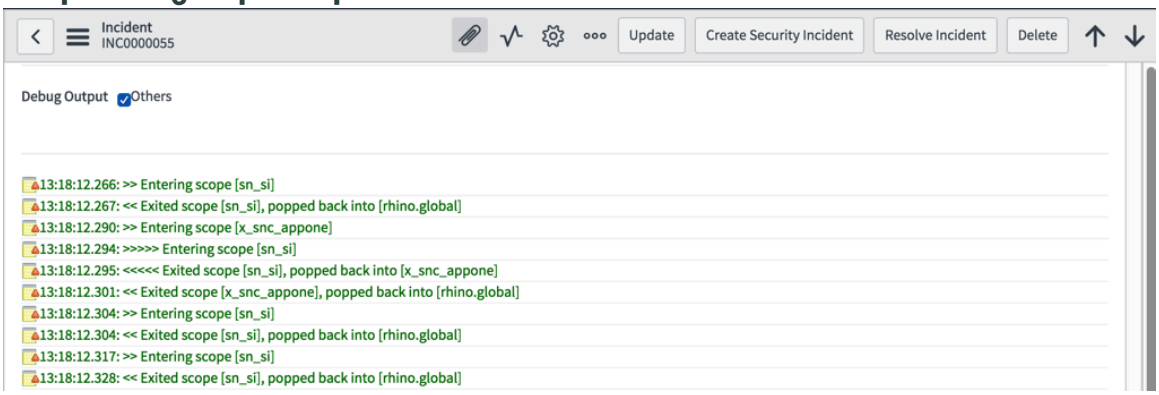


Debugging scopes

Application developers can use the **Debug Scopes** module to display information about when the system switches between custom applications to run server-side scripts.

When enabled, the system displays a message whenever the system switches to a custom application to run a server-side script.

Sample debug scopes output from the incident table



Every time the system runs a server-side script object it enters the script's scope context. When the script finishes running, the script exits the scope context. The debugging messages track changes to the script scope context.

The debugging message displays a greater than character > each time the system enters a script object's context, and displays a less than character < every time the system exits a script object's context. In cases where one script calls another the debugging message adds another greater than character to the path for each call. For example, if a business rule calls a script include, which in turn calls another script object there would three characters in the path such as:

```
> Entering scope [x_app_one]
>> Entering scope [x_app_two]
>>> Entering scope [x_app_three]
```

Note:
The system does not display entering or exiting messages for script objects in the global scope.

Application developers may want to enable other debugging options to in conjunction with this option to see information about the possible source of the server-side script such as Debug Business Rule.

Debugging business rules

Debugging business rules can be achieved with resources available in the ServiceNow product.

1. Tools

The first step in the process is to identify tools which will help you figure out what's wrong.

Debugging tools

Debugging tool	Description
System Dictionary	Navigate to **System Definition** > **Dictionary**. The dictionary provides a list of all tables within your instance and can be invaluable when trying to locate information.
System Log	Navigate to **System Logs** > **System Log**. You can place alert statements in your business rule which can write information to the log.
Debug Business	Navigate to **System Diagnostics** > **Session Debug** > **Debug Business Rule (Details)**. This debugging module displays the results business rules. Use this module to see if conditions are being met and values are being set as expected.

Debugging tools (continued)

Debugging tool	Description
Rule (Details)	
Alert Messages	There are several system functions that allow you to print messages to the page, the field or the log file. See Scripting alert, info, and error messages.
Business Rule Examples	Sometimes you can find what you're looking for in scripts others have written, including business rule error messages, or by building an OR query.
GlideRecord Information	This is the basic syntax used to query the database for information. See Querying tables in script. GlideRecord also includes aggregation support.

2. Variables

The next step is to gain some insight into the behavior of your business rule. For every action except an insert, you will more than likely use a query to get your record(s).

```
var rec = new GlideRecord('incident');
rec.addQuery('active',true);
rec.query();
while (rec.next()) {
 gs.print(rec.number + ' exists');
 }
```

To verify whether your query is actually returning records you can use *gs.addInfoMessage* to display information at the top of the screen.

```
var rec = new GlideRecord('incident');
rec.addQuery('active',true);
rec.query();
gs.addInfoMessage("This is rec.next: " + rec.next());
while (rec.next()) {
 gs.print(rec.number + ' exists');
 }
```

If your query returns no records you see the following:

```
This is rec.next: false
```

Use this technique to verify every variable within your business rule contains expected values.

Tip:
If necessary, break your script down into individual pieces and verify each piece works separate from the whole and then put them all back together one step at a time.

3. Locating information

The last step is to make sure you know where to find the information your rule is looking for.

In the ServiceNow application, one table can extend another table. This means when searching for information, you might need to query the parent table for the extended table's sys_id to find what you seek.

A good example is the sc_task table, which extends the task table. The script below queries the extended table (sc_task) for the current sys_id and then query the parent table (task) for records with the matching sys_id, and then prints out the work notes field.

```
var kids = new GlideRecord('sc_task');
kids.query();

gs.addInfoMessage("This is requested item number: " +
 current.number);
gs.print("This is the requested item number: " +
 current.number);

while (kids.next()) {
 var parents = new GlideRecord('task');
 parents.addQuery('sys_id', '=', kids.sys_id);
 parents.query();

 while(parents.next()) {
  gs.addInfoMessage("This is task number: " + parents.number);
  gs.print("This is task number: " + parents.number);
  gs.addInfoMessage("These are the work notes: " +
 parents.work_notes);
  gs.print("These are the work notes: " + parents.work_notes);
  }

 }
```

Debugging classifications

You must add a system property to enable classification debugging.

Debugging classifications

The resulting log entries list the name of each classifier that runs, along with all the names and values that are available to the criteria in the classifier. To log debugging information about classifications, add the following system property.

System Property	Description
glide.discovery.debug.ci_identification	Enables debugging information for process classification. • Type: true | false • Default Value: false • Location: Add to the System Properties [sys_properties] table

Field watcher

The field watcher tool tracks and displays all actions that the system performs on a selected form field.

Note:
Field watcher is not supported with Next Experience in Utah. For more information about supported features in Next Experience, see Considerations for activating Next Experience.

Administrators can use the field watcher to figure out what happens to the field and how the value of the field changes when an event such as the firing of a business rule or enforcement of a data policy, takes place. Administrators can also impersonate non-admin users to debug what happens when those users make changes on an instance. Only one field can be watched at a time. Non-admin users with the impersonator role have access to the field watcher feature.

How the field watcher works

The Field Watcher tool logs activity when any of the following events occur on a field:

- The default value is set on the field.
- User access rights for the field change due to an ACL or dictionary setting.
- A data policy prevents the value from being set.
- A reference qualifier query of the field value executes.
- A UI policy changes a field to or from read-only, visible, mandatory, or editable.
- A dependent value in another field restricts field choices.
- The value of the field is set or changed based on:
  - Assignment rules
  - Actions from an engine, such as the workflow engine
  - Business rules
  - User entries
  - Client scripts
  - UI actions

Note:
The field watcher works only on form fields. It cannot be used on list fields. Also, field watcher is not available on password-protected fields or encrypted fields. Field watcher is only available within the UI frame. The option to watch a field does not appear in the context menu if you open a record outside of the UI frame, for example, in a new tab.

Use field watcher

Access field-level debugging information using the field watcher.

Before you begin
Role required: none

Procedure

1. Navigate to the form for which you want to view field-level debugging information.
2. Activate field watcher by right-clicking any field label on a form and select **Watch - '<field name>'**.

   The debug icon ( ) appears next to the field label. From this point on, the field watcher records every action taken on the selected field. For example, if you are watching a *Priority* field, if the priority is changed from Moderate to Low and the record is updated, the field watcher will display information about that change.
3. View the field watcher log by clicking the debug icon.
   A new pane opens at the bottom of the screen, showing a field watcher tab. It may also show tabs for JavaScript Logging and JavaScript Debugger.
4. Click the **Field Watcher** tab, if needed.

5. Stop watching a field by right-clicking the field and selecting **Unwatch - <field name>**.
   To watch another field, right-click that field and select **Watch - <field name>**.
6. Clear the field watcher log by clicking the clear log button ().
7. Resize the field watcher pane by dragging the splitter bar up or down.
   Dragging the splitter bar to the bottom of the screen closes the field watcher pane. Reopen the pane by clicking the debug icon again.

Field watcher tab details

The field watcher displays field information and configuration options.

The left-side of the Field Watcher tab shows the following information for the watched field.

- **Table**: table to which the field belongs.
- **Element**: field label.
- **Type**: type of data stored in the field.
- **Dependent**: field on which the current field depends.
- **Reference**: table from which the field's value originates, if applicable.
- **Reference Qual**: reference qualifiers that may be restricting data on the field.
- **Attributes**: attributes on the field as specified in the dictionary entry for that field.

On the right-side of the Field Watcher tab, select the types of activity information you want to see for the selected field. Clear the check box for any type of information that is not needed.

Watching a hidden field

Administrators may need to watch a hidden field.

Procedure

1. Use the dictionary to determine the column name of the field.
2. Elevate privileges to the security_admin role.
3. Navigate to **System Definition** > **Scripts Background**.
4. In *Run script (JavaScript executed on server)*, enter the following command:

   ```
   gs.getSession ( ). setWatchField ( "hidden_field" ) ;
   ```

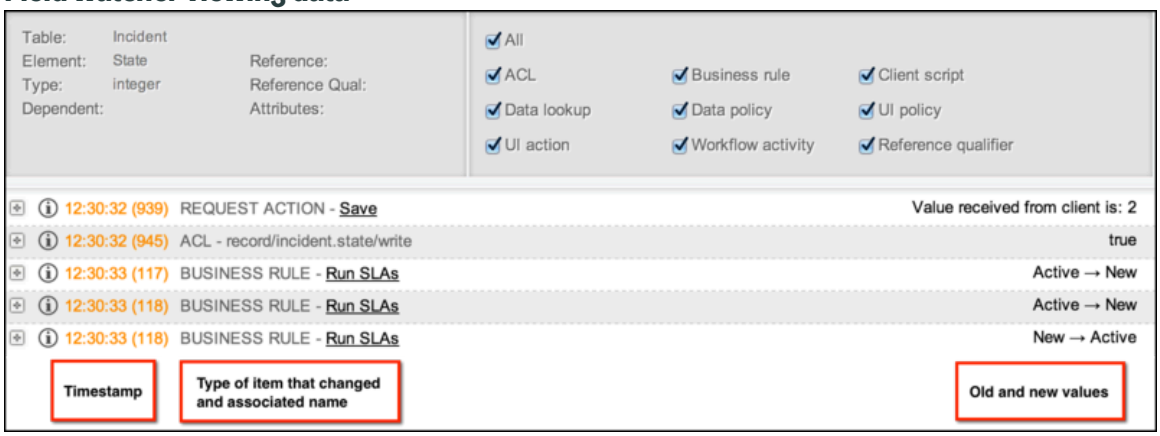
   Replace hidden_field with the column name of the hidden field.
5. Navigate to the form containing the missing field.

   The Field Watcher tab output contains information about the hidden field.

Viewing information for the watched field

When information for a watched field is changed and the record is updated, the field watcher tab displays relevant information at the bottom.

Field watcher viewing data



Field watcher information includes:

- **Timestamp**: time the field was changed using the HH:MM:SS (ms) format.
  - **Orange text**: server-side changes, such as ACLs.
  - **Blue text**: client-side changes, such as client scripts.
- **Type of object that changed the field and its associated name**: The type of item that changed on the field; for example, **CLIENT SCRIPT**, **BUSINESS RULE**, or **ACL**. In the case of scripts, business rules, or other configuration-type fields, field watcher displays the name of the script or business rule that changed the field, if any. Click the name to go directly to the record for that item.
- **Old and new values**: The old and new values for the field, if the value changed0. Field watcher does not record the value if it was inserted in the form by default at the time the record was created.
- **Additional information**: Call tracing information, such as the name of the script engine or workflow that changed the field. Click the plus icon to expand the selection.
  - **Orange text**: Indicates server-side activity.
  - **Blue text**: Indicates client-side activity.

Example: Watching the incident priority

The following example shows what happens to the **Priority** field on the incident form when both the **Impact** and **Urgency** fields change.

The Incident form has two client-side data lookups change the priority. Additionally, server-side ACLs and the data lookup engine fire when the record is saved. Finally, a client-side UI policy sets the **Priority** field back to read-only, which is the default setting.

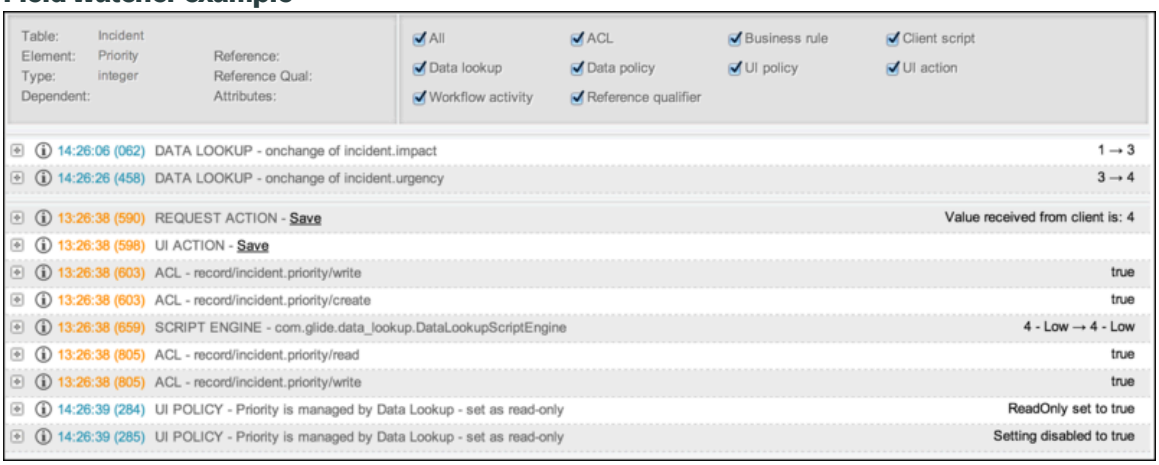
Watching the incident priority

Original values
• Priority:1 - Critical • Impact:1 - High • Urgency:1 - High
First Change

Watching the incident priority (continued)

Original values
1. The user changes the **Impact** value to **3 - Low**. **2.** The priority automatically changes to **3 - Moderate** based on the **Priority Lookup** data lookup definition used by default in ServiceNow incidents. **Note:** At this point, the record has not been saved.
Second Change
1. The user changes the **Urgency** value to **2 - Medium**. **2.** The priority automatically changes to **4 - Low** based on the same **Priority Lookup** data lookup definition. **3.** The user saves the record by right-clicking the form header and choosing **Save**.

Field watcher example



Note:
The values that change from 1 to 3, and then from 3 to 4, refer to the numerical values in the choice list.

Writing to the debug log

To write to the debug log in your client-side JavaScript, or UI policies, make a call to the global function `jslog()`.

An example of using `jslog()` in JavaScript:

```
function logData (r ) {
    lastLogDate  = r. responseXML. documentElement. getAttribute
 ( "last_log_entry" ) ; var items  = r. responseXML.
 getElementsByTagName ( "log" ) ;
    jslog ( "response=" + r. responseText ) ; }
```

Additionally, when client scripts run, the name of the client script and timing information is displayed. This can be useful in determining which scripts are running and whether they are impacting performance.

Debug UI policies

Enabling the *glide.ui.ui_policy_debug* property lets you monitor the processing of UI actions.

Here are some sample log events from an incident policy that sets fields to read-only if the incident_state is closed.

```
GlideFieldPolicy: Evaluating condition
GlideFieldPolicy:     incident_state (7) = 7 -> true
GlideFieldPolicy: --->>> TRUE
GlideFieldPolicy:    Setting opened_at disabled to true
GlideFieldPolicy:    Setting opened_by disabled to true
GlideFieldPolicy:    Setting closed_at disabled to true
GlideFieldPolicy:    Setting closed_by disabled to true
GlideFieldPolicy:    Setting company disabled to true
```

Access the JavaScript log

JavaScript that runs on the browser, such as client scripts, can include a call to jslog() to send information to the JavaScript Log. Users with the admin role can access this log.

Before you begin

Role required: admin

About this task

The steps to access the JavaScript debug window depend on which UI version you are using.

Note:
The JavaScript debug window is not supported with Next Experience in Utah. For more information about supported features in Next Experience, see Considerations for activating Next Experience.

Procedure

1. Open the JavaScript log by navigating to the appropriate location for your version of the UI.

Core UI	**a.** Click the gear icon in the banner frame. **b.** Click the **Developer** section. **c.** Toggle the **JavaScript Log and Field Watcher** switch.
UI15	**a.** Click the gear icon in the banner frame. **b.** Click **JavaScript Log and Field Watcher**.
UI11	Click the debug icon ( ) in the banner frame.

A new pane opens at the bottom of the screen. It shows the JavaScript Log tab and may also show the Field Watcher tab.

2. If needed, select the **JavaScript Log** tab.
3. Click the clear icon ( ) to clear the contents of the log, as needed.

JavaScript debug window

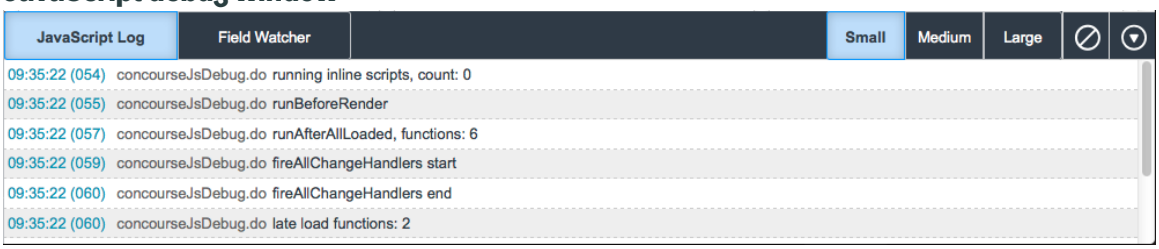
The JavaScript debug window appears in a bottom pane of the user interface when an administrator turns on debugging.

Note:
The JavaScript debug window is not supported with Next Experience in Utah. For more information about supported features in Next Experience, see Considerations for activating Next Experience.

Use the debug window to access these tools.

- JavaScript Log: JavaScript that runs on the browser, such as client scripts, can include a call to `jslog()` to send information to the JavaScript log.
- Field Watcher: a tool that tracks and displays all actions that the system performs on a selected form field.

JavaScript debug window



Using the JavaScript debug window

The JavaScript debug window enables access to the JavaScript Log and the Field Watcher tools.

Before you begin

Role required: admin

About this task

The steps to access the JavaScript debug window depend on which UI version you are using.

Note:
The JavaScript debug window is not supported with Next Experience in Utah. For more information about supported features in Next Experience, see Considerations for activating Next Experience.

Procedure

1. Open the JavaScript debug window by navigating to the appropriate location for your version of the UI.

Core UI	**a.** Click the gear icon in the banner frame. **b.** Click the **Developer** section. **c.** Toggle the **JavaScript Log and Field Watcher** switch.
UI15	**a.** Click the gear icon in the banner frame. **b.** Click **JavaScript Log and Field Watcher**.

UI11	Click the debug icon () in the banner frame.

The JavaScript debug window re-opens at the bottom of the screen. The tab that is currently active in the window is the last tab that was active when the window was closed.

2. Click a tab to use one of the debug window features.
   - JavaScript Log
   - Field Watcher

Related topics

Writing to the debug log

Field watcher

JS Code Coverage Debug

The JS Code Coverage Debug application allows administrators and application developers to log the server-side scripts triggered during a user session and then review which lines of code the system ran.

Users with the js_coverage_debugger role can debug server-side scripts without having to set breakpoints or review onscreen debug messages. Instead, the system saves script usage data in the JavaScript Code Coverage [sys_js_code_coverage] table. Each JavaScript Code Coverage record contains:

- The user session that called the script.
- The script record the system called identified by table, sys_id, and script field.
- The script record the system called identified by type and name.
- The transaction that called the script.
- The start time of the transaction.
- The contents of the script field highlighted to indicate which lines the system ran.

Note:
The JS Code Coverage Debug application doesn't log information for client-side scripts.

Sample code coverage highlighting

Script

```
1  var build = gs.getProperty("glide.buildname");
2  if (GlideStringUtil.notNil(build))
3      build = build.substring(0,1).toLowerCase();
4
5  var collision = new GlideRecord("sys_embedded_help_content");
6  collision.addActiveQuery();
7  collision.addQuery("page", current.page);
8  collision.addQuery("modifier", current.modifier);
9  collision.addQuery("product", current.product);
10 if (current.isValidRecord() && GlideStringUtil.notNil(current.sys_id))
11     collision.addQuery("sys_id", "!=", current.sys_id);
12
13 if (GlideStringUtil.notNil(current.qualifier))
14     collision.addQuery("qualifier", current.qualifier);
15 else
16     collision.addNullQuery("qualifier");
17
18 if (current.version == "all" && GlideStringUtil.notNil(build))
19     collision.addQuery("version", build).addOrCondition("version", "all");
20 else
21     collision.addQuery("version", current.version);
22
23 collision.query();
24 if (collision.next()) {
25     if (collision.canWrite())
26         gs.addErrorMessage(gs.getMessage("Embedded Help Composite Key
Validation"));
```

JS Code Coverage highlighting

The JS Code Coverage application highlights script fields to indicate whether the system ran or skipped each line.

Sample code highlighting

Script

```
1  var build = gs.getProperty("glide.buildname");
2  if (GlideStringUtil.notNil(build))
3      build = build.substring(0,1).toLowerCase();
4
5  var collision = new GlideRecord("sys_embedded_help_content");
6  collision.addActiveQuery();
7  collision.addQuery("page", current.page);
8  collision.addQuery("modifier", current.modifier);
9  collision.addQuery("product", current.product);
10 if (current.isValidRecord() && GlideStringUtil.notNil(current.sys_id))
11     collision.addQuery("sys_id", "!=", current.sys_id);
12
13 if (GlideStringUtil.notNil(current.qualifier))
14     collision.addQuery("qualifier", current.qualifier);
15 else
16     collision.addNullQuery("qualifier");
17
18 if (current.version == "all" && GlideStringUtil.notNil(build))
19     collision.addQuery("version", build).addOrCondition("version", "all");
20 else
21     collision.addQuery("version", current.version);
22
23 collision.query();
24 if (collision.next()) {
25     if (collision.canWrite())
26         gs.addErrorMessage(gs.getMessage("Embedded Help Composite Key
Validation"));
```

The color of the highlight indicates how the system evaluated the code line.

Meaning of code highlighting

Highlight color	Meaning
Green	This is an executable line of code that the system ran during the session.
Red	This is an executable line of code that the system skipped for some reason. The system may have skipped an executable line of code because the necessary script conditions were not met or because the script function was never called. You may want to use the Script Debugger to determine why the system skipped the line of executable code.
Gray	This is a non-executable line of code such as white space, code comment, or a portion of an expression split across multiple lines that cannot run on its own.

Administrators and application developers can use this information to conduct more targeted debugging activities such as using the Script Debugger to determine why script conditions are not being met.

Activate JS Code Coverage Debug

You can activate the JS Code Coverage Debug plugin (com.glide.js.coverage) if you have the admin role.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. Find the plugin using the filter criteria and search bar.

   You can search for the plugin by its name or ID. If you cannot find a plugin, you might have to request it from ServiceNow personnel.
3. Select **Install** to start the installation process.

   Note:
   When domain separation and delegated admin are enabled in an instance, the administrative user must be in the **global** domain. Otherwise, the following error appears: `Application installation is unavailable because another operation is running: Plugin Activation for <plugin name>.`

   You will see a message after installation is completed. For information about the components installed with a plugin, see Find components installed with an application.

What to do next

To see the components the plugin installed, refresh the plugin form and select the **Plugin Files** related list.

Debug with JS Code Coverage Debug

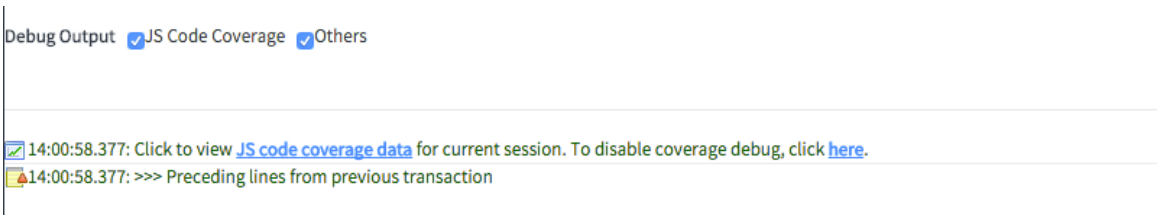
Use JS Code Coverage Debug to record a user session and then review which server-side scripts and lines of code the system ran.

Before you begin

Role required: js_coverage_debugger or admin

Procedure

1. Navigate to **All** > **JS Code Coverage Debug** > **Enable Coverage**.
   The system logs which server-side scripts and code lines the system runs as well as displays session debug messages in the JS Code Coverage namespace.



2. Navigate to the table or page whose logic you want to test.

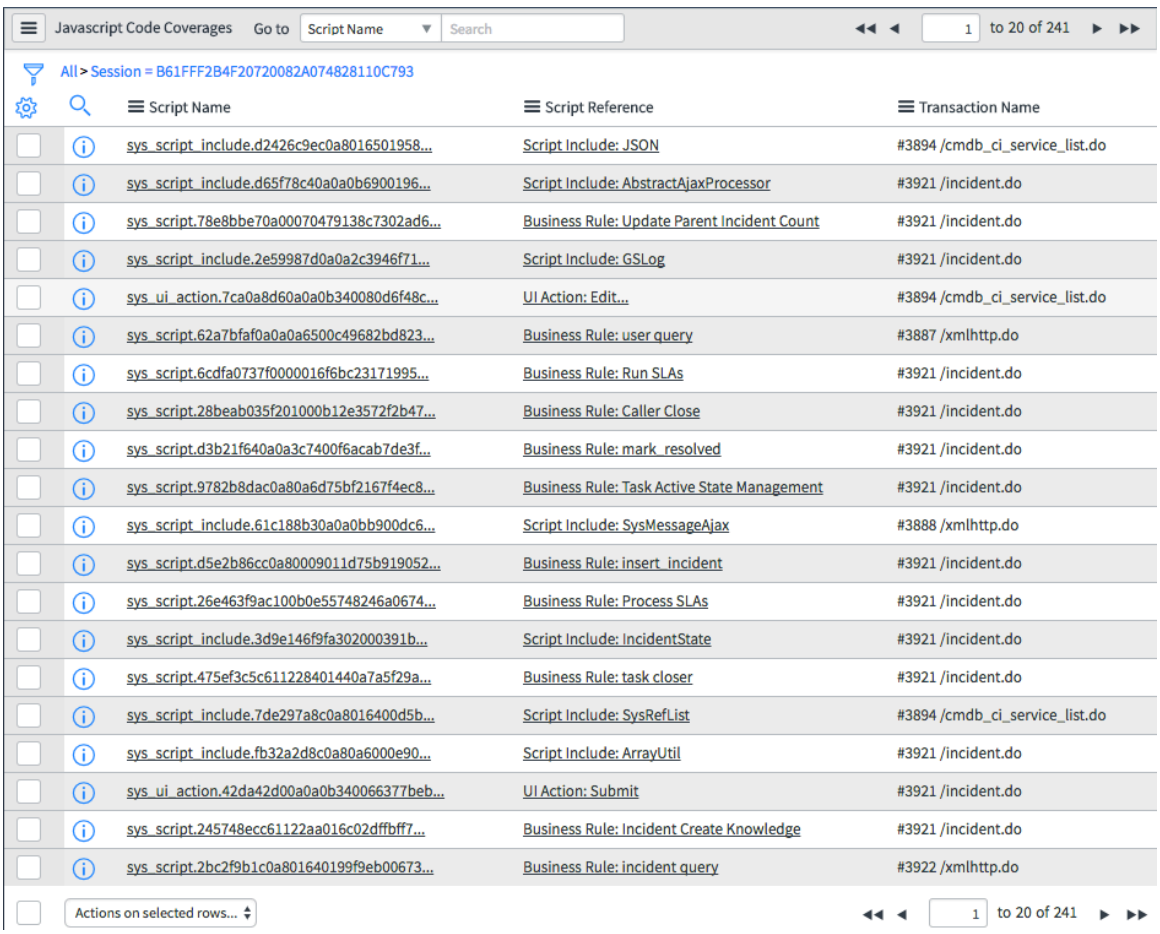
   Example
   For example, navigate to **Incident** > **Create New**.

3. Trigger the server-side script or scripts you want to test.

   Example
   For example, create an incident with an associate CI item to test several business rules.

4. When you have completed testing, navigate to **JS Code Coverage Debug** > **Disable Coverage**.
   The system stops logging script and code lines run.

5. Navigate to **JS Code Coverage Debug** > **Coverage Data**.
   The system displays the list of coverage data associated with the current user session.

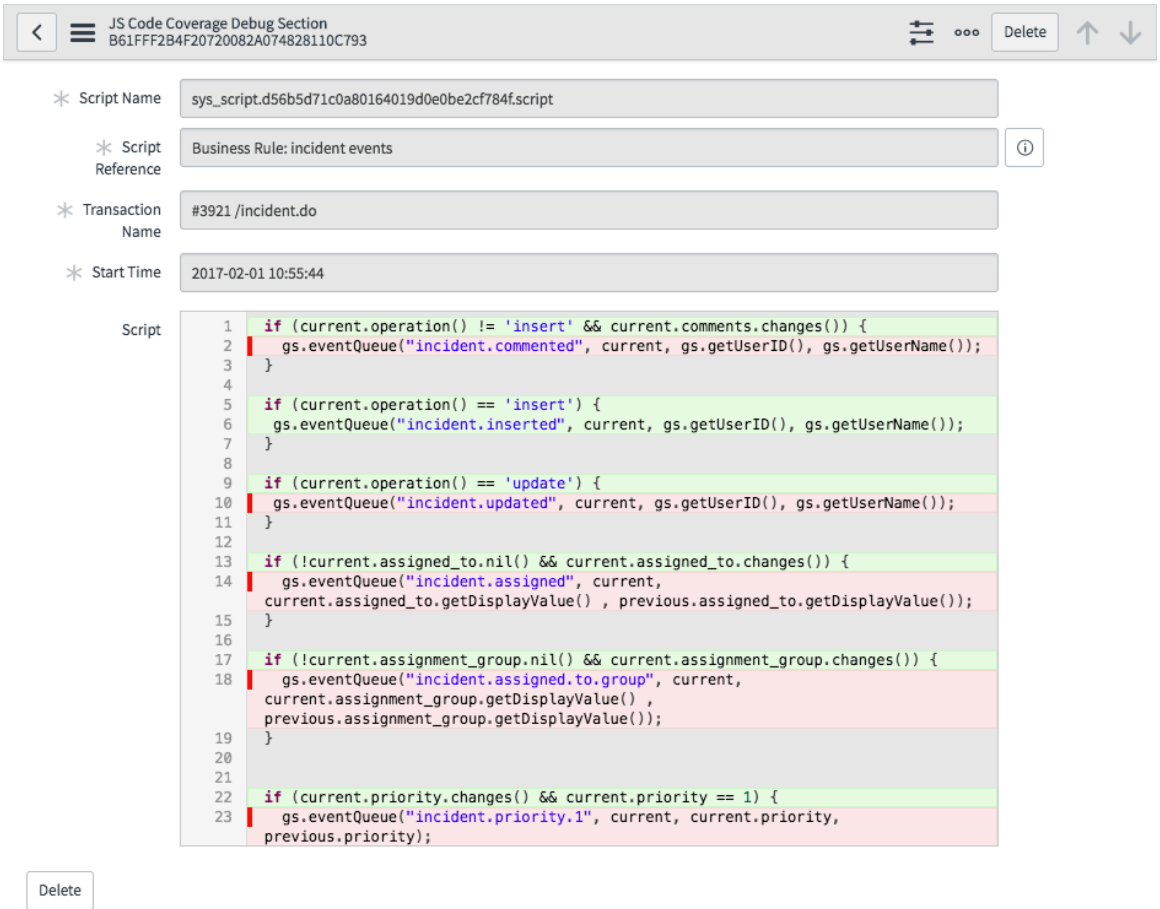


6. Select the script or transaction you want to review.

JavaScript Code Coverage fields

Field	Description
Script Name	Displays the script run by table name, sys_id value, and script field.
Script Reference	Displays the script run by script type and name.
Transaction Name	Displays the transaction that called the script by thread ID and URI.

Example
For example, select the **Script Reference** Business Rule: incident events.
The system displays the JS Code Coverage Debug record.



7. Review the **Script** field to determine which lines of code the system ran.

Example
For example, the business rule added the incident.inserted event to the event queue.

Result
You determine which lines of code the system ran.

What to do next
Use the code coverage information to do more targeted debugging activities such as set breakpoints and review variable values with the Script Debugger.

Packages Call Removal tool

The Packages Call Removal Tool provides modules to identify fields that might contain scripts, find scripts that contain Packages calls to ServiceNow Java classes, and to examine proposed script changes that eliminate those Packages calls.

Note:
This tool is not activated by default, and is only available to users with the admin role.

Packages calls to ServiceNow Java classes will be prevented in a future release. The Packages Call Removal tool helps prepare your instance to use the new API and includes the following scripts and pages:

- The Find Packages Fields script scans scripts for Packages calls to ServiceNow Java classes.
- The Find Packages Calls script proposes changes that remove Packages calls or replaces them with GlideScriptable names.
- The Packages Call Items page lists and enables you to work on proposed changes to scripts.

The tool might generate errors as it tries to generate preferred, scriptable alternatives for Packages calls to ServiceNow Java classes.

Note:
Create an update set before migrating the changes that result from running the Packages Call Removal Tool. For information, see Get started with update sets.

Activate the Packages Call Removal Tool

You must activate the Packages Call Removal Tool plugin to access the tool.

Before you begin

Role required: admin.

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. Find the Packages Call Removal Tool plugin using the filter criteria and search bar.

   You can search for the plugin by its name or ID. If you cannot find a plugin, you might have to request it from ServiceNow personnel.
3. Select **Install** to start the installation process.

   Note:
   When domain separation and delegated admin are enabled in an instance, the administrative user must be in the **global** domain. Otherwise, the following error appears: `Application installation is unavailable because another operation is running: Plugin Activation for <plugin name>.`

   You will see a message after installation is completed. For information about the components installed with a plugin, see Find components installed with an application.

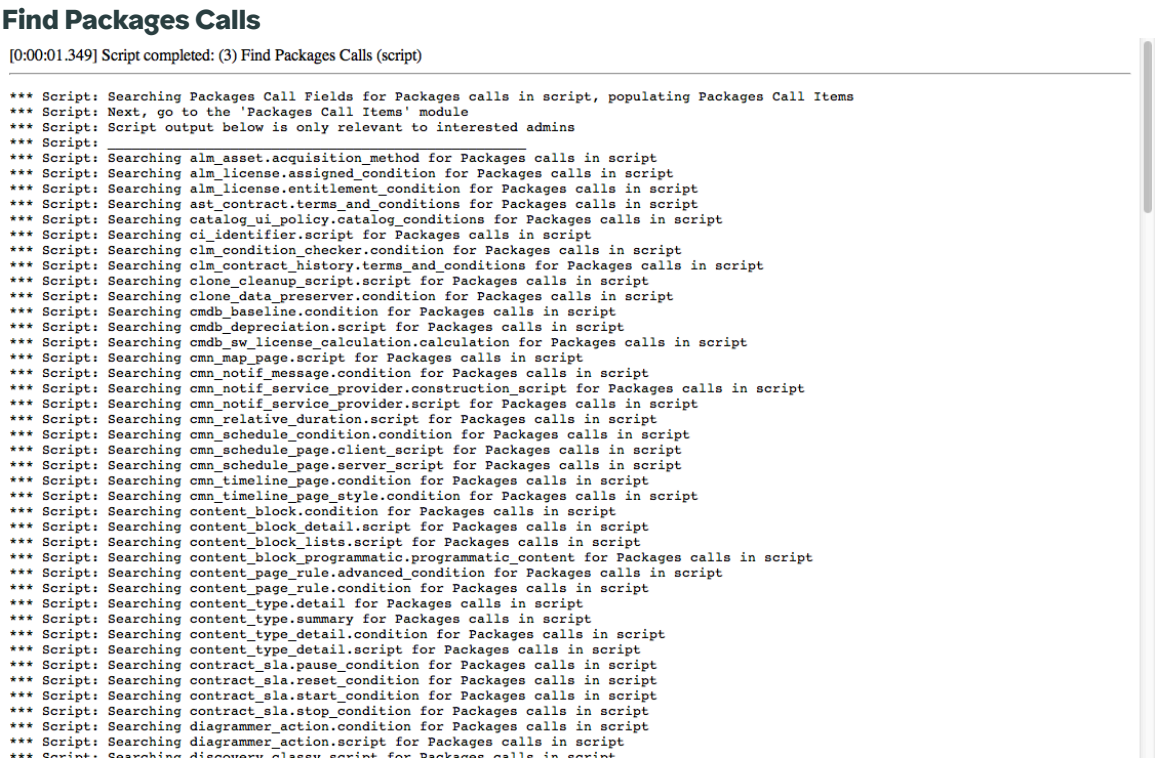
Find a Packages call

After you run the Find Packages Fields script to define the list of fields to search for Packages calls, run the Find Packages Calls script to generate the list of fields with Packages calls to ServiceNow Java classes. The script also proposes changes.

Procedure

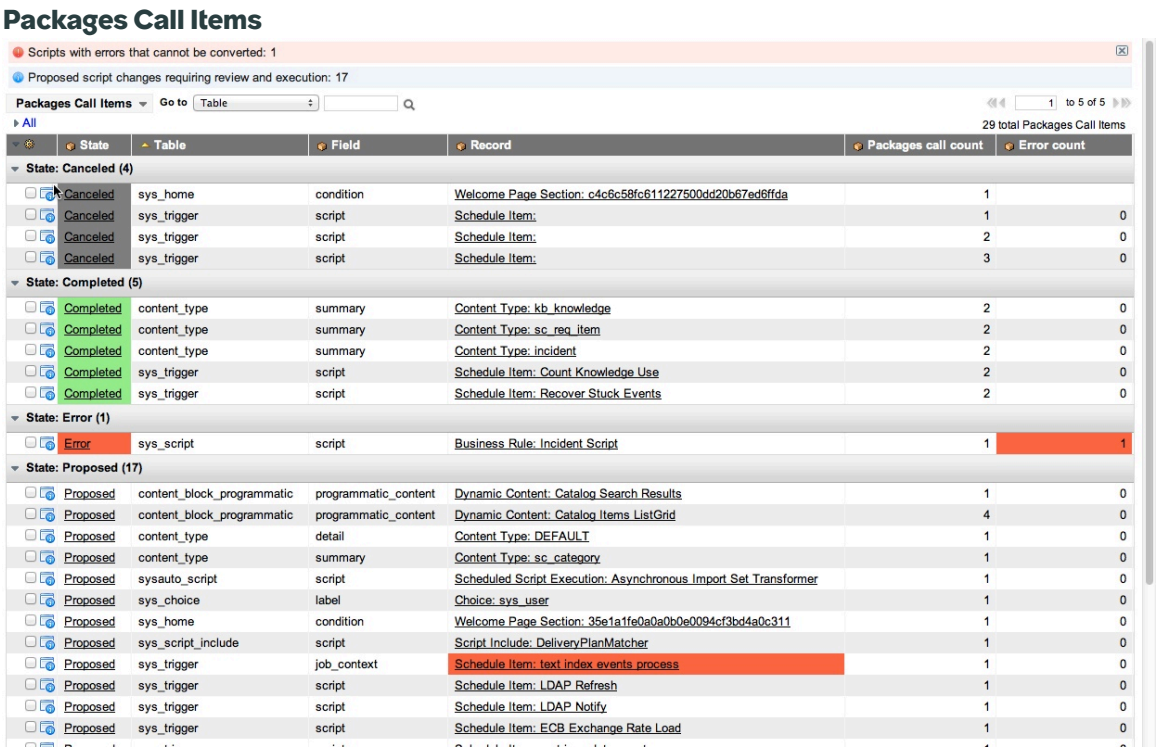
1. Click **(3) Find Packages Calls (script)** to execute a script that searches the fields for packages calls and populates the Packages Call Items list with proposed changes.

   Example

2. Click **(4) Packages Call Items** to display the list of affected fields on the Packages Call Items page.
   The Packages Call Items page lists the items with Packages calls, shows each item's current state, the table that contains the field, the affected record, the number of Packages calls contained in the field's script, and the number of errors that occurred when the proposed script was generated.

Example



3. On the Packages Call Items page, click a record for any of the listed items to open the form and revise as described in Replacing Packages Calls.

   As you work through the list, the **State** of the items are updated and the items are grouped by State:

   - **Proposed:** A proposed revision exists for the field.
   - **Error**(red): One or more errors occurred when the proposed script was generated.
   - **Rejected**(gray): A proposed change has been rejected.
   - **Completed**(green): The field has been successfully revised to remove Packages calls to ServiceNow Java classes.
   - **Canceled** (gray): Since the proposed change was generated, either the original script has been changed to no longer require modification, or the original record no longer exists.

Glide object name

The names of the new Glide objects that replace Packages calls are derived from the Java package name used in the Packages call.

About this task

Although the tool automatically substitutes the appropriate new scriptable name for each Packages call it encounters, in some circumstances it can be useful to know how to manually replace a Packages call with its new scriptable object equivalent. Use the steps below to determine the new script object name from the Packages call name. The replacement objects include the same methods and properties as the objects they replace.

To determine the new object name:

Procedure

1. Note the third term in the Java package name.
   This term is usually glide, but is sometimes snc or glideapp.
2. Drop all of the prefix terms, leaving only the last.
   For example, `Packages.com.glide.monitor.AbstractBucketCollector` becomes AbstractBucketCollector.
3. Capitalize the first letter of the term noted in the first step above and add it to the front of the term defined in step 2.
   For example, `Packages.com.glide.monitor.AbstractBucketCollector` becomes GlideAbstractBucketCollector and `Packages.com.snc.cmdb.BaselineCMDB` becomes SncBaselineCMDB.
4. Verify that the name is valid by executing a gs.print() command in Scripts Background, specifying only the name with no quotes.
   For example:

   ```
   gs.print( SncBaselineCMDB );
   ```

   *** Script: `[JavaClass com.snc.cmdb.BaselineCMDB]`

   Note that there are exceptions to this rule, such as "Glide", which replaces "Packages.com.glide.Glide", "TestExtension", which replaces "Packages.com.glide.junit.misc.TestExtension"; and "UINotification", which replaces "Packages.com.glide.ui.UINotification".

Glide object replacement list

This table lists the Glide classes and the Packages calls they replace.

Note:
The publication of this list does not imply that these scriptable objects are for use by customers, consultants, and partners. The use of the Glide prefix does not imply that these scriptable objects are in the same category as or have the same status as objects such as GlideRecord. Except where documentation is provided in the API Reference, these undocumented APIs are not intended for general use, and ServiceNow, Inc., does not make any commitment to document them, answer questions about them, or maintain them indefinitely in their current form. Over time, ServiceNow, Inc., intends to migrate a subset of the functionality represented by these objects into the documented API and remove the rest.

GlideScriptable object replacement list

GlideScriptable Class	Packages Call
Glide	Packages.com.glide.Glide
GlideAbstractBucketCollector	Packages.com.glide.monitor.AbstractBucketCollector
GlideAbstractDomainProvider	Packages.com.glide.db.domain.AbstractDomainProvider
GlideAbstractExecutionPlan	Packages.com.glide.execution_plan.AbstractExecutionPlan
GlideAbstractListener	Packages.com.glide.listener.AbstractListener
GlideAbstractRenderer	Packages.com.glide.ui.portal.AbstractRenderer
GlideAction	Packages.com.glide.script.Action
GlideActionManager	Packages.com.glide.ui.action.ActionManager

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideAJAXScheduleItem	Packages.com.glide.schedules.AJAXScheduleItem
GlideAJAXSchedulePage	Packages.com.glide.schedules.AJAXSchedulePage
GlideAlertActions	Packages.com.glide.alerts.AlertActions
GlideappAbstractChoiceListQuestion	Packages.com.glideapp.questionset.AbstractChoiceListQuestio
GlideappADSILoader	Packages.com.glideapp.ecc.ADSILoader
GlideappAJAXMapPage	Packages.com.glideapp.google_maps.AJAXMapPage
GlideappCalculationHelper	Packages.com.glideapp.servicecatalog.CalculationHelper
GlideappCart	Packages.com.glideapp.servicecatalog.Cart
GlideappCartItem	Packages.com.glideapp.servicecatalog.CartItem
GlideappCatalogCategoryBatcher	Packages.com.glideapp.servicecatalog.CatalogCategoryBatche
GlideappCatalogItem	Packages.com.glideapp.servicecatalog.CatalogItem
GlideappCategory	Packages.com.glideapp.servicecatalog.Category
GlideappCategoryPopper	Packages.com.glideapp.servicecatalog.CategoryPopper
GlideappCatItemPopper	Packages.com.glideapp.servicecatalog.CatItemPopper
GlideappChartParameters	Packages.com.glideapp.chart.ChartParameters
GlideappChatRoom	Packages.com.glideapp.live.db.ChatRoom
GlideappChatRoom$Error	Packages.com.glideapp.live.db.ChatRoom.Error
GlideappCheckBoxQuestion	Packages.com.glideapp.questionset.CheckBoxQuestion
GlideappCMDBHelper	Packages.com.glideapp.ecc.CMDBHelper
GlideappCMDBSoftwareHelper	Packages.com.glideapp.ecc.CMDBSoftwareHelper
GlideappContainerAwareQuestionSet	Packages.com.glideapp.questionset.ContainerAwareQuestionS
GlideappContextDiagramProcessor	Packages.com.glideapp.workflow.ui.ContextDiagramProcessor
GlideappDateQuestion	Packages.com.glideapp.questionset.DateQuestion
GlideappDateTimeQuestion	Packages.com.glideapp.questionset.DateTimeQuestion
GlideappDeliveryPlan	Packages.com.glideapp.servicecatalog.DeliveryPlan
GlideappECCInputMessage	Packages.com.glideapp.ecc.ECCInputMessage
GlideappECCOutputMessage	Packages.com.glideapp.ecc.ECCOutputMessage
GlideappECCQueueConnector	Packages.com.glideapp.ecc.ECCQueueConnector
GlideappECCQueueProcessor	Packages.com.glideapp.ecc.ECCQueueProcessor
GlideappECCResponseMessage	Packages.com.glideapp.ecc.ECCResponseMessage
GlideappExpandableText	Packages.com.glideapp.live_feed.HTMLTransformers.Expandab
GlideappExpertPanelCatalogOrder	Packages.com.glideapp.servicecatalog.ExpertPanelCatalogOrd
GlideappFixes	Packages.com.glideapp.servicecatalog.Fixes
GlideappHome	Packages.com.glideapp.home.Home

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideappHomePage	Packages.com.glideapp.home.HomePage
GlideappHomePageFactory	Packages.com.glideapp.home.HomePageFactory
GlideappIECC	Packages.com.glideapp.ecc.IECC
GlideappIOUpgrade	Packages.com.glideapp.servicecatalog.IOUpgrade
GlideappItemOptionsQuestionSet	Packages.com.glideapp.servicecatalog.ItemOptionsQuestionSe
GlideappJMSECCReceiver	Packages.com.glideapp.jms.JMSECCReceiver
GlideappJMSECCSender	Packages.com.glideapp.jms.JMSECCSender
GlideappKBIncludes	Packages.com.glideapp.knowledge.KBIncludes
GlideApplication	Packages.com.glide.sys.Application
GlideApplicationModule	Packages.com.glide.processors.ApplicationModule
GlideappListCollectorQuestion	Packages.com.glideapp.questionset.ListCollectorQuestion
GlideappLiveFeedEventHandler	Packages.com.glideapp.live_feed.LiveFeedEventHandler
GlideappLiveFeedJournalWriter	Packages.com.glideapp.live_feed.LiveFeedJournalWriter
GlideappLiveFeedUIAction	Packages.com.glideapp.live_feed.LiveFeedUIAction
GlideappLiveProfile	Packages.com.glideapp.live_common.LiveProfile
GlideappLiveUtils	Packages.com.glideapp.live.LiveUtils
GlideappLookupSelectQuestion	Packages.com.glideapp.questionset.LookupSelectQuestion
GlideappMessageTag	Packages.com.glideapp.live_feed.MessageTag
GlideappOrderGuide	Packages.com.glideapp.servicecatalog.OrderGuide
GlideappProcessQueue	Packages.com.glideapp.ecc.ccmdb.ProcessQueue
GlideappQuestion	Packages.com.glideapp.questionset.Question
GlideappQuestionChoice	Packages.com.glideapp.questionset.QuestionChoice
GlideappQueueHelper	Packages.com.glideapp.ecc.QueueHelper
GlideappQueueReader	Packages.com.glideapp.ecc.QueueReader
GlideappReferenceQuestion	Packages.com.glideapp.questionset.ReferenceQuestion
GlideappRequestItemWorkflow	Packages.com.glideapp.servicecatalog.RequestItemWorkflow
GlideappRequestNew	Packages.com.glideapp.servicecatalog.RequestNew
GlideappScriptHelper	Packages.com.glideapp.servicecatalog.ScriptHelper
GlideappSecurityMask	Packages.com.glideapp.servicecatalog.SecurityMask
GlideappSequencedQuestionSet	Packages.com.glideapp.questionset.SequencedQuestionSet
GlideappTaskApprovalHelper	Packages.com.glideapp.servicecatalog.TaskApprovalHelper
GlideappTimeAgo	Packages.com.glideapp.live_feed.TimeAgo
GlideappUpdateVersion	Packages.com.glideapp.version.UpdateVersion
GlideappUpgradeQuestions	Packages.com.glideapp.survey.UpgradeQuestions

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideappValveProcessor	Packages.com.glideapp.servicecatalog.valve.ValveProcessor
GlideappVariable	Packages.com.glideapp.servicecatalog.variables.Variable
GlideappVariablePoolQuestionSet	Packages.com.glideapp.servicecatalog.variables.VariablePoolQu
GlideappWizardIntercept	Packages.com.glideapp.wizard.WizardIntercept
GlideappWMILoader	Packages.com.glideapp.ecc.WMILoader
GlideappWorkflow	Packages.com.glideapp.workflow.Workflow
GlideappWorkflowHelper	Packages.com.glideapp.workflow.WorkflowHelper
GlideappYesNoQuestion	Packages.com.glideapp.questionset.YesNoQuestion
GlideAQueryExplanation	Packages.com.glide.db.explain.AQueryExplanation
GlideArchiver	Packages.com.glide.db.auxiliary.Archiver
GlideArchiveRecord	Packages.com.glide.db.auxiliary.ArchiveRecord
GlideArchiveRestore	Packages.com.glide.db.auxiliary.ArchiveRestore
GlideArchiveStatus	Packages.com.glide.db.auxiliary.ArchiveStatus
GlideArchiveTable	Packages.com.glide.db.auxiliary.ArchiveTable
GlideARecurrence	Packages.com.glide.schedule.recurrence.ARecurrence
GlideAttachmentIndexDocument	Packages.com.glide.lucene.attachments.AttachmentIndexDocu
GlideAttachmentIndexTypes	Packages.com.glide.lucene.attachments.AttachmentIndexTypes
GlideAttributes	Packages.com.glide.util.GlideAttributes
GlideAuditDelete	Packages.com.glide.audit.AuditDelete
GlideAuditor	Packages.com.glide.script.Auditor
GlideAutomationEncrypter	Packages.com.glide.util.AutomationEncrypter
GlideBaseTag	Packages.com.glide.ui.jelly.tags.BaseTag
GlideBootstrap	Packages.com.glide.db.impex.Bootstrap
GlideBoundedIntProperty	Packages.com.glide.util.BoundedIntProperty
GlideCacheManager	Packages.com.glide.sys.cache.CacheManager
GlideCalendar	Packages.com.glide.schedule.GlideCalendar
GlideCalendarWeekEntry	Packages.com.glide.calendar.GlideCalendarWeekEntry
GlideCanceledUITransaction	Packages.com.glide.ui.CanceledUITransaction
GlideCascadeFromDelete	Packages.com.glide.db.CascadeFromDelete
GlideCatalogCloneWorker	Packages.com.glide.catalog.cloner.CatalogCloneWorker
GlideChannel	Packages.com.glide.channel.Channel
GlideChannelManager	Packages.com.glide.channel.ChannelManager
GlideChannelMessage	Packages.com.glide.channel.ChannelMessage
GlideChartFieldColors	Packages.com.glide.ui.chart.dataset.ChartFieldColors

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideChartGeneratorFactory	Packages.com.glide.ui.chart.ChartGeneratorFactory
GlideChartUtil	Packages.com.glide.ui.chart.dataset.ChartUtil
GlideChartValue	Packages.com.glide.ui.chart.dataset.ChartValue
GlideChecksum	Packages.com.glide.util.Checksum
GlideChoice	Packages.com.glide.choice.Choice
GlideChoiceList	Packages.com.glide.choice.ChoiceList
GlideChoiceListGenerator	Packages.com.glide.choice.ChoiceListGenerator
GlideChoiceListSet	Packages.com.glide.choice.ChoiceListSet
GlideChoiceListUpdateSaver	Packages.com.glide.update.saver.ChoiceListUpdateSaver
GlideClientBrowserTimes	Packages.com.glide.client_transaction.ClientBrowserTimes
GlideClientNetworkTimes	Packages.com.glide.client_transaction.ClientNetworkTimes
GlideClusterMessage	Packages.com.glide.cluster.ClusterMessage
GlideClusterState	Packages.com.glide.cluster.ClusterState
GlideClusterSynchronizer	Packages.com.glide.cluster.ClusterSynchronizer
GlideCMSLinkHelper	Packages.com.glide.cms.CMSLinkHelper
GlideCMSPageLink	Packages.com.glide.cms.CMSPageLink
GlideCollectionEnumerator	Packages.com.glide.util.CollectionEnumerator
GlideCollectionQueryCalculator	Packages.com.glide.ui.CollectionQueryCalculator
GlideCollisionDetector	Packages.com.glide.update.collisions.CollisionDetector
GlideColumnAttributes	Packages.com.glide.db.impex.ColumnAttributes
GlideCompanyResolver	Packages.com.glide.misc.CompanyResolver
GlideCompiler	Packages.com.glide.script.Compiler
GlideCompositeElement	Packages.com.glide.db.CompositeElement
GlideConfiguration	Packages.com.glide.notification.Configuration
GlideContentConfig	Packages.com.glide.cms.ContentConfig
GlideContentPage	Packages.com.glide.cms.ContentPage
GlideContentSite	Packages.com.glide.cms.ContentSite
GlideContentType	Packages.com.glide.cms.ContentType
GlideContextMenu	Packages.com.glide.db_context_menu.ContextMenu
GlideContextMenuItem	Packages.com.glide.db_context_menu.ContextMenuItem
GlideContextualSecurityManager	Packages.com.glide.sys.security.ContextualSecurityManager
GlideController	Packages.com.glide.script.GlideController
GlideConverter	Packages.com.glide.currency.Converter
GlideCookieMan	Packages.com.glide.ui.CookieMan

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideCounter	Packages.com.glide.util.Counter
GlideCredentials	Packages.com.glide.communications.crypto.Credentials
GlideCryptoService	Packages.com.glide.security.CryptoService
GlideCSVExporter	Packages.com.glide.generators.CSVExporter
GlideCustomerScriptFixer	Packages.com.glide.script.api.CustomerScriptFixer
GlideDatabaseVerifier	Packages.com.glide.db.DatabaseVerifier
GlideDatabaseViewLink	Packages.com.glide.database_views.DatabaseViewLink
GlideDataSource	Packages.com.glide.db.impex.datasource.DataSource
GlideDate	Packages.com.glide.glideobject.GlideDate
GlideDateTime	Packages.com.glide.glideobject.GlideDateTime
GlideDateUtil	Packages.com.glide.util.DateUtil
GlideDBAction	Packages.com.glide.db.DBAction
GlideDBAggregateQuery	Packages.com.glide.db.DBAggregateQuery
GlideDBAggregateUtil	Packages.com.glide.db.DBAggregateUtil
GlideDBCategoryDebug	Packages.com.glide.secondary_db_pools.DBCategoryDebug
GlideDBChangeManager	Packages.com.glide.db.change.DBChangeManager
GlideDBCompositeAction	Packages.com.glide.db.DBCompositeAction
GlideDBConfiguration	Packages.com.glide.db.DBConfiguration
GlideDBConfigurationManager	Packages.com.glide.db.DBConfigurationManager
GlideDBConfigurationManagerEventHandler	Packages.com.glide.db.DBConfigurationManagerEventHandler
GlideDBConfigurationParms	Packages.com.glide.db.DBConfigurationParms
GlideDBConfigurationV2Migrator	Packages.com.glide.db.DBConfigurationV2Migrator
GlideDBConnection	Packages.com.glide.db.pool.DBConnection
GlideDBConnectionPool	Packages.com.glide.db.pool.DBConnectionPool
GlideDBConnectionPooler	Packages.com.glide.db.pool.DBConnectionPooler
GlideDBDelete	Packages.com.glide.db.DBDelete
GlideDBI	Packages.com.glide.db.DBI
GlideDBImageProvider	Packages.com.glide.db_image.DBImageProvider
GlideDBIMySQL	Packages.com.glide.db.rdbms.mysql.DBIMySQL
GlideDBIndex	Packages.com.glide.db.DBIndex
GlideDBKeyStoreFactory	Packages.com.glide.certificates.DBKeyStoreFactory
GlideDBMacro	Packages.com.glide.ui.jelly.tags.form.DBMacro
GlideDBMicroStats	Packages.com.glide.db.DBMicroStats
GlideDBMultiTargetAction	Packages.com.glide.db.DBMultiTargetAction

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideDBObjectManager	Packages.com.glide.db.DBObjectManager
GlideDBObjectToken	Packages.com.glide.db.DBObjectToken
GlideDBPoolTest	Packages.com.glide.secondary_db_pools.DBPoolTest
GlideDBPropertiesConfig	Packages.com.glide.db.DBPropertiesConfig
GlideDBQuery	Packages.com.glide.db.DBQuery
GlideDBTypes	Packages.com.glide.db.DBTypes
GlideDBUpdate	Packages.com.glide.db.DBUpdate
GlideDBUtil	Packages.com.glide.db.DBUtil
GlideDBView	Packages.com.glide.db.DBView
GlideDebugEvaluator	Packages.com.glide.jsdebug.DebugEvaluator
GlideDefaultUpdateSaver	Packages.com.glide.update.saver.DefaultUpdateSaver
GlideDiagram	Packages.com.glide.diagrammer.Diagram
GlideDiagramAction	Packages.com.glide.diagrammer.DiagramAction
GlideDiagramEdge	Packages.com.glide.diagrammer.DiagramEdge
GlideDiagramElement	Packages.com.glide.diagrammer.DiagramElement
GlideDiagramNode	Packages.com.glide.diagrammer.DiagramNode
GlideDistUpgradeRunner	Packages.com.glide.dist.upgrade.runner.DistUpgradeRunner
GlideDocument	Packages.com.glide.util.GlideDocument
GlideDomain	Packages.com.glide.db.domain.Domain
GlideDomainDisplay	Packages.com.glide.db.domain.DomainDisplay
GlideDomainHierarchy	Packages.com.glide.db.domain.DomainHierarchy
GlideDomainNumberProvider	Packages.com.glide.db.domain.DomainNumberProvider
GlideDomainPathDisplay	Packages.com.glide.db.domain.DomainPathDisplay
GlideDomainPathProvider	Packages.com.glide.db.domain.DomainPathProvider
GlideDomainSpoolProvider	Packages.com.glide.db.domain.DomainSpoolProvider
GlideDomainSupport	Packages.com.glide.db.domain.DomainSupport
GlideDomainTree	Packages.com.glide.db.domain.DomainTree
GlideDomainUtil	Packages.com.glide.db.domain.DomainUtil
GlideDomainValidator	Packages.com.glide.db.domain.DomainValidator
GlideDuration	Packages.com.glide.glideobject.GlideDuration
GlideECBDownloader	Packages.com.glide.currency.ECBDownloader
GlideECCQueueTransformer	Packages.com.glide.db.impex.ECCQueueTransformer
GlideElement	Packages.com.glide.script.GlideElement
GlideElementDescriptor	Packages.com.glide.db.ElementDescriptor

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideElementIterator	Packages.com.glide.util.ElementIterator
GlideElementUserImage	Packages.com.glide.script.glide_elements.GlideElementUserIma
GlideElementXMLSerializer	Packages.com.glide.script.GlideElementXMLSerializer
GlideEmail	Packages.com.glide.notification.Email
GlideEmailAction	Packages.com.glide.notification.outbound.EmailAction
GlideEmailFormatter	Packages.com.glide.notification.outbound.EmailFormatter
GlideEmailInbound	Packages.com.glide.notification.inbound.EmailInbound
GlideEmailOutbound	Packages.com.glide.notification.outbound.EmailOutbound
GlideEmailReader	Packages.com.glide.notification.inbound.EmailReader
GlideEmailSender	Packages.com.glide.notification.outbound.EmailSender
GlideEmailWatermark	Packages.com.glide.notification.EmailWatermark
GlideEmitter	Packages.com.glide.ui.jelly.Emitter
GlideEncrypter	Packages.com.glide.util.Encrypter
GlideEncryptionContext	Packages.com.glide.sys.EncryptionContext
GlideEncryptionContextCipher	Packages.com.glide.sys.security.EncryptionContextCipher
GlideEncryptionWrapperDB	Packages.com.glide.sys.security.EncryptionWrapperDB
GlideEncryptionWrapperDBAdmin	Packages.com.glide.sys.security.EncryptionWrapperDBAdmin
GlideEncryptionWrapperNAE	Packages.com.glide.sys.security.EncryptionWrapperNAE
GlideEncryptionWrapperNAEAdmin	Packages.com.glide.sys.security.EncryptionWrapperNAEAdmin
GlideEscalationManager	Packages.com.glide.escalation.EscalationManager
GlideEscalationTimerJobMarkII	Packages.com.glide.job.EscalationTimerJobMarkII
GlideEvaluator	Packages.com.glide.script.Evaluator
GlideEvent	Packages.com.glide.policy.Event
GlideEventManager	Packages.com.glide.policy.EventManager
GlideExcelExporter	Packages.com.glide.generators.ExcelExporter
GlideExcelLoader2	Packages.com.glide.db.impex.ExcelLoader2
GlideExecutionPlan	Packages.com.glide.execution_plan.ExecutionPlan
GlideExpressionWrapper	Packages.com.glide.ui.jelly.GlideExpressionWrapper
GlideExtensionPoint	Packages.com.glide.sys.ExtensionPoint
GlideFieldList	Packages.com.glide.processors.FieldList
GlideFile	Packages.com.glide.script.proxy.File
GlideFileUtil	Packages.com.glide.util.FileUtil
GlideFilter	Packages.com.glide.script.Filter
GlideFilterList	Packages.com.glide.script.FilterList

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideFixCatalogPlans	Packages.com.glide.fixes.FixCatalogPlans
GlideFixDeliveryPlans	Packages.com.glide.fixes.FixDeliveryPlans
GlideFixGroups	Packages.com.glide.fixes.FixGroups
GlideFixItemOptionsAgain	Packages.com.glide.fixes.FixItemOptionsAgain
GlideFixRules	Packages.com.glide.fixes.FixRules
GlideFixSpellCheck	Packages.com.glide.fixes.FixSpellCheck
GlideFixStuff	Packages.com.glide.fixes.FixStuff
GlideFixUsers	Packages.com.glide.fixes.FixUsers
GlideForm	Packages.com.glide.ui.GlideForm
GlideFormCommon	Packages.com.glide.ui.GlideFormCommon
GlideFormulator	Packages.com.glide.ui.GlideFormulator
GlideGauge	Packages.com.glide.report.Gauge
GlideGovernor	Packages.com.glide.sys.util.Governor
GlideGregorianCalendar	Packages.com.glide.util.GlideGregorianCalendar
GlideGroup	Packages.com.glide.sys.Group
GlideGroupByListTag	Packages.com.glide.ui.jelly.tags.form.GroupByListTag
GlideGuid	Packages.com.glide.util.Guid
GlideHierarchicalReference	Packages.com.glide.glideobject.HierarchicalReference
GlideHistorySet	Packages.com.glide.audit.HistorySet
GlideHistoryTag2	Packages.com.glide.ui.jelly.tags.mergedata.HistoryTag2
GlideHostUtil	Packages.com.glide.util.HostUtil
GlideHTTPClient	https://www.youtube.com/watch?feature=player_detailpage&v=
GlideHTTPRequest	Packages.com.glide.communications.HTTPRequest
GlideHTTPResponse	Packages.com.glide.communications.HTTPResponse
GlideI18NStyle	Packages.com.glide.ui.I18NStyle
GlideICALUtil	Packages.com.glide.policy.ICALUtil
GlideIConstants	Packages.com.glide.util.IConstants
GlideIGlideRecord	Packages.com.glide.util.IGlideRecord
GlideImageLoader	Packages.com.glide.script.ImageLoader
GlideImpersonate	Packages.com.glide.sys.Impersonate
GlideImportLog	Packages.com.glide.db.impex.ImportLog
GlideImportMap	Packages.com.glide.db.impex.ImportMap
GlideImportMapField	Packages.com.glide.db.impex.ImportMapField
GlideImportSet	Packages.com.glide.system_import_set.ImportSet

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideImportSetLoader	Packages.com.glide.system_import_set.ImportSetLoader
GlideImportSetRun	Packages.com.glide.system_import_set.ImportSetRun
GlideImportSetTransformer	Packages.com.glide.system_import_set.ImportSetTransformer
GlideImportSetTransformerWorker	Packages.com.glide.system_import_set.ImportSetTransformerW
GlideIndexDescriptor	Packages.com.glide.db.IndexDescriptor
GlideIndexUtils	Packages.com.glide.db.IndexUtils
GlideIntegerTime	Packages.com.glide.glideobject.IntegerTime
GlideInternalElementTypeChoiceList	Packages.com.glide.script.InternalElementTypeChoiceList
GlideInternalMonitor	Packages.com.glide.ui.monitor.InternalMonitor
GlideIOMonitor	Packages.com.glide.ui.monitor.IOMonitor
GlideIOStats	Packages.com.glide.db.IOStats
GlideIPAddressUtil	Packages.com.glide.util.IPAddressUtil
GlideIQueryCondition	Packages.com.glide.util.IQueryCondition
GlideIRow	Packages.com.glide.db.meta.IRow
GlideIRQuerySummary	Packages.com.glide.db.ir.IRQuerySummary
GlideIRQuerySummarySimple	Packages.com.glide.db.ir.IRQuerySummarySimple
GlideISecurityManager	Packages.com.glide.sys.security.ISecurityManager
GlideITableIterator	Packages.com.glide.db.access.ITableIterator
GlideJDBCLoader	Packages.com.glide.db.impex.JDBCLoader
GlideJDBCProbeTestWorker	Packages.com.glide.db.impex.JDBCProbeTestWorker
GlideJellyContext	Packages.com.glide.ui.jelly.GlideJellyContext
GlideJellyRunner	Packages.com.glide.ui.jelly.JellyRunner
GlideJID	Packages.com.glide.xmpp.JID
GlideJSTestUtil	Packages.com.glide.autotester.JSTestUtil
GlideJSUtil	Packages.com.glide.script.JSUtil
GlideLabelEventHandler	Packages.com.glide.labels.LabelEventHandler
GlideLabelGenerator	Packages.com.glide.db.LabelGenerator
GlideLabelUtil	Packages.com.glide.labels.LabelUtil
GlideLDAP	Packages.com.glide.sys.ldap.LDAP
GlideLDAPConfig	Packages.com.glide.sys.ldap.LDAPConfig
GlideLDAPConfigurations	Packages.com.glide.sys.ldap.LDAPConfigurations
GlideLDAPErrorAnalyzer	Packages.com.glide.sys.ldap.LDAPErrorAnalyzer
GlideLDAPGroups	Packages.com.glide.sys.ldap.LDAPGroups
GlideLDAPRefresh	Packages.com.glide.sys.ldap.LDAPRefresh

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideLDAPResult	Packages.com.glide.sys.ldap.LDAPResult
GlideLDAPTarget	Packages.com.glide.sys.ldap.LDAPTarget
GlideLDAPTransformQueue	Packages.com.glide.sys.ldap.LDAPTransformQueue
GlideLDAPUsers	Packages.com.glide.sys.ldap.LDAPUsers
GlideLDAPUserUpdate	Packages.com.glide.sys.ldap.LDAPUserUpdate
GlideList	Packages.com.glide.glideobject.GlideList
GlideListGroupProperties	Packages.com.glide.list_v2.ListGroupProperties
GlideListLabel	Packages.com.glide.ui.ListLabel
GlideListM2MBacking	Packages.com.glide.glideobject.GlideListM2MBacking
GlideListProperties	Packages.com.glide.list_v2.ListProperties
GlideListSearchQuery	Packages.com.glide.ui.ListSearchQuery
GlideLoader	Packages.com.glide.db.impex.Loader
GlideLoadTestDirector	Packages.com.glide.load_test.LoadTestDirector
GlideLocale	Packages.com.glide.sys.GlideLocale
GlideLocaleLoader	Packages.com.glide.currency.LocaleLoader
GlideLock	Packages.com.glide.script.Lock
GlideLog	Packages.com.glide.util.Log
GlideLogCleanup	Packages.com.glide.util.LogCleanup
GlideLogFileReader	Packages.com.glide.log_file.LogFileReader
GlideLRUCache	Packages.com.glide.sys.cache.LRUCache
GlideLuceneTextIndexEvent	Packages.com.glide.lucene.TextIndexEvent
GlideMarkupWriter	Packages.com.glide.util.MarkupWriter
GlideMemoryActive	Packages.com.glide.ui.monitor.MemoryActive
GlideMemoryCache	Packages.com.glide.ui.monitor.MemoryCache
GlideMemoryRecord	Packages.com.glide.script.GlideMemoryRecord
GlideMemorySwap	Packages.com.glide.ui.monitor.MemorySwap
GlideMemoryTable	Packages.com.glide.util.MemoryTable
GlideMemoryTotal	Packages.com.glide.ui.monitor.MemoryTotal
GlideMetaData	Packages.com.glide.script.MetaData
GlideMIDServerInfoAccessor	Packages.com.glide.script.MIDServerInfoAccessor
GlideMobileExtensions	Packages.com.glide.ui.MobileExtensions
GlideModule	Packages.com.glide.sys.Module
GlideMultipleAction	Packages.com.glide.db.MultipleAction
GlideMultipleDelete	Packages.com.glide.db.MultipleDelete

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideMultipleInsert	Packages.com.glide.db.MultipleInsert
GlideMultipleUpdate	Packages.com.glide.db.MultipleUpdate
GlideMutex	Packages.com.glide.sys.lock.Mutex
GlideMySQLWatch	Packages.com.glide.sys.stats.MySQLWatch
GlideNumber	Packages.com.glide.script.glide_elements.GlideNumber
GlideNumberManager	Packages.com.glide.db.NumberManager
GlideObjectManager	Packages.com.glide.glideobject.GlideObjectManager
GlideObjectUtil	Packages.com.glide.util.ObjectUtil
GlideOrderingDefinitionCreator	Packages.com.glide.sorting.OrderingDefinitionCreator
GlideOrderingManager	Packages.com.glide.sorting.OrderingManager
GlideOutputWriter	Packages.com.glide.ui.io.GlideOutputWriter
GlideOutputWriterFactory	Packages.com.glide.ui.io.GlideOutputWriterFactory
GlideOverLoadedChoices	Packages.com.glide.script.OverLoadedChoices
GlidePartitionMonitor	Packages.com.glide.ui.monitor.PartitionMonitor
GlidePivotTableSummaryTableWriter	Packages.com.glide.ui.chart.dataset.PivotTableSummaryTableWr
GlidePlugin	Packages.com.glide.sys.Plugin
GlidePluginManager	Packages.com.glide.sys.PluginManager
GlidePluginManagerWorker	Packages.com.glide.sys.PluginManagerWorker
GlidePluginUtils	Packages.com.glide.sys.PluginUtils
GlidePOP3Reader	Packages.com.glide.notification.inbound.POP3Reader
GlidePOP3ReaderJob	Packages.com.glide.job.POP3ReaderJob
GlidePopup	Packages.com.glide.ui.Popup
GlidePriceGenerator	Packages.com.glide.currency.PriceGenerator
GlidePriceLoader	Packages.com.glide.currency.PriceLoader
GlideProcessor	Packages.com.glide.processors.Processor
GlideProcessRunner	Packages.com.glide.util.ProcessRunner
GlideProgressMonitor	Packages.com.glide.worker.ProgressMonitor
GlideProgressWorker	Packages.com.glide.worker.ProgressWorker
GlideProperties	Packages.com.glide.util.GlideProperties
GlidePropertiesDB	Packages.com.glide.util.GlidePropertiesDB
GlideProperty	Packages.com.glide.util.GlideProperty
GlidePublicPage	Packages.com.glide.ui.PublicPage
GlideQueryBreadcrumbs	Packages.com.glide.misc.QueryBreadcrumbs
GlideQueryCondition	Packages.com.glide.db.conditions.QueryCondition

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideQueryFormatter	Packages.com.glide.ui.jelly.tags.form.QueryFormatter
GlideQueryString	Packages.com.glide.db.QueryString
GlideQueryTerm	Packages.com.glide.db.QueryTerm
GlideRecord	Packages.com.glide.script.GlideRecord
GlideRecordCache	Packages.com.glide.sys.RecordCache
GlideRecordEnsurer	Packages.com.glide.misc.RecordEnsurer
GlideRecordFactory	Packages.com.glide.script.GlideRecordFactory
GlideRecordKeySetLoader	Packages.com.glide.script.GlideRecordKeySetLoader
GlideRecordLock	Packages.com.glide.script.RecordLock
GlideRecordPopupGenerator	Packages.com.glide.calendar.RecordPopupGenerator
GlideRecordRollback	Packages.com.glide.script.GlideRecordRollback
GlideRecordSet	Packages.com.glide.script.GlideRecordSet
GlideRecordSimpleSerializer	Packages.com.glide.script.GlideRecordSimpleSerializer
GlideRecordXMLSerializer	Packages.com.glide.script.GlideRecordXMLSerializer
GlideReferenceField	Packages.com.glide.script.ReferenceField
GlideRegexUtil	Packages.com.glide.util.RegexUtil
GlideRegisterEscalationEvents	Packages.com.glide.fixes.RegisterEscalationEvents
GlideRelatedListReconciler	Packages.com.glide.misc.RelatedListReconciler
GlideRelationship	Packages.com.glide.sys.Relationship
GlideRelationships	Packages.com.glide.db.Relationships
GlideRelationshipUtil	Packages.com.glide.sys.RelationshipUtil
GlideRemoteGlideRecord	Packages.com.glide.communications.RemoteGlideRecord
GlideRenderProperties	Packages.com.glide.ui.RenderProperties
GlideReplaceUpdateFiles	Packages.com.glide.util.ReplaceUpdateFiles
GlideReplicationEngine	Packages.com.glide.replicator.ReplicationEngine
GlideReport	Packages.com.glide.report.Report
GlideReportChoiceList	Packages.com.glide.script.ReportChoiceList
GlideReportViewManagement	Packages.com.glide.report.ReportViewManagement
GlideRequestMap	Packages.com.glide.util.RequestMap
GlideRevertToOutOfBox	Packages.com.glide.update.RevertToOutOfBox
GlideRhinoEnvironment	Packages.com.glide.script.GlideRhinoEnvironment
GlideRhinoHelper	Packages.com.glide.script.GlideRhinoHelper
GlideRhinoScope	Packages.com.glide.script.RhinoScope
GlideRhinoScopeHandler	Packages.com.glide.script.GlideRhinoScopeHandler

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideRhinoTestCase	Packages.com.glide.autotester.RhinoTestCase
GlideRRDBAlertProcessor	Packages.com.glide.rrdb.alerts.RRDBAlertProcessor
GlideRRDBDefinition	Packages.com.glide.rrdb.RRDBDefinition
GlideRunScriptJob	Packages.com.glide.job.RunScriptJob
GlideSchedule	• Packages.com.glide.schedules.Schedule • Packages.java.util.TimeZone
GlideScheduleDateTime	Packages.com.glide.glideobject.ScheduleDateTime
GlideScheduleDateTimeSpan	Packages.com.glide.schedules.ScheduleDateTimeSpan
GlideScheduleItem	Packages.com.glide.schedules.ScheduleItem
GlideScheduler	Packages.com.glide.schedule.GlideScheduler
GlideScheduleTimeMap	Packages.com.glide.schedules.ScheduleTimeMap
GlideScheduleTimeSpan	Packages.com.glide.schedules.ScheduleTimeSpan
GlideScriptChoiceList	Packages.com.glide.script.ChoiceList
GlideScriptedProgressWorker	Packages.com.glide.worker.ScriptedProgressWorker
GlideScriptEvaluator	Packages.com.glide.script.ScriptEvaluator
GlideScriptGlobals	Packages.com.glide.script.GlideScriptGlobals
GlideScriptListener	Packages.com.glide.listener.ScriptListener
GlideScriptProcessor	Packages.com.glide.processors.ScriptProcessor
GlideScriptRecordUtil	Packages.com.glide.script.GlideRecordUtil
GlideScriptSystemUtilDB	Packages.com.glide.script.GlideSystemUtilDB
GlideScriptViewManager	Packages.com.glide.ui.ViewManager
GlideScriptWriter	Packages.com.glide.script.ScriptWriter
GlideSearchQueryFormatter	Packages.com.glide.text_search.SearchQueryFormatter
GlideSecondaryDatabaseBehindnessChecker	Packages.com.glide.secondary_db_pools.SecondaryDatabaseB
GlideSecondaryDatabaseConfiguration	Packages.com.glide.secondary_db_pools.SecondaryDatabaseC
GlideSecurityManager	Packages.com.glide.sys.security.GlideSecurityManager
GlideSecurityQueryCalculator	Packages.com.glide.sys.security.SecurityQueryCalculator
GlideSecurityUtils	Packages.com.glide.util.SecurityUtils
GlideSelfCleaningMutex	Packages.com.glide.sys.lock.SelfCleaningMutex
GlideServiceAPIWrapper	Packages.com.glide.service_api.ServiceAPIWrapper
GlideServlet	Packages.com.glide.ui.GlideServlet
GlideServletRequest	Packages.com.glide.ui.GlideServletRequest
GlideServletResponse	Packages.com.glide.ui.GlideServletResponse

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideServletStatus	Packages.com.glide.ui.ServletStatus
GlideSession	Packages.com.glide.sys.GlideSession
GlideSessionDebug	Packages.com.glide.sys.SessionDebug
GlideSessions	Packages.com.glide.ui.Sessions
GlideSessionSandbox	Packages.com.glide.script.GlideSessionSandbox
GlideShellCommand	Packages.com.glide.util.ShellCommand
GlideSimmerDown	Packages.com.glide.db.change.command.SimmerDown
GlideSimmerUp	Packages.com.glide.db.change.command.SimmerUp
GlideSimpleDateFormatEx	• Packages.com.glide.util.SimpleDateFormatEx • Packages.com.glide.util.SimpleDateFormat **Note:** The Packages.com.glide.util.SimpleDateFormatEx package e Packages.com.glide.util.SimpleDateFormat package.
GlideSimpleHTTPClient	Packages.com.glide.communications.SimpleHTTPClient
GlideSimpleScriptListener	Packages.com.glide.listener.SimpleScriptListener
GlideSMTPConnection	Packages.com.glide.notification.outbound.SMTPConnection
GlideSMTPSender	Packages.com.glide.notification.outbound.SMTPSender
GlideSMTPSenderJob	Packages.com.glide.job.SMTPSenderJob
GlideSOAPDocument	Packages.com.glide.communications.soap.SOAPDocument
GlideSOAPRequest	Packages.com.glide.communications.soap.SOAPRequest
GlideSOAPResponse	Packages.com.glide.communications.soap.SOAPResponse
GlideSOAPSecurity	Packages.com.glide.processors.soap.SOAPSecurity
GlideSOAPSigner	Packages.com.glide.communications.soap.SOAPSigner
GlideSocket	Packages.com.glide.script.proxy.Socket
GlidesoftGlideAttributesImpl	Packages.com.glidesoft.util.xml.GlideAttributesImpl
GlidesoftXMLMemoryTable	Packages.com.glidesoft.util.xml.XMLMemoryTable
GlideSQLChildMonitor	Packages.com.glide.monitor.sql.SQLChildMonitor
GlideSQLDebug	Packages.com.glide.ui.diagnostics.SQLDebug
GlideSQLDeleteMonitor	Packages.com.glide.monitor.sql.SQLDeleteMonitor
GlideSQLInsertMonitor	Packages.com.glide.monitor.sql.SQLInsertMonitor
GlideSQLResponseMonitor	Packages.com.glide.monitor.sql.SQLResponseMonitor
GlideSQLSelectMonitor	Packages.com.glide.monitor.sql.SQLSelectMonitor
GlideSQLUpdateMonitor	Packages.com.glide.monitor.sql.SQLUpdateMonitor
GlideSSHClient	Packages.com.glide.communications.SSHClient

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideStack	Packages.com.glide.sys.GlideStack
GlideStatistician	Packages.com.glide.sys.stats.Statistician
GlideStatsInfo	Packages.com.glide.monitor.StatsInfo
GlideStatus	Packages.com.glide.util.GlideStatus
GlideStopWatch	Packages.com.glide.util.StopWatch
GlideStorageUtils	Packages.com.glide.db.meta.StorageUtils
GlideStringCache	Packages.com.glide.sys.cache.StringCache
GlideStringInputStream	Packages.com.glide.util.StringInputStream
GlideStringList	Packages.com.glide.collections.StringList
GlideStringUtil	Packages.com.glide.util.StringUtil
GlideSubQuery	Packages.com.glide.db.conditions.SubQuery
GlideSubstituteURL	Packages.com.glide.notification.substitution.SubstituteURL
GlideSummaryTableGroupReader	Packages.com.glide.ui.chart.dataset.SummaryTableGroupReade
GlideSummaryTableOrderedReader	Packages.com.glide.ui.chart.dataset.SummaryTableOrderedRea
GlideSummaryTableReader	Packages.com.glide.ui.chart.dataset.SummaryTableReader
GlideSummaryTableWriter	Packages.com.glide.ui.chart.dataset.SummaryTableWriter
GlideSynchronizedLRUCache	Packages.com.glide.sys.cache.SynchronizedLRUCache
GlideSysAttachment	Packages.com.glide.ui.SysAttachment
GlideSysAttachmentInputStream	Packages.com.glide.ui.SysAttachmentInputStream
GlideSysBRThreadMonitor	Packages.com.glide.monitor.threads.SysBRThreadMonitor
GlideSysChoice	Packages.com.glide.script.SysChoice
GlideSysConcurrencyMonitor	Packages.com.glide.monitor.threads.SysConcurrencyMonitor
GlideSysCPUThreadMonitor	Packages.com.glide.monitor.threads.SysCPUThreadMonitor
GlideSysDateUtil	Packages.com.glide.sys.util.SysDateUtil
GlideSysDBThreadMonitor	Packages.com.glide.monitor.threads.SysDBThreadMonitor
GlideSysField	Packages.com.glide.db.SysField
GlideSysFileUtil	Packages.com.glide.sys.util.SysFileUtil
GlideSysForm	Packages.com.glide.ui.SysForm
GlideSysForms	Packages.com.glide.ui.SysForms
GlideSysList	Packages.com.glide.ui.SysList
GlideSysListControl	Packages.com.glide.ui.SysListControl
GlideSysLog	Packages.com.glide.sys.SysLog
GlideSYSMany2Many	Packages.com.glide.db.SYSMany2Many
GlideSysMessage	Packages.com.glide.ui.SysMessage

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideSysNetThreadMonitor	Packages.com.glide.monitor.threads.SysNetThreadMonitor
GlideSysRelatedList	Packages.com.glide.ui.SysRelatedList
GlideSysSection	Packages.com.glide.ui.SysSection
GlideSysSemaphore	Packages.com.glide.sys.util.SysSemaphore
GlideSystem	Packages.com.glide.script.GlideSystem
GlideSystemDateUtil	Packages.com.glide.script.system.GlideSystemDateUtil
GlideSystemUtil	Packages.com.glide.util.SystemUtil
GlideSystemUtilDB	Packages.com.glide.script.system.GlideSystemUtilDB
GlideSystemUtilScript	Packages.com.glide.script.system.GlideSystemUtilScript
GlideSysThreadMonitor	Packages.com.glide.monitor.threads.SysThreadMonitor
GlideSysUserList	Packages.com.glide.ui.SysUserList
GlideTable	Packages.com.glide.db.meta.Table
GlideTableChoiceList	Packages.com.glide.script.TableChoiceList
GlideTableCleaner	Packages.com.glide.misc.TableCleaner
GlideTableCleanerJob	Packages.com.glide.job.TableCleanerJob
GlideTableCreator	Packages.com.glide.db.impex.TableCreator
GlideTableDescriptor	Packages.com.glide.db.TableDescriptor
GlideTableGroupMover	Packages.com.glide.db.auxiliary.TableGroupMover
GlideTableManager	Packages.com.glide.db.TableManager
GlideTableMover	Packages.com.glide.db.auxiliary.TableMover
GlideTableParentChange	Packages.com.glide.db.table.TableParentChange
GlideTableParentColumnChange	Packages.com.glide.db.table.TableParentColumnChange
GlideTaskToken	Packages.com.glide.execution_plan.TaskToken
GlideTemplate	Packages.com.glide.script.Template
GlideTestAgent	Packages.com.glide.autotester.GlideTestAgent
GlideTextIndexEvent	Packages.com.glide.ts.event.TextIndexEvent
GlideThreadAttributes	Packages.com.glide.ui.GlideThreadAttributes
GlideThreadUtil	Packages.com.glide.util.ThreadUtil
GlideTime	Packages.com.glide.glideobject.GlideTime
GlideTimelineFrameSeparator	Packages.com.glide.schedules.TimelineFrameSeparator
GlideTimelineItem	Packages.com.glide.schedules.TimelineItem
GlideTimelineSpan	Packages.com.glide.schedules.TimelineSpan
GlideTomcatHelper	Packages.com.glide.startup.TomcatHelper
GlideTransaction	Packages.com.glide.sys.Transaction

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideTransactionManager	Packages.com.glide.sys.TransactionManager
GlideTransferAuditDataHelper	Packages.com.glide.audit.TransferAuditDataHelper
GlideTransformer	Packages.com.glide.db.impex.transformer.Transformer
GlideTreePicker	Packages.com.glide.ui.TreePicker
GlideTSAnalysisViewer	Packages.com.glide.ts.cluster.TSAnalysisViewer
GlideTSAnalyticsProcessor	Packages.com.glide.ts.cluster.TSAnalyticsProcessor
GlideTSChainsHandler	Packages.com.glide.ts.trends.TSChainsHandler
GlideTSChainsLoader	Packages.com.glide.ts.trends.TSChainsLoader
GlideTSChainsPusher	Packages.com.glide.ts.trends.TSChainsPusher
GlideTSChainsSummarizer	Packages.com.glide.ts.trends.TSChainsSummarizer
GlideTSClusterDefinitions	Packages.com.glide.ts.cluster.TSClusterDefinitions
GlideTSDebug	Packages.com.glide.ts.util.TSDebug
GlideTSDidYouMean	Packages.com.glide.ts.util.TSDidYouMean
GlideTSGlobalKeywordSummarizer	Packages.com.glide.ts.trends.TSGlobalKeywordSummarizer
GlideTSIndexStatistician	Packages.com.glide.ts.stats.TSIndexStatistician
GlideTSIndexStopGenerator	Packages.com.glide.ts.stats.TSIndexStopGenerator
GlideTSIndexTables	Packages.com.glide.ts.indexer.TSIndexTables
GlideTSKeywordHandler	Packages.com.glide.ts.trends.TSKeywordHandler
GlideTSKeywordLoader	Packages.com.glide.ts.trends.TSKeywordLoader
GlideTSKeywordPusher	Packages.com.glide.ts.trends.TSKeywordPusher
GlideTSMoversViewer	Packages.com.glide.ts.cluster.TSMoversViewer
GlideTSSearchStatistician	Packages.com.glide.ts.stats.TSSearchStatistician
GlideTSSearchSummary	Packages.com.glide.ts.trends.TSSearchSummary
GlideTSTopSearches	Packages.com.glide.ts.util.TSTopSearches
GlideTSUtil	Packages.com.glide.ts.util.TSUtil
GlideTSVersion	Packages.com.glide.ts.TSVersion
GlideUIAction	Packages.com.glide.ui.action.UIAction
GlideUISession	Packages.com.glide.ui.Session
GlideUnloader	Packages.com.glide.db.impex.Unloader
GlideUpdateManager2	Packages.com.glide.update.UpdateManager2
GlideUpdateSet	Packages.com.glide.update.UpdateSet
GlideUpdateSetController	Packages.com.glide.system_update_set.UpdateSetController
GlideUpdateSetPreviewer	Packages.com.glide.update.UpdateSetPreviewer
GlideUpdateSetWorker	Packages.com.glide.update.UpdateSetWorker

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
GlideUpdateSyncher	Packages.com.glide.policy.UpdateSyncher
GlideUpdateTableChoiceList	Packages.com.glide.script.UpdateTableChoiceList
GlideUpgrade	Packages.com.glide.sys.Upgrade
GlideUpgradeArtifactManager	Packages.com.glide.misc.UpgradeArtifactManager
GlideUpgradeLog	Packages.com.glide.update.UpgradeLog
GlideUpgradeMonitor	Packages.com.glide.update.UpgradeMonitor
GlideURI	Packages.com.glide.ui.GlideURI
GlideURL	Packages.com.glide.util.GlideURL
GlideURLUTF8Encoder	Packages.com.glide.util.URLUTF8Encoder
GlideURLUtil	Packages.com.glide.util.URLUtil
GlideUser	Packages.com.glide.sys.User
GlideUserAuthenticator	Packages.com.glide.sys.UserAuthenticator
GlideUserGroup	Packages.com.glide.sys.UserGroup
GlideUtil	Packages.com.glide.util.GlideUtil
GlideViewRuleNavigator	Packages.com.glide.ui.ViewRuleNavigator
GlideWarDeleter	Packages.com.glide.misc.WarDeleter
GlideWarDownloader	Packages.com.glide.misc.WarDownloader
GlideWhiteListManager	Packages.com.glide.script.api.GlideWhiteListManager
GlideWikiModel	Packages.com.glide.wiki.GlideWikiModel
GlideWorkerThread	Packages.com.glide.worker.WorkerThread
GlideWorkerThreadManager	Packages.com.glide.sys.WorkerThreadManager
GlideWSClient	Packages.com.glide.util.WSClient
GlideWSDefinition	Packages.com.glide.wsdlreader.WSDefinition
GlideWSDLReader	Packages.com.glide.wsdlreader.GlideWSDLReader
GlideXMLChoiceListSerializer	Packages.com.glide.choice.XMLChoiceListSerializer
GlideXMLDocument	Packages.com.glide.util.XMLDocument
GlideXMLElementIterator	Packages.com.glide.util.XMLElementIterator
GlideXMLGlideRecordSerializer	Packages.com.glide.processors.xmlhttp.XMLGlideRecordSeriali
GlideXMLParameters	Packages.com.glide.util.XMLParameters
GlideXMLStats	Packages.com.glide.ui.XMLStats
GlideXMLSysMetaSerializer	Packages.com.glide.processors.xmlhttp.XMLSysMetaSerializer
GlideXMLUtil	Packages.com.glide.util.XMLUtil
GlideZipUtil	Packages.com.glide.util.ZipUtil
RhinoEnvironment	Packages.com.glide.script.RhinoEnvironment

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
RhinoHelper	Packages.com.glide.script.RhinoHelper
SecurelyAccess	Packages.com.glide.sys.util.SecurelyAccess
ServiceAPI	Packages.com.glide.service_api.ServiceAPI
SncAddress32Bit	Packages.com.snc.commons.networks.Address32Bit
SncAliasApplier	Packages.com.snc.field_normalization.AliasApplier
SncAppFiles	Packages.com.snc.apps.api.AppFiles
SncApplicationFileListener	Packages.com.snc.apps.db.ApplicationFileListener
SncAppsAccess	Packages.com.snc.apps.api.AppsAccess
SncAppsUI	Packages.com.snc.apps.api.AppsUI
SncASensor	Packages.com.snc.discovery.sensor.ASensor
SncAuthentication	Packages.com.snc.authentication.digest.Authentication
SncBaselineCMDB	Packages.com.snc.cmdb.BaselineCMDB
SncBulkCopy	Packages.com.snc.ha.clone.BulkCopy
SncClassifiedProcess	Packages.com.snc.discovery.sensor.ClassifiedProcess
SncClassify	Packages.com.snc.discovery.sensor.snmp.Classify
SncCloneController	Packages.com.snc.ha.clone.CloneController
SncCloneInstance	Packages.com.snc.ha.clone.instance.Instance
SncCloneLogger	Packages.com.snc.ha.clone.CloneLogger
SncCloneTask	Packages.com.snc.ha.clone.CloneTask
SncCloneUtils	Packages.com.snc.ha.clone.CloneUtils
SncConfiguration	Packages.com.snc.field_normalization.db.Configuration
SncConfigurations	Packages.com.snc.field_normalization.Configurations
SncConnectionTest	Packages.com.snc.ha.connectivity.ConnectionTest
SncDBChangeManagerFactoryHA	Packages.com.snc.db.clone.change.DBChangeManagerFactory
SncDeviceHistory	Packages.com.snc.discovery.logging.DeviceHistory
SncDiscoveryCancel	Packages.com.snc.discovery.DiscoveryCancel
SncDiscoveryClassification	Packages.com.snc.discovery.DiscoveryClassification
SncDiscoveryLog	Packages.com.snc.discovery.logging.DiscoveryLog
SncDiscoveryRanges	Packages.com.snc.commons.networks.DiscoveryRanges
SncDiscoveryRangesDB	Packages.com.snc.discovery.DiscoveryRangesDB
SncDiscoveryReconciler	Packages.com.snc.discovery.database.DiscoveryReconciler
SncDiscoverySNMPClassification	Packages.com.snc.discovery.sensor.snmp.DiscoverySNMPClass
SncDiscoveryUtils	Packages.com.snc.discovery.DiscoveryUtils
SncDropBackupTablesTask	Packages.com.snc.ha.clone.instance.DropBackupTablesTask

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
SncDropTablesTask	Packages.com.snc.ha.clone.DropTablesTask
SncEC2Properties	Packages.com.snc.ec2.EC2Properties
SncECMDBUtil	Packages.com.snc.cmdb.ECMDBUtil
SncElrondClient	Packages.com.snc.customer_logon.ElrondClient
SncExpert	Packages.com.snc.expert.Expert
SncExpertInstance	Packages.com.snc.expert.ExpertInstance
SncExpertPanel	Packages.com.snc.expert.ExpertPanel
SncExtantDataJob	Packages.com.snc.field_normalization.db.ExtantDataJob
SncExtantDataJobState	Packages.com.snc.field_normalization.ExtantDataJobState
SncFailoverController	Packages.com.snc.da.gateway.replication.FailoverController
SncFileTree	Packages.com.snc.apps.file.FileTree
SncGatewayCache	Packages.com.snc.da.gateway.GatewayCache
SncGatewayClone	Packages.com.snc.da.gateway.clone.GatewayClone
SncGatewayConnectivity	Packages.com.snc.da.gateway.GatewayConnectivity
SncGatewayPluginStartup	Packages.com.snc.da.gateway.replication.GatewayPluginStartup
SncGatewayTruncateHierarchy	Packages.com.snc.da.gateway.clone.GatewayTruncateHierarchy
SncGlideGateways	Packages.com.snc.da.gateway.GlideGateways
SncHAClone	Packages.com.snc.ha.clone.HAClone
SncHAConnectionTest	Packages.com.snc.ha.connectivity.HAConnectionTest
SncHADatabaseCheck	Packages.com.snc.ha.tablecheck.HADatabaseCheck
SncHAPairingUtils	Packages.com.snc.ha.HAPairingUtils
SncHAReplicationController	Packages.com.snc.ha.HAReplicationController
SncHAReplicationQueueSnapshotBuilder	Packages.com.snc.ha.tablecheck.HAReplicationQueueSnapsho
SncHATableCheck	Packages.com.snc.ha.tablecheck.HATableCheck
SncHATableCheckThread	Packages.com.snc.ha.tablecheck.HATableCheckThread
SncHATableQuickCheck	Packages.com.snc.ha.tablecheck.HATableQuickCheck
SncHATableRepair	Packages.com.snc.ha.tablecheck.HATableRepair
SncHAUtils	Packages.com.snc.ha.HAUtils
SncHostname	Packages.com.snc.discovery.utils.Hostname
SncInstanceClone	Packages.com.snc.ha.clone.instance.InstanceClone
SncInstanceConnectionTest	Packages.com.snc.ha.connectivity.InstanceConnectionTest
SncInstanceRollback	Packages.com.snc.ha.clone.instance.InstanceRollback
SncIPAddressV4	Packages.com.snc.commons.networks.IPAddressV4
SncIPAddressV6	Packages.com.snc.commons.networks.IPAddressV6

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
SncIPAuthenticator	Packages.com.snc.ipauthenticator.IPAuthenticator
SncIPIterator	Packages.com.snc.commons.networks.IPIterator
SncIPList	Packages.com.snc.commons.networks.IPList
SncIPMetaCollection	Packages.com.snc.commons.networks.IPMetaCollection
SncIPNetmaskV4	Packages.com.snc.commons.networks.IPNetmaskV4
SncIPNetworkV4	Packages.com.snc.commons.networks.IPNetworkV4
SncIPRangeV4	Packages.com.snc.commons.networks.IPRangeV4
SncJRobinGraphDef	Packages.com.snc.jrobin.JRobinGraphDef
SncLayer7Connections	Packages.com.snc.discovery.Layer7Connections
SncMACAddress	Packages.com.snc.commons.networks.MACAddress
SncMACAddressMfr	Packages.com.snc.commons.networks.MACAddressMfr
SncMakeAndModel	Packages.com.snc.cmdb.MakeAndModel
SncMIDConfigParameter	Packages.com.snc.commons.MIDConfigParameter
SncMIDServerRangesDB	Packages.com.snc.discovery.MIDServerRangesDB
SncNormalCoalescer	Packages.com.snc.field_normalization.NormalCoalescer
SncNormalizer	Packages.com.snc.field_normalization.Normalizer
SncNormalValueChanger	Packages.com.snc.field_normalization.NormalValueChanger
SncNotifySNC	Packages.com.snc.system.NotifySNC
SncOnCallRotation	Packages.com.snc.on_call_rotation.OnCallRotation
SncPendingValueCollector	Packages.com.snc.field_normalization.PendingValueCollector
SncPreferences	Packages.com.snc.field_normalization.Preferences
SncPrintServerHelper	Packages.com.snc.discovery.database.PrintServerHelper
SncProbe	Packages.com.snc.discovery.Probe
SncProbeRunTime	Packages.com.snc.discovery.perfmon.ProbeRunTime
SncRBSensorProcessor	Packages.com.snc.discovery_automation.RBSensorProcessor
SncReadTest	Packages.com.snc.ha.ReadTest
SncReclassifyCI	Packages.com.snc.cmdb.ReclassifyCI
SncRelationships	Packages.com.snc.cmdb.Relationships
SncReplicationAdvisor	Packages.com.snc.db.replicate.ReplicationAdvisor
SncReplicationEngine	Packages.com.snc.db.replicate.ReplicationEngine
SncReplicationQueue	Packages.com.snc.db.replicate.ReplicationQueue
SncRequestCredentials	Packages.com.snc.customer_logon.RequestCredentials
SncRrdGlideBackendFactory	Packages.com.snc.jrobin.RrdGlideBackendFactory
SncRuleApplier	Packages.com.snc.field_normalization.RuleApplier

GlideScriptable object replacement list (continued)

GlideScriptable Class	Packages Call
SncRuleToPending	Packages.com.snc.field_normalization.RuleToPending
SncSAMCounter	Packages.com.snc.software_asset_management.SAMCounter
SncScheduleDropBackupTablesTask	Packages.com.snc.ha.clone.instance.ScheduleDropBackupTabl
SncScrapeIANAEnterpriseNumbers	Packages.com.snc.discovery.database.ScrapeIANAEnterpriseNu
SncScrapeIEEENICCodes	Packages.com.snc.discovery.database.ScrapeIEEENICCodes
SncSendNotificationTask	Packages.com.snc.ha.clone.instance.SendNotificationTask
SncSensorProcessor	Packages.com.snc.discovery.SensorProcessor
SncSerialNumber	Packages.com.snc.discovery.SerialNumber
SncSerialNumberList	Packages.com.snc.discovery.SerialNumberList
SncSessionMate	Packages.com.snc.discovery.SessionMate
SncSimmerControl	Packages.com.snc.ha.clone.instance.SimmerControl
SncTableEditor	Packages.com.snc.apps.api.TableEditor
SncTableRotation	Packages.com.snc.db.replicate.TableRotation
SncTableRotationExtension	Packages.com.snc.db.replicate.TableRotationExtension
SncTableRotationExtensions	Packages.com.snc.db.replicate.TableRotationExtensions
SncTableRotationWatcher	Packages.com.snc.db.replicate.TableRotationWatcher
SncTransformApplier	Packages.com.snc.field_normalization.TransformApplier
SncTreeBuilder	Packages.com.snc.apps.tree.TreeBuilder
SncTriggerSynchronizer	Packages.com.snc.automation.TriggerSynchronizer
SncValue	Packages.com.snc.field_normalization.db.Value
TestExtension	Packages.com.glide.junit.misc.TestExtension
UINotification	Packages.com.glide.ui.UINotification

Packages Call Removal Tool error types

Possible error types generated by the Packages Call Removal Tool.

Error types

Error message	Description	Solution
Class Name Could Not Be Parsed	A line of script has what appears to be a Packages call (for example, the line contains `Packages.com.glide`) but on examination there is insufficient text to parse out a class name that corresponds to a scriptable object name.	This error type generally does not pr nevertheless flagged as an error in c a valid class that the parser, for what parse.
Class Does Not Exist	A line of script has an identifiable Packages call to a Java class that does not exist. This may be because it was incorrectly typed, or because the Java class no longer exists.	This error type usually identifies a sc what its author intended, if it ever did should be revisited, and the invalid P

Error types (continued)

Error message	Description	Solution
	Either way, the line of script is currently not doing anything and is potentially generating errors whenever it is run.	
Class Is Not a Scriptable Class	A line of script has an identifiable Packages call to an existing ServiceNow Java class that is not currently marked as scriptable. The call cannot be replaced, because there is currently no corresponding scriptable name.	To convert this script, either rewrite t not use the Java class or wait until Se a scriptable name for the class.
Class Does Not Have a Scriptable Name	A line of script has an identifiable Packages call to an existing ServiceNow Java class that does not currently have a scriptable name.	This error is less likely to occur than a Scriptable Class error; however, as w this script, either rewrite the script so Java class or wait until ServiceNow, l name for the class.
Variable Name Being Assigned Conflicts With a Scriptable Name	A variable name has the same name as a scriptable name. For example, `var GlideSession = Packages.com.glide.sys.GlideSession.get();` would generate this error because the variable name `GlideSession` is the same as the scriptable name for the GlideSession Java class.	Before this script can be converted, t must be changed wherever it occurs example, the line could be changed t `Packages.com.glide.sys.Gl` to remove the conflict. Also, be sure t name elsewhere in the script wherev
Internal Error - Unable To Find String To Be Replaced	The tool is unable to find the script text it is currently evaluating for Packages call replacement.	This error type is very unlikely. If it do code and correct any errors. If the er open an incident with Technical Sup

Service Creator

Service creator enables a department to offer custom services through the service catalog, such as the HR department offering tuition reimbursement for further education.

Each published service has an associated record producer catalog item. Users designated as managers and editors create and design these catalog items. End users can request services by ordering the catalog item.

All services belong to a published service category, which has an associated application and modules. When a user orders the catalog item for a service, the ServiceNow system creates a new task record within the application for that service category. Users designated as service fulfillers for the department complete these tasks to fulfill the service request.

Service creator process

The service creator process involves requesting and publishing a service category, designating editors and service fulfillers, creating and publishing services, and submitting and fulfilling service requests.

Request and publish a service category

A user, typically the department manager, can request a service category for the department. This user provides high-level information regarding the service category, such as the name, the department, and the manager for the service category.

A catalog administrator can approve the request which publishes the service category, creates a ServiceNow application for managing service requests associated with the category, and creates system components for the application.

Designate editors and service fulfillers

After a service category is published, the associated manager designates editors and service fulfillers. Editors can create and modify services within that service category. Service fulfillers can complete tasks that are generated by service requests.

The manager, editors, and service fulfillers must be members of the department the service category belongs to.

Create and publish services

The manager and editors create services within a service category. The service design interface provides a work area for creating and modifying services.

When the service is complete, the manager publishes the service to the service catalog.

Submit and fulfill service requests

End users can request published services by submitting a service catalog request. This request creates a new task record within the service category application. Service fulfillers then complete the task to fulfill the service request.

Related topics

Service Creator

Activate Service Creator

If the Service Creator plugin is not already activated, an administrator can activate it to access the application.

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. Find the plugin using the filter criteria and search bar.

   You can search for the plugin by its name or ID. If you cannot find a plugin, you might have to request it from ServiceNow personnel.
3. Select **Install** to start the installation process.

   Note:
   When domain separation and delegated admin are enabled in an instance, the administrative user must be in the **global** domain. Otherwise, the following error appears: `Application installation is unavailable because another operation is running: Plugin Activation for <plugin name>.`

   You will see a message after installation is completed. For information about the components installed with a plugin, see Find components installed with an application.

Installed with Service Creator

Several types of components are installed with Service Creator.

Demo data is available with Service Creator. The demo data provides the Departmental Services service catalog category.

Creating a new service category also creates components for that service category.

The following components are added with Service Creator:

Tables

Service Creator tables

Table	Description
Service Category [catalog_category_request]	Stores all service categories.
Service Category Request User [catalog_category_request_user]	Tracks fulfillers for a service category. Use these records to grant or remove roles as needed.
Service [sc_cat_item_producer_service]	Stores all services.
Service Category App Menu [service_category_app_menu]	Stores the application menus for each service category.
Service Category User Role [service_category_user_role]	Tracks users who have been granted a role due to being an editor of a service category.

UI actions

UI action	Description
Create Category and Table	Approves a requested service category and creates system components for that category.
Request Category Publication	Lets a service creator request their category be published.
Create New Service	Creates a new service within the service category.
View Table Definition	Opens the task table definition [sys_db_object] for a service category.
View Task List	Opens the list of tasks associated with the service category.

UI policies

UI policy	Description
Hide Due Date	Hides the Due date field on the Service Category form if State is Requested or Due date is empty.
Hide Category If Empty	Hides the Category field, if empty, on the Service Category form.
Show Published	Shows the Published check box on the Service Category form if State is Created but Unpublished or Ready for Publication.

UI policy	Description
Hide Table name	Shows Table and hides Table name on the Service Category form if Table has a value.
Hide Category Name	Hides Name on the Service Category form if State is Requested or Rejected.
Table name read only	Makes Department and Table name read only on the Service Category form if State is not Requested.
Hide Editors	Hides the Editors field on the Service Category form if State is Requested or Rejected.

Properties

Property	Description
glide.citizen_developer.category.auto_publish	Automatically adds new service categories to the service catalog as subcategories of the Departmental Services category. • Type: true|false • Default value: true • Location: System Properties [sys_properties] table
glide.citizen_developer.set_category_roles	Comma-separated list of roles that can set the category for a new service. • Type: String • Default value: admin, catalog_admin • Location: System Properties [sys_properties] table
glide.service_creator.auto_add_to_category	Automatically adds new services to the Departmental Services service catalog category, in addition to the department-specific category. • Type: true|false • Default value: true • Location: System Properties [sys_properties] table

Script includes

Script include	Description
serviceCategoryIsUnpublished	Global function that returns true if the service category is unpublished.

Script include	Description
getMyCatalogCategories	Global function that returns a list of categories for which the current user is the manager or an editor.

Client scripts

Client script	Description
Duplicate Category Name Check	Displays a warning on the Service Category Request form when the requested service category has the same name as an existing service category.
Fix Table Name	Ensures a valid table name on the Service Category Request form.
Hide Draft Services	Hides the Draft Services related list on the Service Category Request form when appropriate.
Propose Table Name	Proposes a valid table name on the Service Category Request form for new service category requests.
Category Published	Displays a help message when Published is selected on the Service Category Request form.
Hide Fulfillers	Hides the Fulfillers related list on the Service Category Request form when appropriate.
Editors Message	Displays a help message for the Editors field when appropriate.
Other Tables Message	Provides information about existing service category tables for the selected Department.
State Message	Displays a help message for the State field.

Business rules

Business rule	Description
Service Query	Restricts users without the catalog_admin role to viewing service records within service categories they are the manager or editor of.
New Service	Provides a message when a new sc_cat_item_producer_service record is created.
Table Name Required	Ensures a service category request has a valid table name before approval.
Remove Fulfiller Role	Removes relevant role from service fulfillers when they are removed from a category.
Category Request query	Restricts users without the catalog_admin role to viewing service category records they are a manager or editor of.
Editor Role	Adds and removes relevant roles from service category editors.
Delete User Role	Removes the relevant role from service category editors when appropriate.
Category Published	Sets State to Published to Catalog when the Published check box is selected on the Service Category Request form.

Business rule	Description
Populate Service Name if Empty	Populates a service name if none is provided.
Add Departmental Services Category	Adds a new service to the Departmental Services service catalog category.
Default Fulfillment User	Makes a category manager the assignee of service tasks if no assignee is specified.
Scratchpad Draft Services Count	Generates field help messages.
Catalog Category Request Approved	Creates components necessary to use of a new service category.
Manager Role	Grants relevant roles to category managers.
New Service Script	Populates the script of a new Service to set assignment group or user.
getDepartmentUsers	Returns the users of a department.
Draft Item Query	Restricts users without the catalog_admin role to viewing draft service records they are a manager or editor of.
Grant Fulfiller Role	Grants relevant role to service fulfillers.
Scratchpad Department Name	Generates field help messages.
Scratchpad	Generates field help messages.
Other Tables For Department	Generates field help messages.
Set Single Catalog from Single Category	Populates a default Catalog for a new service.

Email notifications

Service Creator email notifications

Name	Description
Service Category Published	Notifies the manager of a service category when the category request is approved.
Service Category Rejected	Notifies the manager of a service category when the category request is rejected.
Service Category Request Inserted	Notifies catalog administrators when a new category request is created.
Service Category Created	Notifies the manager of a service category when the category is created.
Service Category Publication Requested	Notifies catalog administrators when publication of a category has been requested.
Service Category Request Opened	Notifies the manager of a service category when a new category request is created on their behalf.

Components created with new service categories

When you publish a new service category using the Service Creator application, the ServiceNow system creates components for the services in that category.

These components are distinct from the components installed with the Service Creator application. The following components are added for each new service category:

Tables created with new service categories

Name	Description
<Department Name> Tasks [<service category table name>]	The table that stores request task records for the service category. This table extends the Task table. The name of this table is defined by the department the service category is associated with, and the **Table name** field on the service category record. A new application menu and modules are created to allow users to access records on this table. Records on this table are numbered using a new Numbers [sys_numbers] record.

User roles created with new service categories

Role	Description
<service category table name>_user	The user role required to access request records for a service category. The Table name for the service category determines the name of the role. Users designated as the manager, editors, or service fulfillers for a service category automatically receive this role. Only users with this role can be assigned task records for the service category. ACLs are created to allow users with this role to access the relevant service task table.

Email notifications created with new service categories

Name	Description
Task commented for group	Notifies the group a service task record is assigned to when a user adds a comment.
Task commented for assignee	Notifies the user a service task record is assigned to when a user adds a comment.
Task closed for group	Notifies the group a service task record is assigned to when the record is closed.
Task worknoted for assignee	Notifies the user a service task record is assigned to when a user adds a work note.
Task assigned to group	Notifies the group a service task record is assigned to when the record is assigned to that group.
Task assigned to assignee	Notifies the user a service task record is assigned to when the record is assigned to that user.
Task worknoted for group	Notifies the group a service task record is assigned to when a user adds a work note.
Task closed for assignee	Notifies the user a service task record is assigned to when the record is closed.
Task opened for user	Notifies the user that opened a service task record when the record is created.

Email notifications created with new service categories (continued)

Name	Description
Task closed for user	Notifies the user that opened a service task record when the record is closed.
Task commented for user	Notifies the user that opened a service task record when a user adds a comment.

Forms created with new service categories

Name	Description
<department name> Task	The form for viewing request task records for the service category. By default, this form uses a layout that includes a formatter to display the questions for the service and the answers provided by the requesting user.

Service catalog categories

Name	Description
<service category name>	The default service catalog category for new services created within a service category.

Service Creator roles

The Service Creator application uses the specific roles.

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

Roles

Role Title [Name]	Description
`[service_category_user]` `[service_category_table_name_user]`	Accesses request records for a service category. The table name for the service category determines the name of the role. Users designated as the manager, editors, or service fulfillers for a service category automatically receive this role.
Catalog Administrator `[catalog_admin]`	Creates, edits, and publishes service categories and services, and creates and edits notifications including template notifications. Catalog administrators are primarily responsible for approving service category requests.
Catalog Manager `[catalog_manager]`	Creates, edits, and publishes services, and designates editors and service fulfillers. A user designated as the manager of a service category receives this role automatically.

Roles (continued)

Role Title [Name]	Description
Catalog Editor `[catalog_editor]`	Creates and edits services. A user designated as an editor of a service category receives this role automatically.

Manage a service

Using the Service Creator, department managers can request a new service category, designate editors and service fulfillers for that category, and create and publish services.

About this task

Editors create and modify services. Service fulfillers complete the tasks generated from service requests.

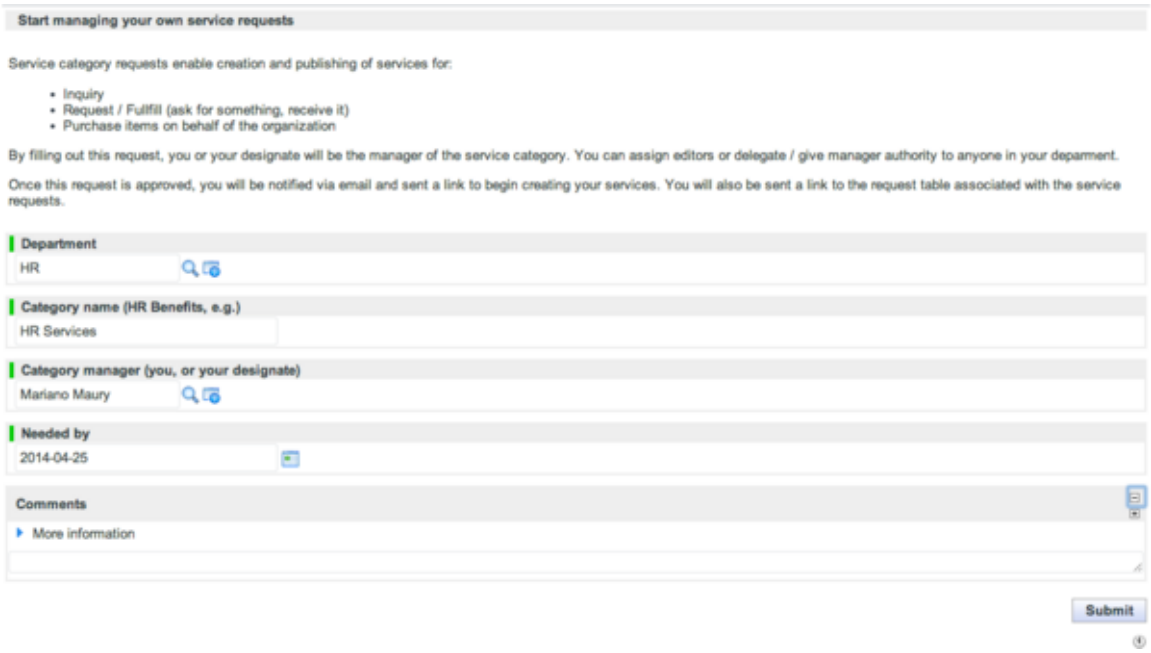
A service category request involves assigning a service category manager, which is typically the department manager who makes the request. After the request is submitted, a catalog administrator approves the request to publish the service category. When the category is published, the service category manager can assign service category editors and service fulfillers, and create services to offer in the service catalog.

To request a new service category:

Procedure

1. Navigate to **All** > **Self-Service** > **Service Catalog**.
2. Select the **Departmental Services** category.

   The *Departmental Services* category is part of the demo data available with service creator. If this category does not exist, a catalog administrator must add the *Service Category Request* catalog item to an existing category.
3. Select the **Service Category Request** item.
4. Change the default values, as necessary (see table).
5. Click **Submit**.



Managing Services

Field	Description
Department	Department this category request is for. By default this value is the department of the current user. Changing this value also changes the *Category name* and *Category manager* values.
Category name	Name for the new service category. By default, ServiceNow uses a name based on the *Department* value.
Category manager	Designated manager for the new service category. By default, ServiceNow uses the manager for the selected department.
Needed by	Date that the new service category should be available.
Comments	Additional comments describing the service category. This information appears as a journal entry on the Service Category form.

Designing services

Service creator includes an interface for designing services.

Using this interface, service category managers and editors can create and publish services, and edit service details.

All services must belong to a service category. If your department or group does not have an existing service category, you must create a new service category before you can design services for that category.

Add a template notification

Adding a template notification.

Procedure

1. Navigate to **All** > **Service Creator** > **Template Notifications**.
2. Click **New**.

3. In the *Send when* field, select **Event is fired**.
4. In the *Event name* field, select **ccrTemplate**.
5. Enter other notification details.
6. Click **Submit**.

   The new template notification creates a notification for all service categories published after that point.

Notification configurations

All service categories start with a set of associated notifications, such as the notification when a task to fulfill a service request is assigned.

Notifications defined in the **Service Creator** > **Template Notifications** module are copied when a user creates a new service category.

Template notifications are distinct from the notifications for the Service Creator application itself, such as the notification when a new service category is approved or rejected. Notifications for the Service Creator application are defined in **Service Creator** > **Notifications**.

A system administrator can add and delete template notifications.

Create the category and table

After the request has been submitted, a catalog administrator can approve or reject the request.

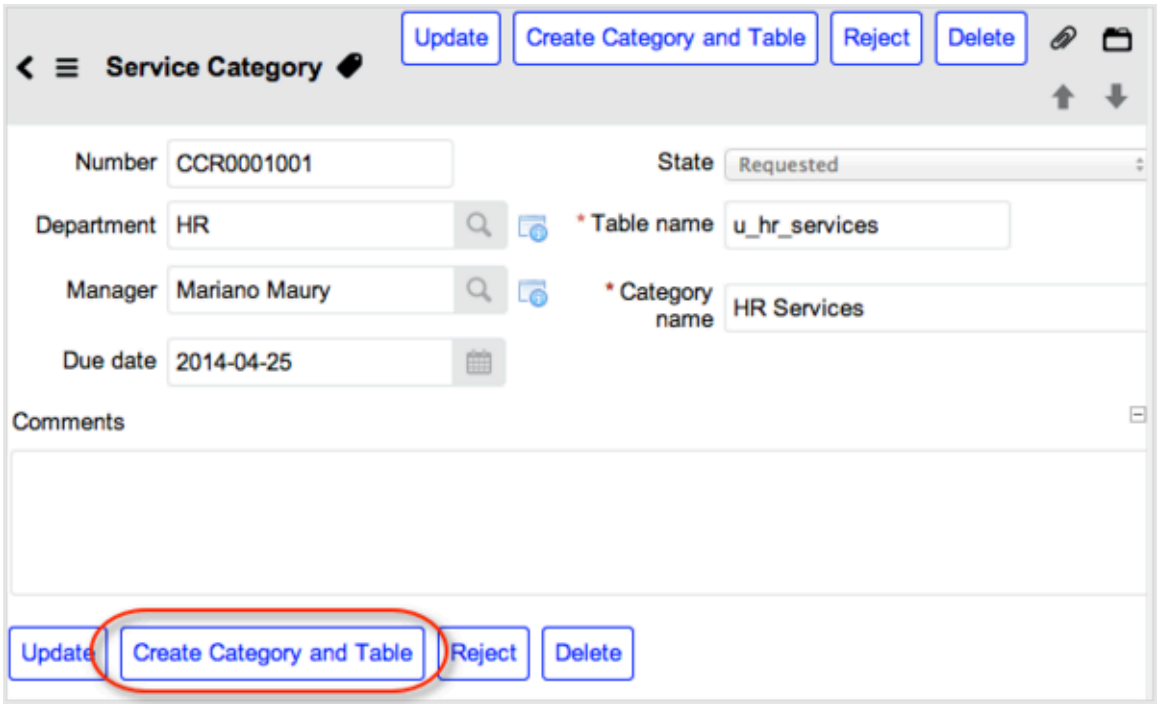
About this task

Approving the request creates a new table for the service category, adds an application to the application navigator using the *Category name* as the application label, and sets the *State* of the service category to *Published to Catalog*.

Procedure

1. Navigate to **All** > **Service Creator** > **Category Requests**.
2. Open a record with a *State* of *Requested*.
3. Review the requested service category. ServiceNow provides a suggested *Table name* based on the *Department*.
   If a service category exists with the specified category name or table name, a message appears under that field. Use a unique value for these fields.
4. Click **Create Category and Table** to approve the request or **Reject** to reject the request.

   If notifications are enabled for the instance, the service category *Manager* is notified of the approval or rejection.



After publishing the service category, you can access the new table by navigating to the new application in the application navigator, or by clicking the **View Task List** related link on the Service Category form.

Related topics

Manage a service

Delete a template notification

Deleting a template notification prevents new service categories from using the notification, but does not delete notifications for service categories that have already been created.

Procedure

1. Navigate to **All** > **Service Creator** > **Template Notifications**.
2. Select a notification record.
3. Click **Delete**.
4. Click **OK** to confirm.

Designate an editor

Editors can create and modify services within a service category.

About this task

Editors automatically receive the catalog_editor role.

The service category manager can designate editors for a published service category.

Procedure

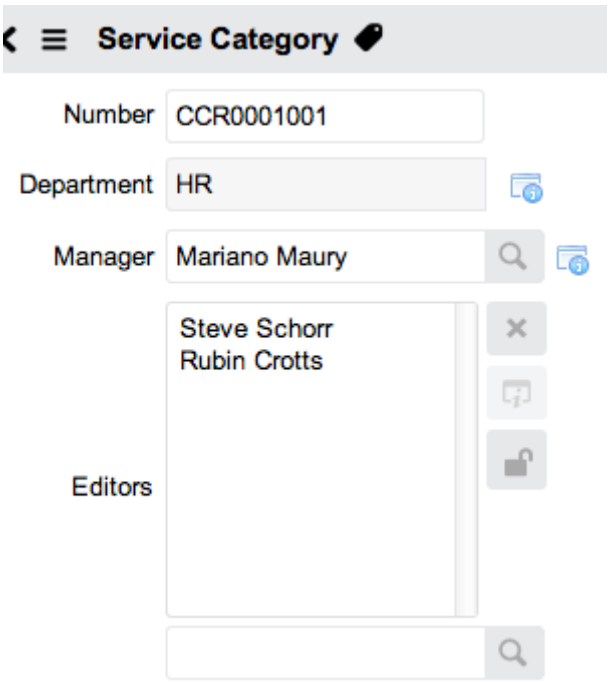
1. Navigate to **All** > **Service Creator** > **My Service Categories**.
2. Select a record with a *`State`* of *`Published to Catalog`*.
3. Click the lock icon beside the *`Editors`* field.

4. Select users to designate as editors using the reference lookup icon.

   Only users in the appropriate department are available for selection.

5. After adding all editors, click **Update**.

   Editors receive the Catalog Editor role.



Designate a service fulfiller

Service fulfillers can complete service requests submitted for a service category.

About this task

Service fulfillers can access applications for service categories they are assigned to, but cannot access the *Service Creator* application.

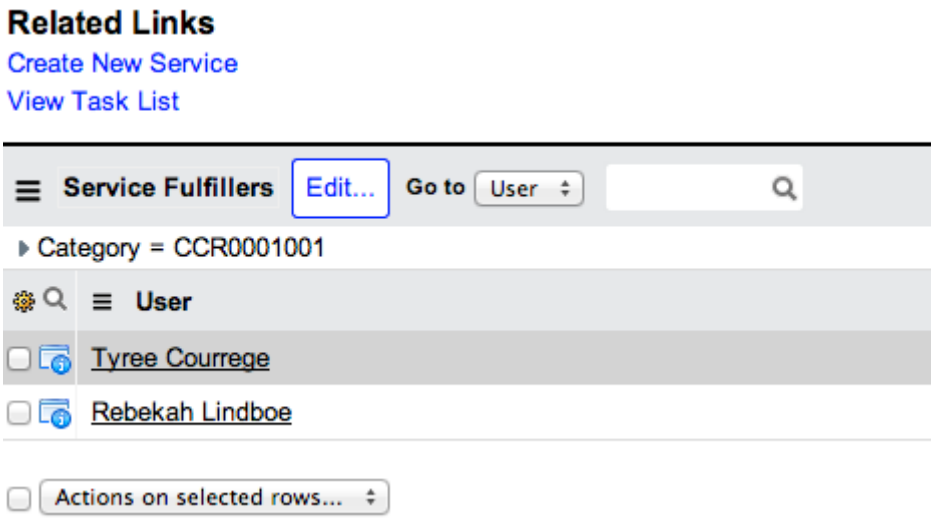
The *Service Fulfillers* related list on the Service Category form displays all users assigned as fulfillers for that service category. The service category manager can designate service fulfillers for a service category.

Procedure

1. Navigate to **All** > **Service Creator** > **My Service Categories**.
2. Select a service category with a *`State`* of *`Published to Catalog`*.
3. In the *`Service Fulfillers`* related list, click **Edit**.
4. Use the slushbucket to add the appropriate service fulfillers.

   Only users in the appropriate department are available for selection.

5. Click **Save**.



Fulfill a service request

End users can request published services through the service catalog.

About this task

When a user requests a service, the ServiceNow system creates a new task for that service category. Service fulfillers complete these tasks to fulfill the request.

New request tasks are automatically assigned to the group or user specified in the *Fulfillment Group* or *Fulfillment User* service setting. If no fulfillment group or user is set, new records are assigned to the service category manager.

Questions for a particular service and the answers entered by the requesting user appear in the *Variables* section on the fulfillment task record.

Procedure

1. Navigate to **All** > **<Your service application>** > **Assigned to me**.
2. Select a record.
3. Review the information presented.
4. Complete the task in accordance with department policies and procedures.
5. Set the *State* of the service request record to *Closed Complete*.
6. Click **Update**.

Publish a service

A service must be published to appear in the service catalog. When first created, new services appear in the Draft Services related list for the service category. Published services appear in the Services related list for the service category. The manager of a service category can publish draft services.

Procedure

1. Navigate to **All** > **Service Creator** > **My Service Categories**.
2. Select a service category with a State of *Published to Catalog*.

3. On the Service Category form, right-click a service in the **Draft Services** related list.
4. Select **Publish**.

Legacy - ServiceNow Studio

The legacy ServiceNow Studio provides an Integrated Development Environment (IDE)-like interface for application developers to work on custom applications in one centralized location. It offers a simple way to create, review, and update application files from a tabbed environment.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

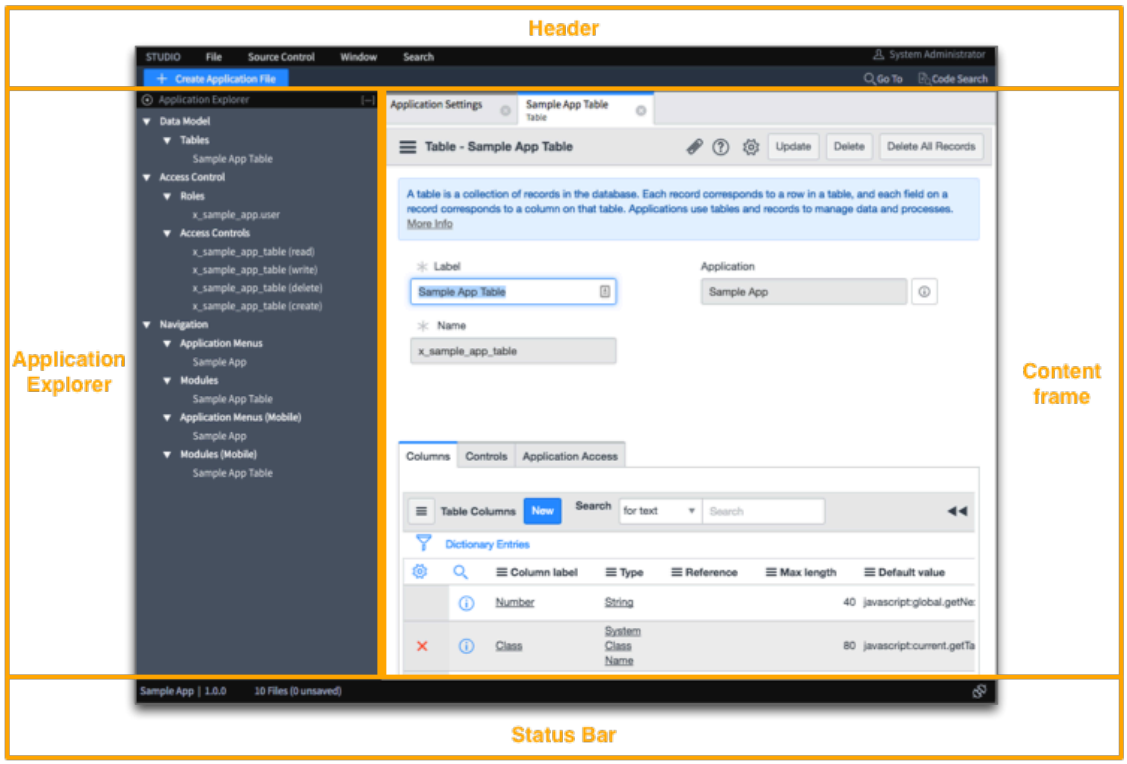
Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

The system opens the new ServiceNow Studio whenever you edit a custom application.

Capabilities

- Create an application and application artifacts.
- Perform code search.
- Integrate with source control.
- Create your company's customizations to store applications that belong to other organizations.
- Use Virtual Agent Designer to create and manage topics, which are blueprints for conversations between a virtual agent and a user. You can design topics that help your users resolve common work issues or guide them through self-service tasks.

Studio



With Studio, application developers can:

- See exactly what files comprise their application in the **Application Explorer**.
- Add new files to their application using a single **Create Application File** interface.
- Navigate to files using familiar search-by-name or by-type behavior with the **Go To** dialog.
- Find code both within and outside an application using the **Code Search** tool.
- Operate on multiple files at once using the tabbed interface.
- Operate on multiple applications at once using multiple studio windows.
- Publish their application to company instances or the ServiceNow Store.
- View information about their current application from the **Status Bar**.

Which builder should I use to create an app?

Types of builders

Want to build an app easily, without code?

Creator Studio specializes in helping you craft request-fulfillment applications without writing code. For example, an application to request office supplies by filling out a form, and someone approves or denies your request. For more information, see Exploring Creator Studio.

Need a more general app but still want low-code options?

App Engine Studio lets you build a broader range of apps than Creator Studio without being a programming pro. For more information, see Exploring App Engine Studio.

Are you a developer who wants more control in a centralized user interface?

Build apps smarter and deliver them faster with the new ServiceNow Studio. ServiceNow Studio empowers platform developers with a modern, unified environment for building on the ServiceNow AI Platform. ServiceNow Studio features streamlined navigation to applications and metadata, integrated low-code tools, efficient tracking and packaging of development work that accelerates development processes and enhances productivity. For more information, see Exploring ServiceNow Studio.

Are you a developer who wants to use industry-standard development tools and processes?

The ServiceNow IDE and ServiceNow SDK support developing applications in source code with ServiceNow Fluent, creating JavaScript modules, and using third-party libraries. ServiceNow Fluent is a domain-specific programming language for creating application metadata in code.

The ServiceNow IDE is an implementation of Visual Studio Code for the Web on the ServiceNow AI Platform. The ServiceNow SDK uses Visual Studio Code Desktop locally. For more information, see Building applications in source code.

Working with the Studio UI

Parts of the Studio UI

UI element	Description
Header	Displays menus and controls.
File Menu	Contains a list of application-specific options. • **Create File** • **Import From Source Control** • **Create Application** • **Publish** • **Settings** • **Switch** • **Manage Developers** • **Launch Script Debugger**
Source Control Menu	Contains a list of source control options. • **Link To Source Control** • **Edit Repository Configuration** • **Apply Remote Changes** • **Commit Changes** • **Stash Local Changes**

Parts of the Studio UI (continued)

UI element	Description
	• **Switch Branch** • **Create Branch** • **Create Tag** • **Manage Stashes** • **View History**
Window Menu	Contains a list of tab management options. • **Close Current Tab** • **Close All Tabs** • **Close Other Tabs** • **Close Unmodified Tabs**
Search Menu	Contains a list of search options. • **Go To** • **Code Search**
User name	The header displays the name of current user.
Create Application File	Allows developers to add an application file to an application.
Go To	Search for application files by name or type.
Code Search	Search within application files for a text string. Search options include: • Restrict search to a particular table • Include all applications
Application Explorer	Displays a list of application files by type. Resize the Application Explorer to see more about application files or to provide more space for the content frame.
Collapse All	Collapses all nodes in the application explorer.
Expand All	Expand all nodes in the application explorer.
Data Model > Tables	A list of application tables. Click a table name to display and edit it in the content frame.
Access Control	A list of application access elements such as: • **Roles** • **Access Controls**

Parts of the Studio UI (continued)

UI element	Description
	Click a record name to display and edit it in the content frame.
Navigation	A list of application navigational elements such as: • **Application Menus** • **Modules** • **Application Menus (Mobile)** • **Modules (Mobile)** Click a record name to display and edit it in the content frame.
Content frame	Displays a detail form for each record in its own tabs.
Welcome to Studio	A list of keyboard shortcuts.
Tabs	Each tab contains a specific application file record identified by the record name and file type. Click a tab to display and edit the record. A tab with a blue circle icon indicates that the record contains unsaved changes.
Status Bar	Displays information about the application and the source control integration.
Application name	The status bar displays the name of the current application.
Application version	The status bar displays the current application version.
Total files	The status bar displays the total number of application files.
Unsaved files	The status bar displays the current number of application files with unsaved changes.
Source control integration status	The status bar displays an icon indicating the current status of the source control integration.

Related topics

Contextual development environment

Legacy - Access ServiceNow Studio

Application developers access ServiceNow Studio to create, import, or open applications.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Role required: admin

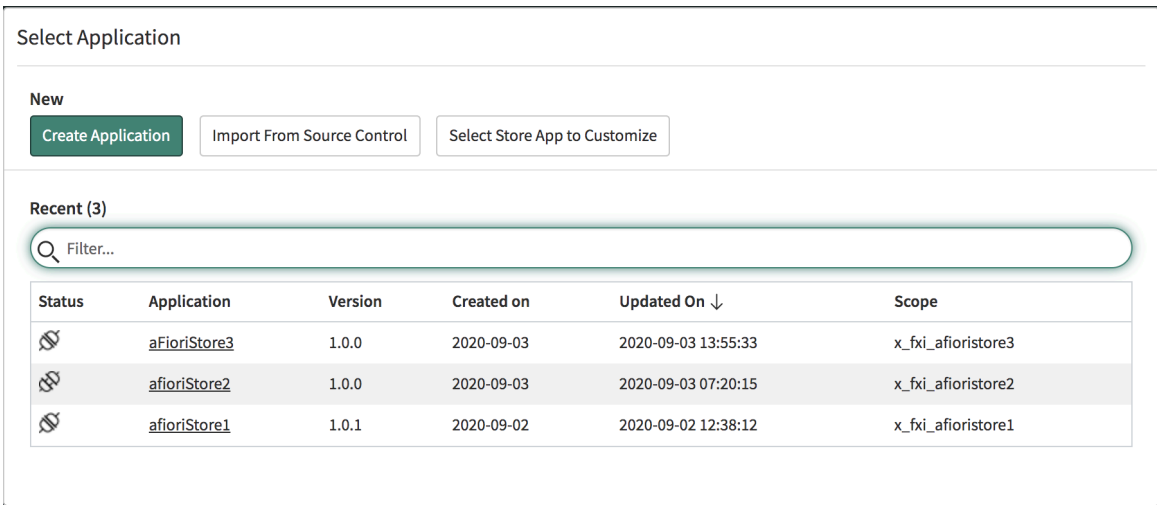
Procedure

1. Select the path available to your role.

Paths to Studio

User role	Action to open Studio
admin	Navigate to **All** > **System Applications** > **Studio**.

The Studio opens in a new browser tab and displays the Select Application modal dialog.

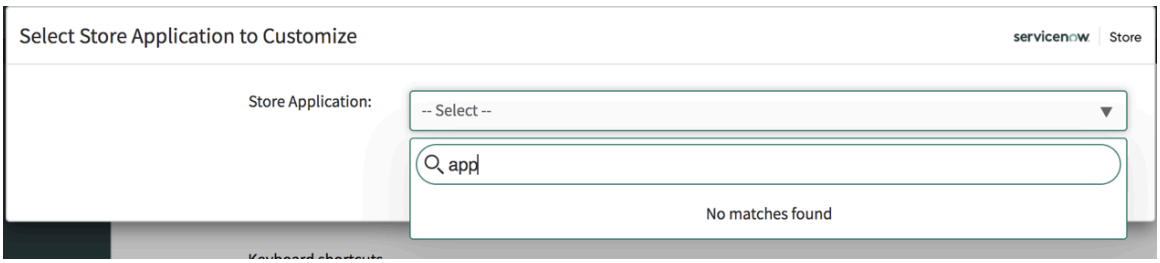


2. To create an application, click **Create Application**.
   The system displays a list of application creation options.
3. To import an existing application from source control, click **Import From Source Control** list.
   Studio opens **Import From Source Control**page.
4. To view an existing application, click the application name from the **Applications** list.

   The list also includes any store applications or scoped applications installed via plugins that are customized.

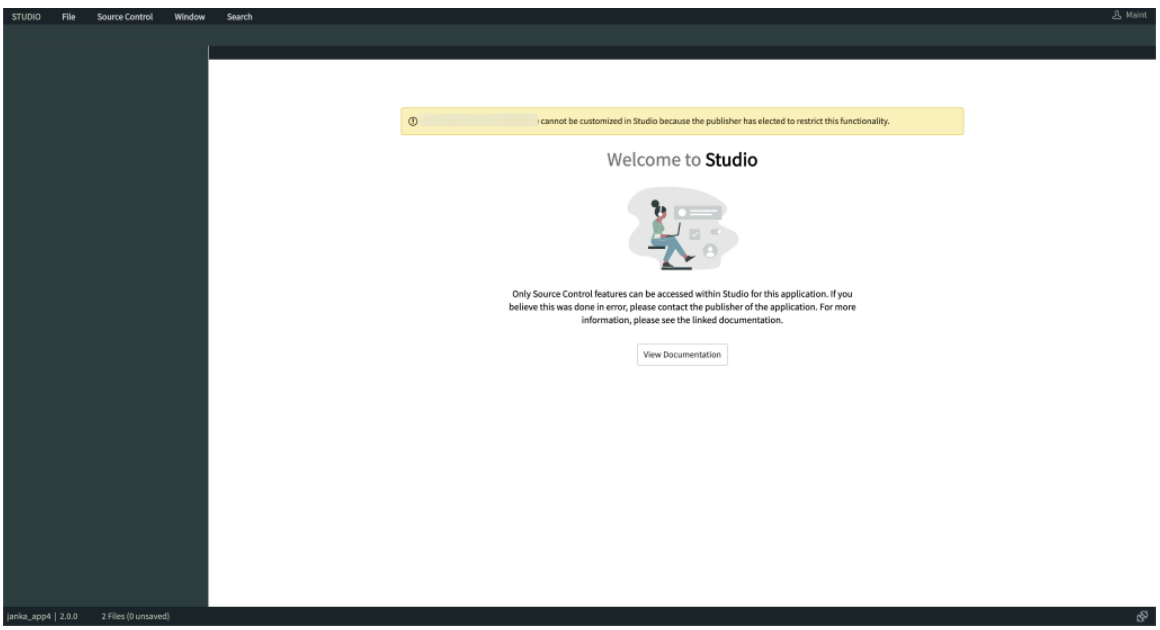
Note:
In a few scenarios for scoped applications installed via plugins, the **Name** field in the Store Applications (sys_store_app) table doesn't match the plugin name. To make sure you are choosing the correct application, search the Select Store Application list picker with the **Name** of the application mentioned in the sys_store_app. For example: The plugin **Agile - Scaled Agile Framework - Essential SAFe** creates the sys_store_app record with the name "SAFe". Therefore you would search the Select Store Application list picker with the word "SAFe".

5. To open an installed store application so you can create your company's application-customization for it, click **Select Store application** and select the store application from the list picker display.



Studio opens the selected application.

Note:
The linked icon indicates that there are uncommitted local changes in the application. The property is ignored for your company applications in development. The **Can Edit Application in Studio** property for a store application determines if the application can be edited in Studio. If set to false, you see a warning that you cannot edit the application in Studio. But you can open the application in Studio to use Source Control features or to publish its customizations to the application repository. The default is true for new applications. The application's owner can change the property value before releasing a new version.



Legacy - ServiceNow Studio keyboard shortcuts

ServiceNow Studio supports various keyboard shortcuts to manage and edit application files.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Studio keyboard shortcuts

Keyboard combination	Description
Windows: Control+Shift+O *Mac:* Command +Shift+O	**Go To**. Open any file in your application.
Windows: Control+Shift+C *Mac:* Command +Shift+C	**Create New**. Create an application file.
Windows: Control+Shift+F *Mac:* Command +Shift+F	**Code Search**. Search for any file or value.
Windows: Control+Shift+X *Mac:* Command +Shift+X	**Close Tab**. Close the current tab. If the tab contains unsaved changes, the system prompts the user to save them.

Related topics

Legacy - ServiceNow Studio

Legacy - Add an application file to an application

Studio allows application developers to add new application files by type.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Role required: admin

This procedure requires creating a scoped application.

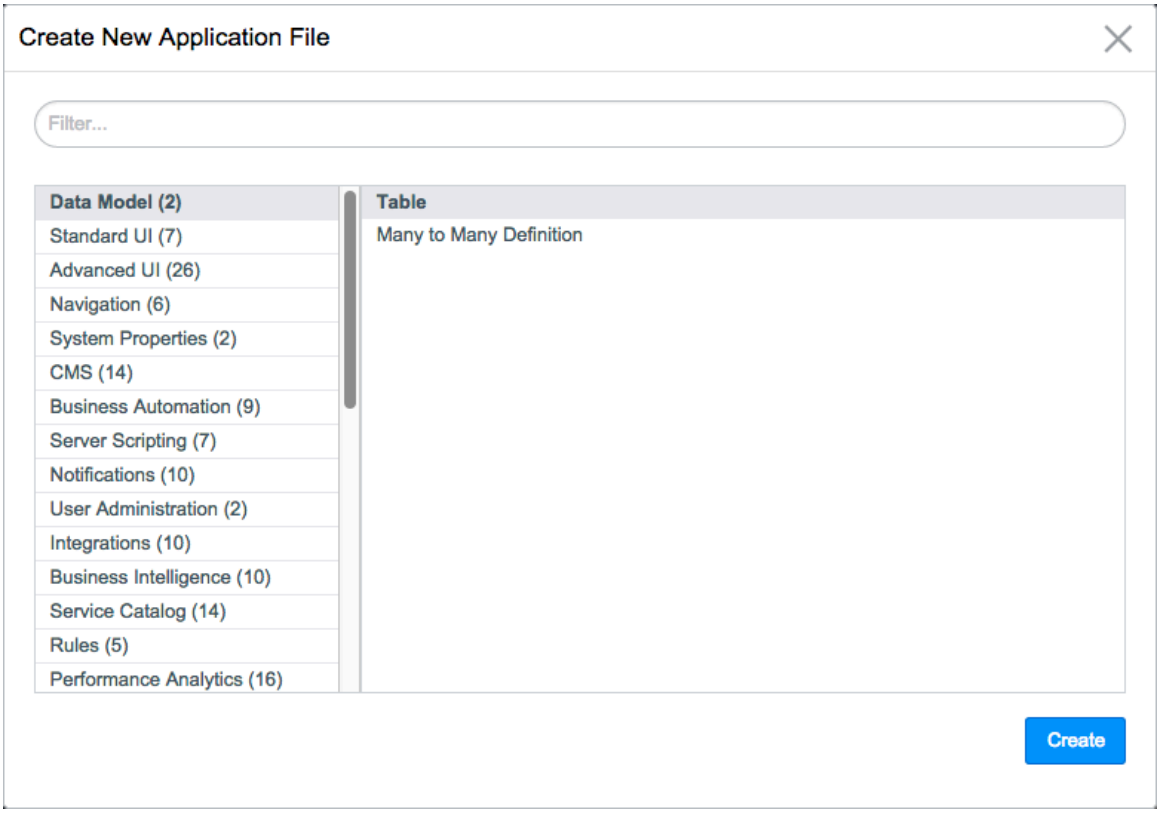
About this task
You can add application files to update the features of a custom application.

Procedure
1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. From the **Develop** tab, click the **Edit** button next to the application you want to modify.
   The system opens the application in the Studio.
3. From the content frame, click **Create Application File**.
   You can also use a Studio keyboard shortcut.

Studio keyboard shortcut

Keyboard combination	Description
Windows: Control+Shift+C *Mac:* Command+Shift+C	**Create New**. Create an application file.

Studio opens the Create New Application File pop-up window.



Note:
Not all application file types will display in Studio. Some types of application files needs to created outside Studio, such as dashboards, even though they extend sys_metadata table.

4. Find the application file type you want to create.

Option	Description
Search by application file name	In the **Filter** entry field, enter the name of the application file.
Search by category	From the left pane, select a category name.

5. From the results pane, select the application file type you want to create.
6. Click **Create**.
   The system displays a blank form for the application file type in a new studio tab.

What to do next
Enter the necessary fields for the particular application file type you selected.

Related topics
Application files

Legacy - Publish an application from ServiceNow Studio when linked to Source Control
You can publish a custom application from ServiceNow Studio when linked to Source Control.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Role required: admin

About this task
When you publish an application from ServiceNow Studio that is linked to Source Control, there is a different outcome than when you publish via the sys_app or sys_store_app **Publish** related link.

Procedure

1. When you select **Publish**, the sys_app or sys_app_customization record for the application is updated with the new version.
2. The current state of the application is committed to Source Control, including any untracked or uncommitted changes.
   The value of the **glide.sourcecontrol.default_commit_mode** property is ignored.
   This occurs because when the application is published, all the untracked and uncommitted changes are also published. Therefore, the state of the application in the Git repository matches what is published. See the Legacy - Commit changes topic for more information about the **glide.sourcecontrol.default_commit_mode** property.
3. A Source Control tag is created for the new version and the application is published.
   If needed, the sys_app record is updated with the new store correlation ID.

   Note:
   If your application is linked to Source Control and you publish a new version outside of Studio, a source control commit and tag are not created.

Legacy - Search for an application file by name or type

Application developers can use Studio to search for application files.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Role required: admin

This procedure requires creating a scoped application.

About this task
You can search for application files to add, remove, or update the features of a custom application.

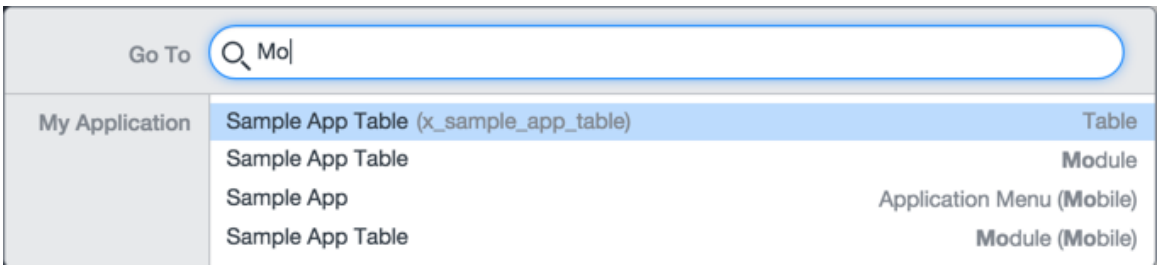
Procedure

1. Navigate to **All** > **System Applications** > **Applications**.
2. From the **Develop** tab, click the **Edit** button next to the application you want to modify.
   The system opens the application in the Studio.
3. From the header, click **Go To**.
   You can also use a Studio keyboard shortcut.

   Studio keyboard shortcut

Keyboard combination	Description
Windows: Control+Shift+O *Mac:* Command+Shift+O	**Go To**. Open any file in your application.

   Studio opens the Go To window.
4. Enter a search string.



   Studio displays a list of matching application files as you type.
5. From the list of search results, click a record name.
   Studio opens the application file record in a new tab in the content frame.

Legacy - Search within application files

Studio allows application developers to search within application files for matching record values.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Role required: admin

This procedure requires creating a scoped application.

About this task

You can search within application files to add, remove, or update application file values.

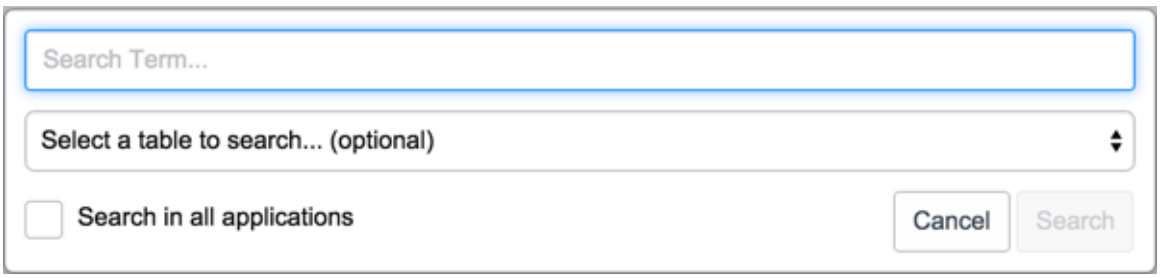
Procedure

1. Navigate to **All** > **System Applications** > **Applications**.
2. From the **Develop** tab, click the **Edit** button next to the application you want to modify.
   The system opens the application in the Studio.
3. From the header, click **Code Search**.
   You can also use a Studio keyboard shortcut.

   Studio keyboard shortcut

Keyboard combination	Description
Windows: Control+Shift+F *Mac:* Command+Shift+F	**Code Search**. Search for any file or value.

   Studio opens the Search pop-up window.
4. In **Search term**, enter a search string.

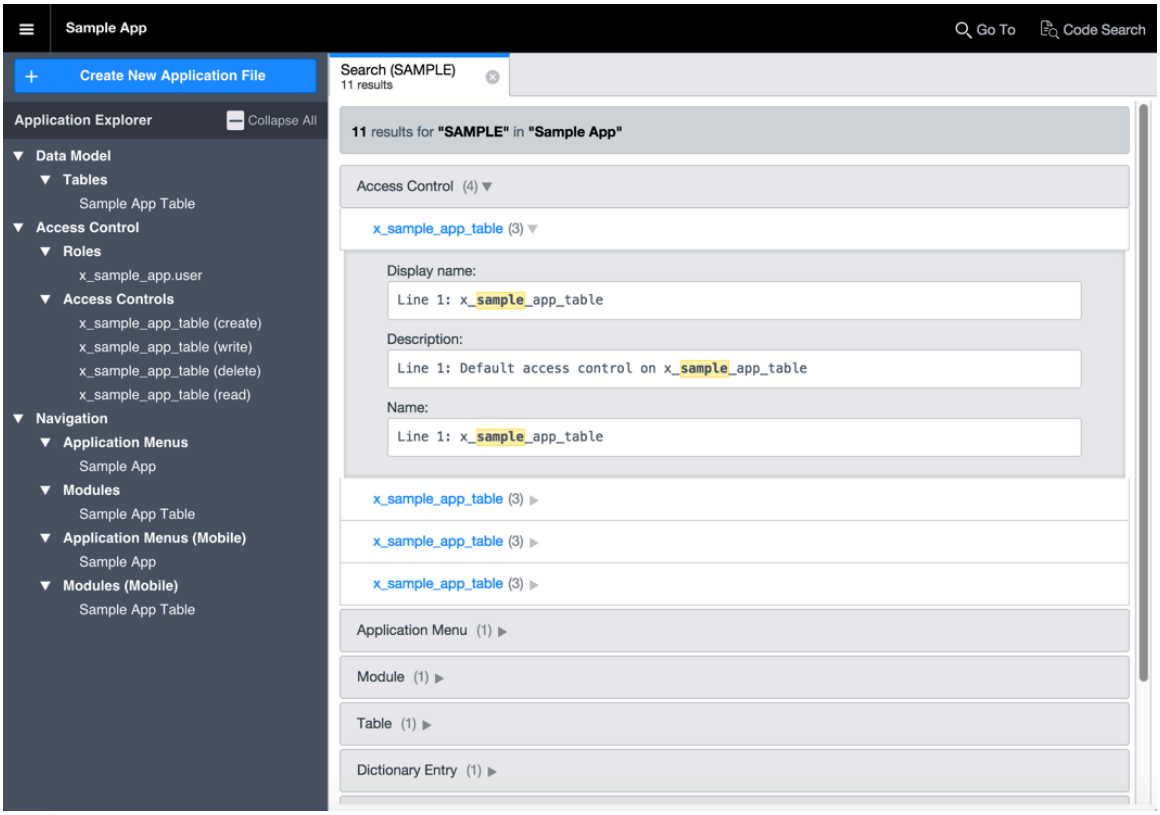


5. **Optional:** Select any additional search criteria.

Option	Description
Select a table to search	Search for matches only within the selected file type.
Search in all applications	Search for matches throughout the instance, not just within the current application.

   Note:
   Searches across all applications can take a long time.

6. Click **Search**.
   The Studio conducts a case-insensitive search of the application files you selected. While the search is running, Studio displays a search progress indicator. You can click the cancel icon to stop the search. When the search is complete, the system opens a new tab in the content frame to display the search results by application file type. Each application file type displays the number of matching search results.



7. From the search results tab, expand an application file type and click a record name.
   Studio opens the application file record in a new tab in the content frame.

Legacy - Update a custom application record

You can update a custom application record to add new features or change application functionality.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Role required: admin or a delegated developer role granting full access

About this task

You can only update applications in development on your local instance. You cannot edit applications downloaded from your company application repository or the ServiceNow Store.

Procedure

1. Navigate to **All** > **System Applications** > **Applications** > **Studio**.
2. Click the application name or the **Edit** button for the application you want to update.
   The system displays the application and application files in Studio.
3. Click **File** > **Settings**.

Studio opens a tab containing the Custom Application record for the current application.

4. Fill in the fields, as appropriate.

Custom Application form

Field	Description
Name	[Required] Enter a label for the application. Changing the name does not change any other field value derived from the application name such as the **Scope**, **Menu**, or **User role** fields.
Version	Enter version information for the application. Both the ServiceNow application repository and the store use this value to determine whether updates are available to your application users. **Note:** To publish the application in the ServiceNow Store, the version must conform to the application certification standards.
Application Scoping	
Scope	[Read Only] Displays the unique application scope set during the creation process. You can change this value only by deleting and recreating the application with a new value. For more information about the protections offered, see Application scope.
Application administration	Select whether to protect sensitive application data by restricting how users acquire application-specific roles. See Application administration.
Design and Runtime	
JavaScript Mode	The JavaScript standard that the application supports. Select **ECMAScript 2021 (ES12)** to support features in ECMAScript 12th edition or **ES5 Standards Mode** to support features in ECMAScript 5th edition. Select **Compatibility Mode** to support earlier ECMAScript editions. For more information, see Set the JavaScript mode for an application.
Runtime Access Tracking	Select how the application handles script access requests to resources in other applications. Select **None** to authorize all access requests to cross-scope resources without logging them. Select **Tracking** to log and authorize all access requests to cross-scope resources. Select **Enforcing** to log access requests to cross-scope resources but require an administrator to authorize each request.
Restrict Table Choices	Clear the option to allow the application to see tables from other application scopes. Select the option to restrict design choices to only tables in the same application.
Subscription Management	
Licensable	Specifies whether the application is tracked by the Subscription Management application.
Subscription requirement	Not applicable for ServiceNow customers who build custom applications for their own use. Used only by partners who sell and monitor the usage of resellable applications on the ServiceNow Store. Specifies whether the application requires a separate subscription (**Required**) or is monitored only.

Field	Description
Subscription model	Not applicable for ServiceNow customers who build custom applications for their own use. Used only by partners who sell and monitor the usage of resellable applications on the ServiceNow Store. Specifies how the Subscription Management application tracks usage. See Types of subscriptions in Subscription Management.
Primary Menu	
Menu	Select the application menu where you want to display modules. For more information about menus and modules, see Create an application menu.
End user access	
User role	Select the user role required to access the application menu. For more information about user roles, see Create a role Create a role.
Short description	Enter a description of the application purpose or usage.
Logo	Select the image the system displays in the applications list and ServiceNow Store.
Application Files	View configuration records associated with this application in platform feature tables.
Dependencies	View or add tables or applications on which this application depends. The system automatically adds records to this list when you extend tables or when another application creates application files for this application. Add script-based dependencies. See Dependencies for custom applications.
Cross scope privileges	View or create cross-scope privilege records to determine which script operations and targets the system allows to run. See Cross-scope privilege record.
Design Access	View or specify which other applications have design access to tables or records in this application. See Application design access record.

5. Click **Update**.

Legacy - Switch between applications

Application developers can switch between applications without leaving the Studio environment.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Before starting this procedure, you must create at least one custom application with its own application scope.

Role required: admin

About this task

The contextual development environment restricts some changes when the application files belong to another application. Switching to the application that owns the application files ensures that you change the proper application.

Procedure

1. Navigate to **All** > **App Options Menu** > **Switch Applications**.
   The system displays the list of applications.
2. Click the application you want to switch to.
   The system reloads Studio to display the selected application.

Legacy - Global application file management

Once you create a globally scoped application in the ServiceNow Studio, you can add existing globally scoped files to it, remove files from it, or move application files between global applications.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Legacy - Add a file from the global scope to a global application

Add existing globally scoped files into a selected global application. You can search for files in another globally scoped application by the update set name, table, or file name.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

The Global App File Management plugin must be activated.

Role required: admin

About this task

When you select an application in the application picker, application files are automatically assigned to it. You cannot move an application file into or out of a scoped application. See Application scope.

Procedure

1. Navigate to **All** > **System Applications** > **Applications** > **In Development**.
2. To the right of the application name, click **Edit in Studio**.
3. In Studio, click **File**, then select **Add Existing Files**.

4. Use any of the following options to find existing globally scoped application files that you can add into the current globally scoped application.

   Note:
   You can view up to 250 files at a time. If you want to view more files, perform another search with more specific search criteria.

5. Click **Continue** to list application files that are associated with the selected update set, table, or name.
   You can filter the results as needed.
   The system displays the filtered results but omits any files that are already associated with the current global application. The system displays the file type, application, when the file was last updated, and if it is a customized file.
6. Select the files that you want to add, and then click **Add**.
   The Confirm Changes dialog appears, stating that the files being moved are now considered customizable files and may appear on a skip list during an upgrade.
7. Click **Continue** to add the selected files.

   Note:
   The ability to add globally scoped files to globally scoped applications also allows you to move dictionary collection entries to a custom global scope. This action may result in unexpected cross-scope failures as the **sys_db_object** table and collection are not in the same scope anymore. The implication of this action is that CRUD operations (that are allowed based on sys_db_object application access and existent cross-scope privileges) would not be permitted without an obvious reason.

Result

The system moves the selected files to the selected application from the global scope, and flags them as customizable files. For example, if you select the sys_ui_policy file, the associated sys_ui_policy_actions file is also added to the selected application, and flagged as customizable. You see error messages when errors occur in the add process, or a confirmation when your files are successfully added to the globally scoped application.

Legacy - Remove a file from a global application

Remove selected files from the selected global application and return them to the previous global application that owned the file. If there is no previous owner, file is returned back to global scope. The removed files are still considered customized files.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

This option is only available if the Global Application File Management plugin is active.

Role required: admin or a delegated_developer role that grants full access.

Procedure

1. Navigate to **All** > **System Applications** > **My Company Applications** > **In Development**.
2. To the right of the application name, select **Edit in Studio**.
3. In Studio, select **File** > **Remove Files**.
   The system displays a list of the files in this global application.
4. Select the file that you want to remove from this application.
5. Click **Remove**.

Legacy - Move an application file between global applications

Move selected application files and their descendant files to a selected global application. When you select a global application, the selected application files are automatically assigned to it. You cannot move an application file into or out of a scoped application.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

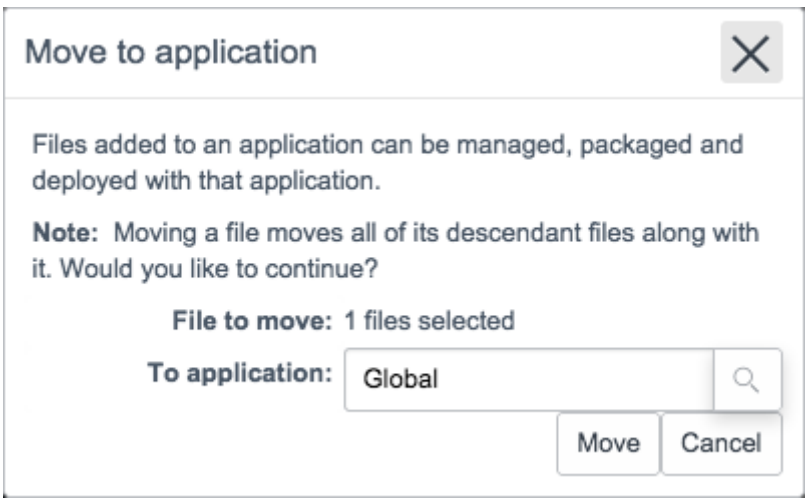
Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Roles required: admin or a delegated_developer role that grants full access.

This option is only available if the Global App File Management plugin is active.

Procedure

1. Navigate to the application file in a list or form view.
   For example, navigate to **All** > **System UI** > **UI Policies** to select UI policy files to move.
2. Select the check box beside each file you wish to move, and then select **Move to Application** in the **Actions on selected rows** choice list.
3. Select the global application.



4. Click **Move**.
   The selected application files and its descendant files are moved to the selected global application.

Legacy - Add multiple files from global update sets to a global application

Add multiple globally scoped application files at once from one or more global update sets to a new or existing global application.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Role required: admin

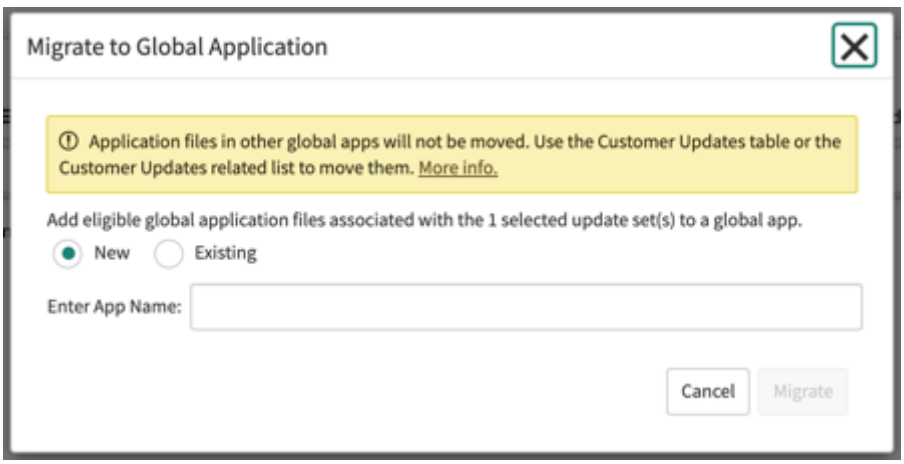
The Global App File Management plugin must be activated.

About this task

This feature is for customers who manage their customizations to globally scoped files via update sets. You can migrate these files in bulk to new or existing global applications. The global application can then leverage ServiceNow CICD tools which support only applications. Select multiple update sets to move globally scoped application files to a new or existing global application. All the eligible application files associated with the customer updates in those update sets will be moved. This operation moves only the application file as it is loaded on the instance. The payload in different customer updates to the same file is ignored.

Procedure

1. Navigate to **All** > **System Update Sets** > **Local Update Sets**.
2. Select the check box beside each update set you want to move, and then select **Migrate to Global Application** in the **Actions on selected rows** list.
3. Enter the name of a new global application or select an existing global application.



4. Select **Migrate**.

Note:
Selecting update sets to migrate does not move application files that are part of other global applications. Only files that are in the global scope are moved. To add files that are part of other global applications, see Legacy - Add multiple files from customer updates to a global application.

Trouble?

If an application file associated with an update was not moved, it could be because:

- The application file is not global.
- The application file is part of another global application.
- The application file was deleted.
- The application file is not eligible to move to another global application. For example, a `sys_app` file. \
- The application file is read-only.

Legacy - Add multiple files from customer updates to a global application

Add multiple globally scoped application files at once from one or more customer updates to a new or existing global application.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Role required: admin

The Global App File Management plugin must be activated.

To add global files that are part of other global applications, `glide.source_control.customize_files_in_other_global_apps` must be true.

About this task

This feature is for customers who manage their customizations to globally scoped files via update sets. You can migrate these files in bulk to new or existing global applications. The global application can then leverage ServiceNow CICD tools which support only applications. Select multiple customer updates to move globally scoped application files to a new or existing global application. All the eligible application files associated with the customer updates will be moved. This operation moves only the application file as it is loaded on the instance. The payload in different customer updates to the same file is ignored.

Procedure

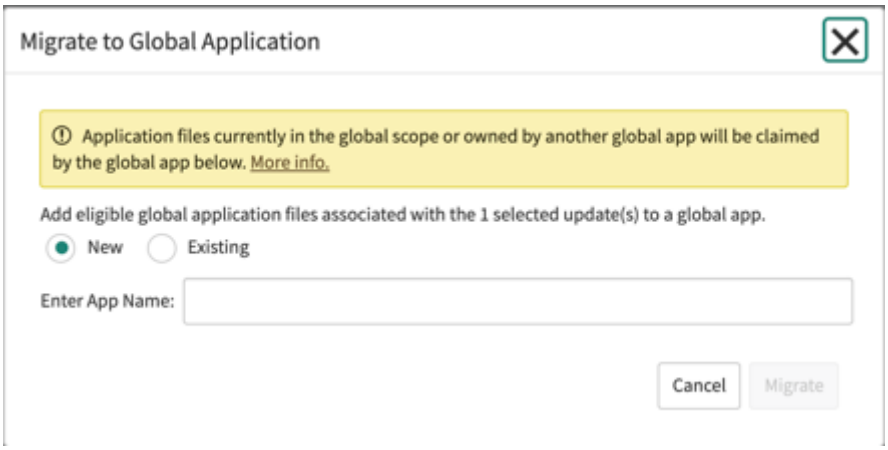
1. Navigate to **All** > **System Update Sets** > **Local Update Sets**, open an update set in Form view and go to its Customer Updates Related List.
   Or

Navigate to **System Maintenance** > **Customers Updates [sys_update_xml]** table and add updates from multiple update sets.

2. Select the check box beside each update you want to move, and then select **Migrate to Global Application** in the **Actions on selected rows** list.
3. Select **Migrate**.

> **Note:**
> To learn how to add multiple files from global update sets to a global application, see Legacy - Add multiple files from global update sets to a global application.

4. Enter the name of a new global application or select an existing global application.



Trouble?

If an application file associated with an update was not moved, it could be because:

- The application file is not global.
- The application file is part of another global application.
- The application file was deleted.
- The application file is not eligible to move to another global application. For example, a `sys_app` file.
- The application file is read-only.
- The application file is already associated to the global application being migrated to.

Legacy - Review claimed or skipped global files

Review files when one or more of your files has been skipped or claimed by another application.

> **Important:**
> Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.
>
> Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

During the development of a global application, warnings about skipped or claimed files may display on various progress screens. Skipped or claimed files can occur when the files in one application are claimed by other application(s).

These warnings display during the installation of an application from the Application or Git repositories or when you publish your application. When you see that your application files have been either skipped or claimed by another application, contact the team that owns the other application to determine the best course of action for both teams.

What your options are

Only one version can be loaded for production. The options are as follows.

- Decide with the other team from which application the file should be removed so that only one file is loaded for production.
- Use the Merge tool: If changes to files in both apps are needed in production, use the merge tool to pull the files into a new common application that serves both parties effectively.

To learn more see the Legacy - Resolve conflicts topic.

Legacy - Automatic recovery of draft records

Studio can maintain a version of any open existing record with unsaved changes. Users can recover unsaved changes when their user session ends unexpectedly due to network latency, session timeout, or service interruption.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

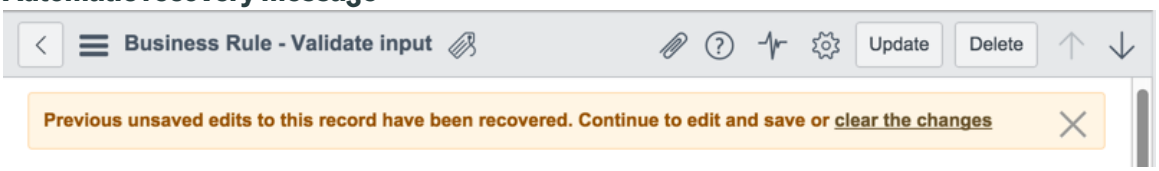
Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Automatic recovery only applies to:

- Records open in Studio. The system does not save all draft records.
- Changes made in desktop (non-mobile) browsers. The system does not save draft records from mobile browsers.
- Changes made to existing records. The system does not save draft changes to new records.
- Records containing unsaved changes that are the most recent update to the record. The system discards draft changes when another user has updated the same record.
- Records for tables that extend the Application File [sys_metadata] table.

After the user re-establishes a session, Studio displays a message for each record with recovered changes.

Automatic recovery message



For each recovered record, users can:

- Continue editing and save the record.
- Clear the changes from the recovery cache.

The system automatically clears changes from the recovery cache when a user:

- Saves the record. The system removes the saved record from the recovery cache.
- Confirms navigating away from a record without saving changes. The system removes the abandoned record from the recovery cache.
- Reaches the recovery cache limit of 5 MB of changes. The system removes the record with the oldest update date-time stamp.

By default, automatic recovery is enabled for all Application File [sys_metadata] tables while working from Studio.

Administrators can configure automatic recovery properties to:

- Disable or re-enable automatic recovery.
- Specify a list of field types to exclude from automatic recovery.

Users can enable or disable automatic recovery as a user preference.

Related topics

User preferences

Legacy - Auto recovery properties

Administrators can configure how Studio handles the recovery of draft records by navigating to **Auto Recovery** > **Properties**.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Auto recovery properties

Property	Description
Enable Auto Recovery *glide.ui.auto.recovery*	By default, automatic recovery is enabled for all Application File [sys_metadata] tables while working from Studio. Set this property to disable automatic recovery of records.
Comma separated list of tables not supported for auto recovery *glide.ui.auto.recovery.unsupported.tables*	By default, automatic recovery is enabled for all Application File [sys_metadata] tables while working from Studio. Set this property

Auto recovery properties (continued)

Property	Description
	to exclude specific tables from automatic recovery.
Comma separated list of field types not supported for auto recovery *glide.ui.auto.recovery.unsupported.field.types*	By default, automatic recovery supports all field types. Set this property to exclude certain field types from automatic recovery.
Comma separated list of field types to exclude from auto recovery *glide.ui.auto.recovery.exclude.field.types*	By default, automatic recovery is enabled for all field types. Set this property to exclude certain field types from automatic recovery.

Legacy - Auto recovery dictionary attribute

Administrators can configure how the Studio handles the recovery of draft records with a dictionary attribute.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Auto recovery properties

Dictionary attribute	Description
Exclude Auto Recovery *exclude_auto_recovery*	Disables automatic recovery of draft records for this table.

Related topics

Dictionary attributes

Legacy - Source Control integration

Enable application developers to integrate with a Git Source Control repository. Save and manage multiple versions of an application from a non-production instance.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Linking an application to Source Control enables all application developers on a non-production instance to:

- Import applications from a Git repository.
- Pull and apply remote changes from a Git repository.
- Commit all local changes on the instance to a Git repository.
- Create tags to permanently link to a given version of an application.
- Create branches to maintain multiple versions of an application simultaneously.

Integration requirements

To link an application to source control:

- The user must have the admin role.
- The non-production instance must have network access to the Git repository.
- Each application must be within its own Git repository.
- The repository user credentials must grant read and write access.

Note:
All application developers on the instance share a single set of credentials per repository.

Options available from ServiceNow Studio

After linking an application to Source Control, application developers can use ServiceNow Studio to manage the repository. From Studio, developers can:

- Edit the application repository credentials.
- Commit all local changes on the instance.
- Apply remote changes from the repository.
- Create a branch.
- Switch branches.
- Import an application from a remote repository.

It is not recommended to use Source Control to manage applications on a production instance, deploying to production may lead to unintended consequences, see Legacy - Production deployment tips. Instead, you can manage applications on a production instance using the application repository, an update set, or the ServiceNow Store. For more information about managing applications on a production instance, see Application sharing.

Options available from a Git repository

The ServiceNow platform offers limited support for modifying linked application files outside of an instance. From Git, developers can:

- Move application files to a different Git directory structure.
- Edit application files outside of ServiceNow Studio.

The system generates a properties text file called `sn_source_control.properties` at the root level of the repository. To move application files to a different Git directory structure, application developers can set the *`path`* parameter to specify the subfolder path containing their application files. For example, if you moved your application to the `src/app` subfolder, set the *`path`* to `path=src/app`.

The system generates a `checksum.txt` file in the Git repository to determine if any application files have been changed outside of Studio. When the checksum value from the file matches the current checksum value, the integration skips the validation and sanitization process. When the checksum values do not match, the integration validates and sanitizes the application files as part of the Source Control operation. The sanitization process:

- Creates upgrade log entries for each sanitization action taken.
- Removes unsupported folders and files from the repository.
- Aborts all Source Control operations when a system application file fails XML schema validation. For example, if a database dictionary record fails XML schema validation, the system aborts all operations.
- Skips the current Source Control operation when a non-system application file fails XML schema validation.

The Source Control integration sanitizes only content within the application path listed in the `sn_source_control.properties` file. Repository content outside the application path is ignored.

MID Server support

Use an existing MID Server to connect to a Source Control repository. Linking or importing an application through a MID Server enables access to repositories behind a firewall.

Configure MID server for source control integration

Configure a MID server for Source Control Integration to enable communication and the movement of data between a ServiceNow instance and external applications, data sources, and services.

- Add "bundle" extension to `glide.attachment.extensions` properties.
- If the MID Servers must go through a proxy to access the remote git repository, do as follows:
  - Add the following line to `agent\conf\wrapper-override.conf` file:
  - `wrapper.java.additionial.3=-Dhttps.proxyHost=<proxyHost>`
  - `wrapper.java.additionial.4=-Dhttps.proxyPort=<proxyPort>`
  - Restart the MID
  - Add a new entry in "MID Security Policy":
  - Name: <proxy host name>
  - Check "Active" only

Denoise source control commits

Skip unloading into XML files of noisy update fields for any metadata record in source control.

Legacy - Production deployment tips

When you develop customizations to applications on the ServiceNow® platform, you deploy them via the application repository to a production instance. This topic examines and provides cautions for the tradeoffs between installing an application from the application repository versus Git repository with source control.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Overview

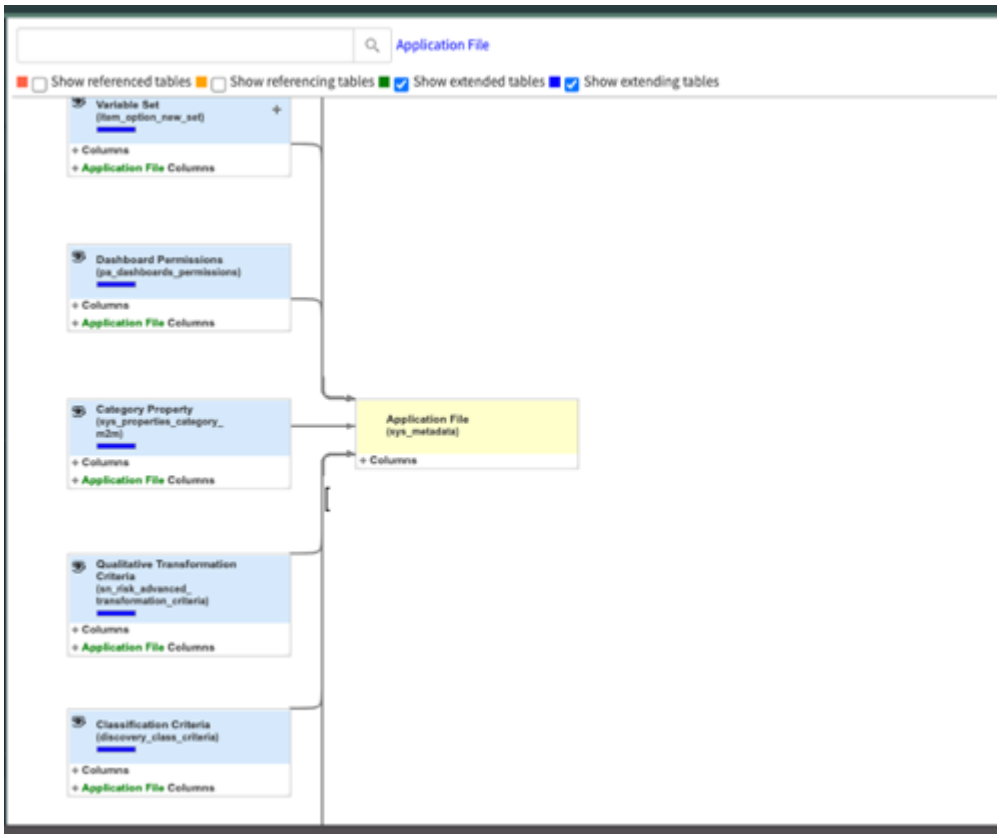
Technically, you can still "deploy" an application from a Git repository to a production instance using source control. This can have unintended consequences.

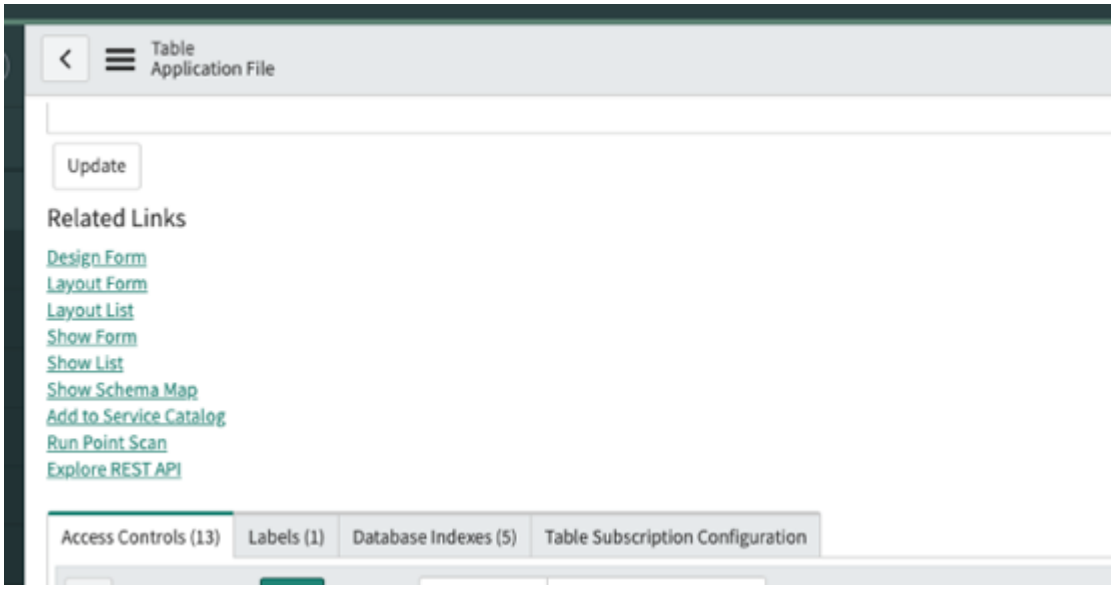
Glossary of terms

Term	Definition
Metadata or application files	The sys_metadata records that define configuration in ServiceNow and are packaged in an application. These records alter the behavior of the instance but do not contain data such as incident or CMDB records. (See Note below)
Scoped applications	ServiceNow applications that restrict allowing only updates and operations within the boundary of the scope. This mechanism is used for most new development.
Global applications	Global applications are developed in the legacy global scope. Work is often done in this scope to customize existing ServiceNow applications such as IT Service Management (ITSM).
Application repository	Applications are typically published here for deployment in production instances. Although the application repository has separate entitlement rules, it operates similarly to the ServiceNow Store.
ServiceNow Store	Repository for third-party (vendor) applications as well as ServiceNow published applications. Most customers do not publish to the Store, but often install applications from it.
Update sets	Standard method of packaging customizations for deployment in each successive instance. They contain the incremental collection of insertions, updates, and deletions.

Term	Definition
Legacy - Delta loading	The most efficient method of loading because it changes only from source control rather than earlier uninstall/reinstall methods.
Schema	Definition of tables and columns in the tables.
Rollback	Administrators can roll back the last installation of a selected application. A rollback removes all code, table, and file updates from the initial installation.

Note:
The **sys_metadata** table is the parent table of all application files in the ServiceNow platform using the table inheritance model. You can view summary information for metadata by visiting the parent table or tables that extend directly or indirectly as indicated by the Extends table(super_class) field on the Table(sys_db_object record). You can also see the whole schema by visiting the Table(sys_db_object) form for the **sys_metadata** table and selecting the **Show Schema Map** related link at the bottom of the form. The schema is large and so takes some time to render.





Installation location

When you install source control, it facilitates the ongoing development of a custom application. Therefore, the application is managed as an "In development" application in the Custom Application [sys_app] table rather than as an "Installed" application in the Store Application [sys_store_app] table. Both tables are extensions of sys_scope so they both provide the same protections and restrictions as the scope. So when you search for the installation of a source control deployed application, refer to the System Application [sys_app] table and the **in development** section of the Application Manager page.

You cannot have a sys_app record on the instance while deploying that same application from the ServiceNow Store or application repository. The two deployment models are mutually exclusive. If at any point the deployment model changes, the sys_app record must be converted to a sys_store_app record first. You can contact ServiceNow Support for help with performing that operation.

Delta loading

Prior to the ServiceNow Paris release, application installation from source control always removed and reinstalled the entire application when triggered, including the Apply Remote Changes function. With Legacy - Delta loading, now only the changes update, simplifying the process considerably.

The Delta loading process loads the changes from source control incrementally. When you apply remote changes, you don't drop existing tables or columns unless they were removed from the repository. This preserves the data for tables and fields that continued to be present.

Note:
The **glide.source_control.allow_delta_loading_in_scopedapp** property allows you to disable Delta loading in Paris; however, this will revert to the more destructive behavior of removing and reinstalling the application. Global applications in Paris always use Delta loading.

Below is a table of the different expected outcomes in an instance using Delta loading.

Application type	Install source	Schema present in package	Schema contains data	Claim by another app (Global)	Expected outcome for data and schema
Scoped	Application repository or Store	Yes	Yes/No	N/A	Preserved
Scoped	Application repository or Store	No	Yes/No	N/A	Preserved
Scoped	Source control	Yes	Yes/No	N/A	Preserved
Scoped	Source control	No	Yes/No	N/A	Removed
Global	Source control, Application repository, or Store	Yes	Yes/No	Yes/No	Preserved
		No	Yes	No	Preserved (1)
		No	No	Yes	Preserved (2)
	Source control	No	No	No	Removed (3)
	Application repository	No	No	No	Preserved

Note:

- While the database schema and the data are preserved, they will be moved into the default global application.
- While the database schema and the data are preserved, they will be moved into the global application which previously held claim to these files if installed. Otherwise, they will move into default global application.
- Applicable only on custom columns with **u_ prefix**. ServiceNow platform-authored columns are not dropped.

When switching branches in source control for a scoped application, use extreme caution in a production environment. If the target branch is missing schema elements found in the current branch, the related schema is dropped, destroying any data it contains. (Global applications do not drop schema when data is present.)

Just as with update sets, only a subset of the incremental changes needs to be applied with Delta loading. Unlike update sets, the application package represents the complete application. Files that are absent from the new package are deleted. This can alter functionality and delete data. Update sets and applications upgraded from the application repository or ServiceNow Store must have an explicit **DELETE** payload to remove a file or drop a schema.

If application files are being generated dynamically in any fashion, the next install/apply remote changes the operation of an application deletes those records. They are considered absent from the incoming application package. If you stash local changes, the application files may be recoverable by a stash commit, but if data is lost as a result of the changes, the data is not recovered.

Note:
Removing or suppressing sys_update_xml records prevents them from being removed by Delta loading; however, this action can have other severe or undesirable results.

Direct edits to the repository, especially to remove files, can have significant ramifications, including loss of data and cascading deletes. Perform this action with care.

Legacy - Author elective and customer updates

When you install an application from the application repository or ServiceNow® store, you can set a series of properties to define the behavior of delete and choice processing. These kinds of choices are called an "author-elective" feature.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Overview

You can find details on these properties in the Skipped records that occur during application installation topic. With these properties, you can opt in and out of deletes and choice updates depending on whether you are using your own or a third-party application.

When you install from source control, however, these records do not skip, except when a global application file is claimed by a different global application. Other than **com.glide.apps.include_my_deletes** and **com.glide.apps.include_global_deletes** that disable the processing of the author_elective_update folder altogether, those properties are not effective for source installed apps.

Note:
An “absent” file detected in Delta loading for source control is vastly different than a Delete payload housed inside the author_elective_update folder. Author_elective_update properties do not prevent Delta loading in source control from deleting the file.

Similarly, update sets protect customizations that you make in an instance against incoming changes that force a preview decision. Before you commit an update set, a preview must be run to attempt to detect collisions. You must address all preview problems before committing the changes. Source control may ask you to stash a local change, but the outcome of the installation is to load what is present in the source even if a change had been made locally.

Loading what is present in the source is challenging when properties must have different versions based on the target of the installation. For example, it’s difficult to resolve when a property containing an integration URL differs based on instance production role. The **is_private** flag is effective with a source control installation and does not overwrite the property if set, mitigating this concern.

Legacy - Roll back, back out, and uninstall

The application installation from the application repository is recorded for rollback, which means as an administrator, you can roll back the last installation of a selected application. When you roll back an application, you remove all code, table, and file updates from the initial installation.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Source control installations are not recorded for rollback, which means that the rollback feature to undo an application repository, store installation, or an in-family upgrade is not available for source control installations. They also cannot be uninstalled in the same sense that ServiceNow® Store applications can be (that is, with the option to retain tables and columns). They can be deleted by deleting the **sys_app** record. However, for scoped applications, this destroys the underlying schema and its data.

You cannot delete global applications until all application files are moved to another global application. Deleting a **sys_app** record for an application also creates **sys_update_xml** records with **DELETE** payloads. These payloads can generate skips if the same application is then installed via the application repository or ServiceNow Store.

To avoid these skips, the **sys_update_xml** records must be manually deleted before the application is installed from the application repository. Installing from source control also does not have the Back out option that update sets contain to remove a subset of new changes. That means that recoverability in this model is limited to installing a corrected version (older or newer) of the application on top of the current version. This action restores only metadata or configuration. Data lost as a result of a poor installation must be recovered from a database restore.

Legacy - Upgrade history

Upgrade history records and history line dispositions are generated as part of the source control installation providing a record of the install.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

The source control installation is more aggressive and generally doesn't skip unless there is a global claim or **is_private protection** on a property. Global applications also protect columns containing data, or those from the ServiceNow base installation, from being removed.

Legacy - Development considerations

When you are developing code, consider some of these suggestions for the most efficient performance.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Production instances shouldn't publish to the Git repository

To protect the integrity of production, you shouldn't push changes to publish to the Git repository from production even though technically you can open and commit changes in the application in ServiceNow® Studio. The "Can Edit Application in Studio" option can be disabled in production on the **sys_app** record. But it resets at the next source code operation that triggers an update (that is, when you apply remote changes or switch branches).

Dependencies are not installed when installing an application from the Git repository

Source-controlled applications do not automatically install or upgrade if they are listed as the dependency of your application. Instead, the customer must install or upgrade the source-controlled application on its own. The customer is also responsible for installing and upgrading the applications in the correct order.

Tight control on source control privileges in production

The source code operations are generally available to those with development privileges in the instance. Since production environments shouldn't have ongoing development, be sure to tightly control admin and delegated development privileges to avoid data loss and other serious consequences.

Legacy - Available source control operations

The source control integration primarily supports operations from Studio, but can also support some operations directly from the GIT repository.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Available source control operations

Operation	Description	Available from
Import from Source Control	Imports an application from the repository to the local instance.	Studio
Link to Source Control	Allows developers to manage application changes from a GIT repository.	Studio
Edit Repository Configuration	Updates the GIT repository user credentials.	Studio
Apply Remote Changes	Updates the local version of the application to match the repository version.	Studio

Available source control operations (continued)

Operation	Description	Available from
Commit Changes	Updates the repository version of the application to match the local version.	Studio
Stash Local Changes	Removes and saves local changes for later work.	Studio
Switch Branch	Updates the local version of the application to match the repository branch version.	Studio
Create Branch	Creates a branch in the repository to save a different version of the application.	• Studio • GIT repository
Create Tag	Creates a tag in the repository to link to a particular application version.	• Studio • GIT repository
Manage Stashes	Allows developers to apply or delete stashed changes.	Studio
Create repository	Creates a repository to store application changes	GIT repository
Create credentials	Creates credentials to the repository.	GIT repository
Grant access to repository	Provides read and write access to the repository tied to a specific set of credentials.	GIT repository
Source Control repository sync	Allows admins to apply remote changes from the Source Control repository, to resynchronize with the instance. **Note:** This operation is only available if Delta loading is being used for the repository. See Legacy - Delta loading.	Repository Configuration form

Legacy - Import application or application-customization from source control

Import an application or application-customization from a source control repository to continue developing it on this instance.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Role required: admin or source_control
  - Restrict permissions on the access token to allow read and write access to the Git repository.
  - The repository user credentials must grant read and write access.

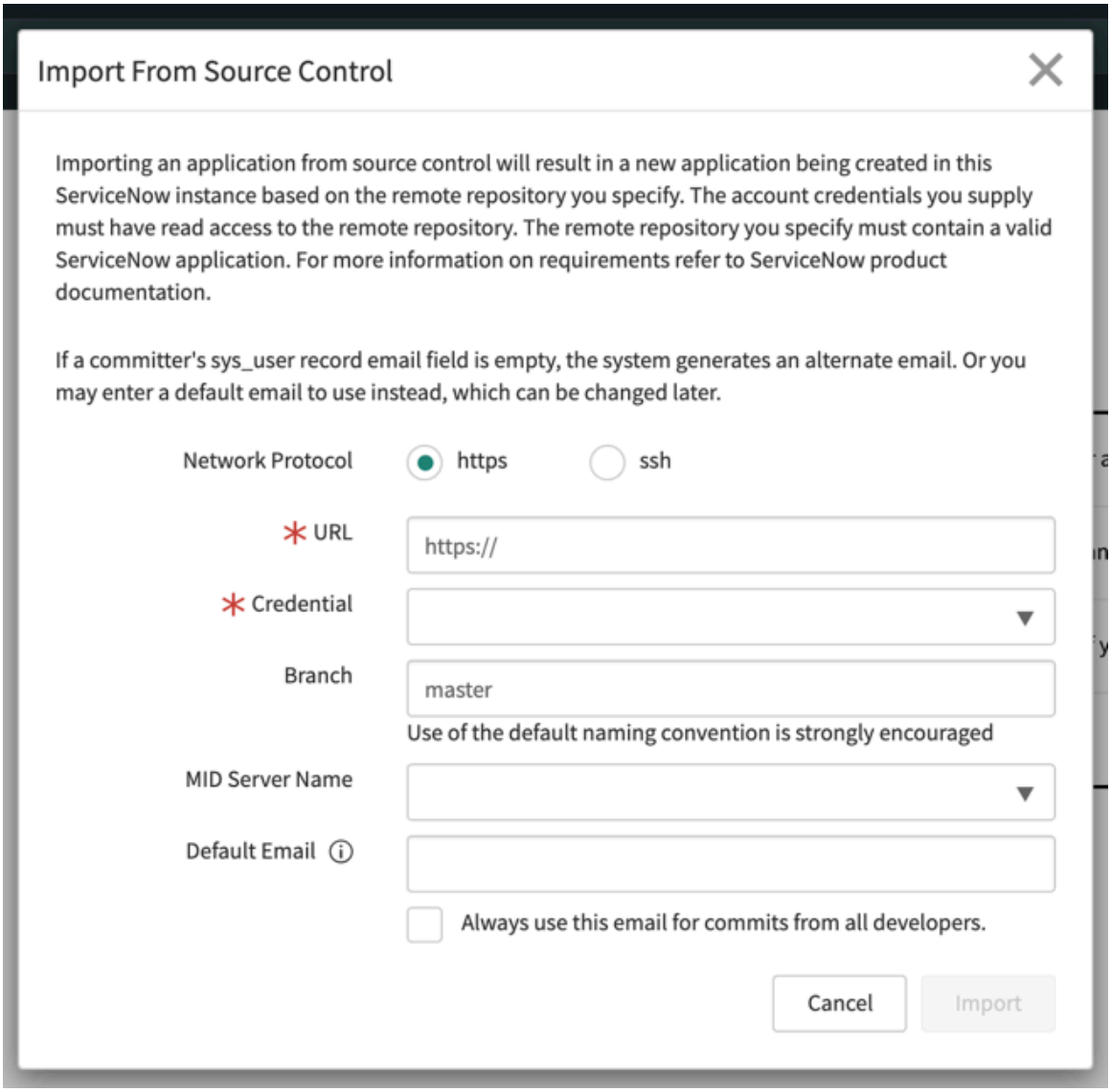
  For more information, see Link an application or application-customization to source control.
- Verify that the non-production instance has network access to the Git repository.
- Verify that the repository contains a valid application.
- Ensure that users add the email address to their respective Users Table (ServiceNow sys_user) record that they use in their commits to the Git repository.
- Learn more about application-customizations Managing application-customizations.

About this task

The source control integration does not support importing an application on a production instance. Instead install applications on a production instance from the application repository, an update set, or the ServiceNow Store.

Procedure

1. Navigate to **All** > **System Applications** > **Studio**.
   The system displays the Welcome to Studio page.
2. Click **Open Studio** > **Go**.
   The system opens Studio and the Switch Applications window.
3. Click **Import from Source Control**.
   Studio displays the Import from Source Control fields.



4. Enter the following field values.

Import from source control fields

Field	Description
Network protocol	Https or ssh credential type that enables secure channel data exchange.
URL	The URL to the Git repository where the application files reside. **Note:** If the Git repo URL for SSH provided by your Git server does not work, check with your Git server owner or provider for the correct URL. There may be additional specifications such as scheme protocol prefixes, port numbers, and so on, required for your Git repo URL to function.
Credential	Select the credential for your Git repository. (See Getting started with Credentials .)

Field	Description
	Note: If you select the ssh network protocol, enter a valid credential of the SSH Private Key type. If you select the https protocol, enter a valid credential of the Basic Auth Credentials type.
Branch	The repository branch to work on within the application. **Note:** The default branch is named after your instance. If you do not choose a name, the branch defaults to **master**.
MID Server Name	Select an existing MID Server to link to a Git repository stored behind your corporate firewall. **Note:** Use a separate MID Server to prevent conflicts with Discovery activities.
Default email	The committer email address is defined by the sys_user record if available. But if a committer's sys_user record email field is empty, the system generates an alternate email (username@instancename.service-now.com). You can also enter a default email address and change it later. To use that default email address in all cases, select the check box.

Note:
All application developers on the instance share the credential used to link a Git repository to an application.

5. Click **Import**.
   The system compares the checksum in the `checksum.txt` file to current checksum. When the checksum values match, the integration skips validation and imports the application. When the checksum values do not match, the integration first validates and sanitizes the application files before importing them.
6. Click **Select Application**.
   Studio displays the application as a new choice in the Switch Applications window.

What to do next

- Review the upgrade logs for any sanitization applied to application files during the import.
- Select the imported application to edit it.

Related topics

MID Server

Getting started with credentials

Legacy - Link an application or application-customization to source control

Linking an application or application-customization to source control allows application developers to manage changes from a Git repository.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Role required: admin
- Learn more about Managing application-customizations.
- Create a dedicated Git repository for the application. For increased security, enable multi-factor authentication for the Git repository.
- Generate an access token that the source control integration can use instead of a password and multi-factor authentication passkey. Search for personal access token on GitHub or GitLab.
- Restrict permissions on the access token to allow read and write access to the Git repository.
- Verify that the non-production instance has network access to the Git repository.
- Ensure that users add the email address to their respective Users Table (ServiceNow sys_user) records that they use in their commits to the Git repository.
- Learn more: Legacy - Migrate completed update set history to Source Control

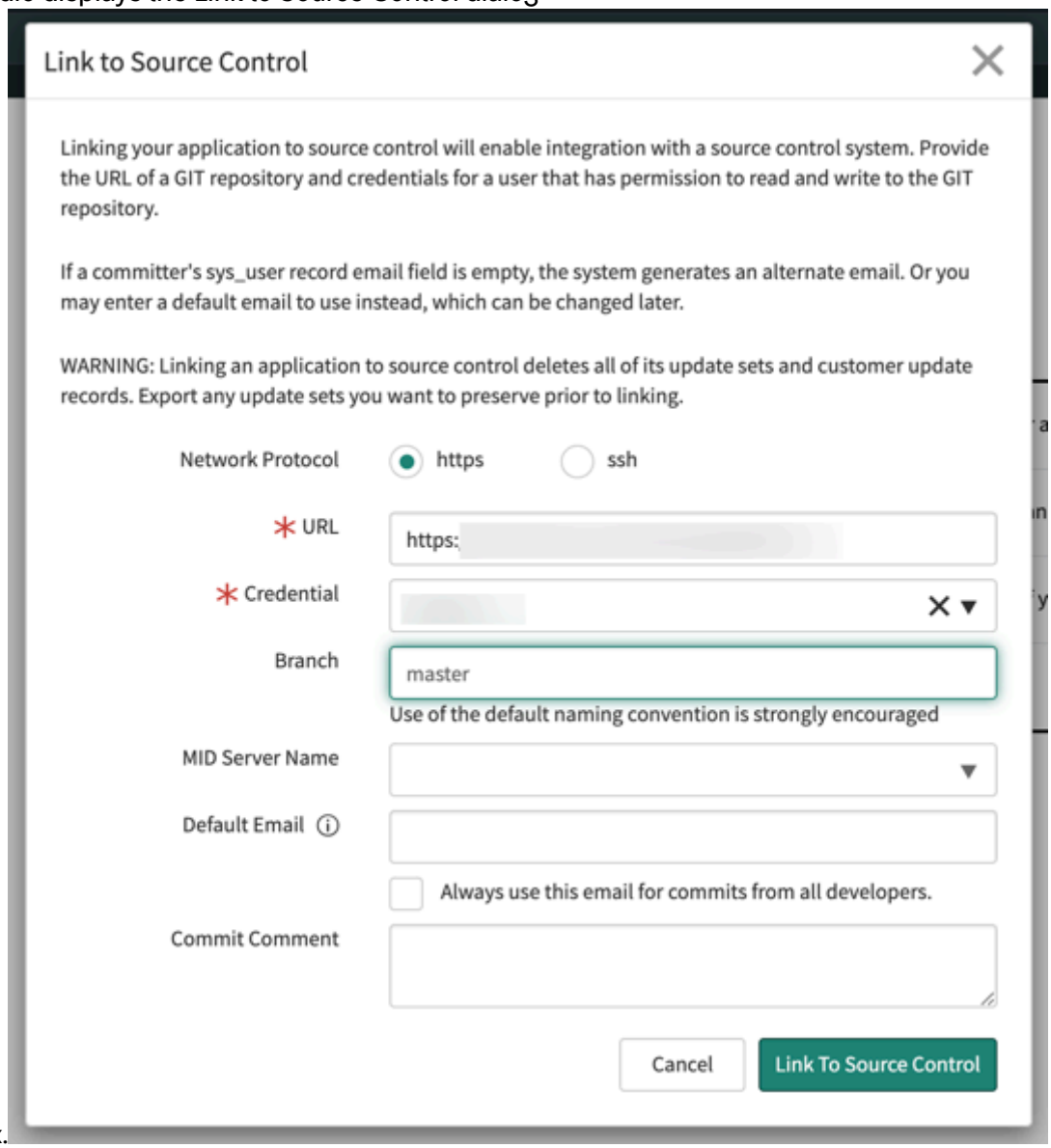
About this task

The source control integration does not support linking to an application or customization on a production instance. Instead, install applications on a production instance from the application repository, an update set, or the ServiceNow Store.

Procedure

1. Open the application you want to link to source control in Studio.
2. Navigate to **Source Control** > **Link to Source Control**.

Studio displays the Link to Source Control dialog box.



3. Enter the connection details for the Git repository.

Source control connection details

Field	Description
Network protocol	HTTPS or SSH credential type that enables secure channel data exchange.
URL	The URL to the Git repository where you want to save application files. For SSH protocol, use command to generate private key `ssh-keygen -t rsa -m PEM -b 4096 -C "email@address"`. **Note:** If the Git repo URL for SSH provided by your Git server does not work, check with your Git server owner or provider for the correct URL. There may be additional specifications such as scheme protocol prefixes, port numbers, and so on, required for your Git repo URL to function.

Field	Description
Credential	The credential to be used with the selected protocol. See Getting started with Credentials to learn more about creating credentials. **Note:** If you select the SSH network protocol, enter a valid credential of the SSH private key type. If you select the https protocol, enter a valid credential of the Basic Auth credentials type.
Branch	The repository branch to work on within the application.
MID Server Name	The name of the existing MID Server to link through. **Note:** Use a separate MID Server to prevent conflicts with Discovery activities. Be sure that the MID server user can create files to the sys_attachment table and that the table can accept files of the "bundle" type. Linking or importing an application through a MID Server enables access to repositories behind a firewall. See the Using MID Server with source control and MID Server topics to learn more
Default email	The committer email address is defined by the sys_user record if available. But if a committer's sys_user record email field is empty, the system generates an alternate email (username@instancename.service-now.com). You can also enter a default email address and change it later. To use that default email address in all cases, select the check box.
Commit Comment	An optional description of the repository or application.

Note:
All application developers on the instance share a single set of repository credentials.

4. Click **Submit**.
   The system validates the connection and user credentials and displays a success message.

   All application developers on the instance can use the linked Git repository to manage changes.

Related topics

Getting started with credentials

Legacy - Using MID Server with source control

The ServiceNow® MID Server enables communication and the movement of data between a ServiceNow instance and external applications, data sources, and services.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

How bundle files work with MID Server

The **.bundle** file helps source control function with a MID Server. A bundle file is the way Git packages a local repository in a single file. This makes sharing or moving the repository simpler and more streamlined. The file is then sent to the MID Server, which passes it on to the remote repository.

The `outgoing.bundle` (commit operations) and `incoming.bundle` (apply remote changes) are attached to the MID Server attachment table [ecc_agent_attachment] for any request that goes to the MID Server. The `outgoing.bundle` is created on the instance while the `incoming.bundle` is created on the MID Server.

After an operation completes successfully, the bundle file is "promoted" into a `golden.bundle` that is attached to the Repository configuration table [sys_repo_config]. It's used to initialize the repository on a node that has not performed any Source Control operations yet.

The Auto Flush tool [sys_auto_flush] is a "table cleaner" that removes any `ecc_agent_attachment` record older than 30 days. This action removes the corresponding attachment as well.

The bundle files are kept on the MID Server and then saved to the Import directory on the MID Server.

On the MID Server, the bundle file is saved in the Import folder. This folder is under the user directory defined by the system property (user.dir), which users can configure. The bundle file is removed as part of the system flushing at the end of every operation.

Working with the MID Server

Note:
Source control operations can take more time for larger applications when using an MID server, as the entire app is bundled after an export and is dependant on the size of the app.

- Avoid conflicts with Discovery and create files for the system attachment [sys_attachment] table: MID Server
- Learn about system properties restrictions: Configure attachment system properties

Legacy - Migrate completed update set history to Source Control

When linking to Source Control, this feature allows application developers the choice of migrating the information in completed update sets to Source Control history.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Before migrating

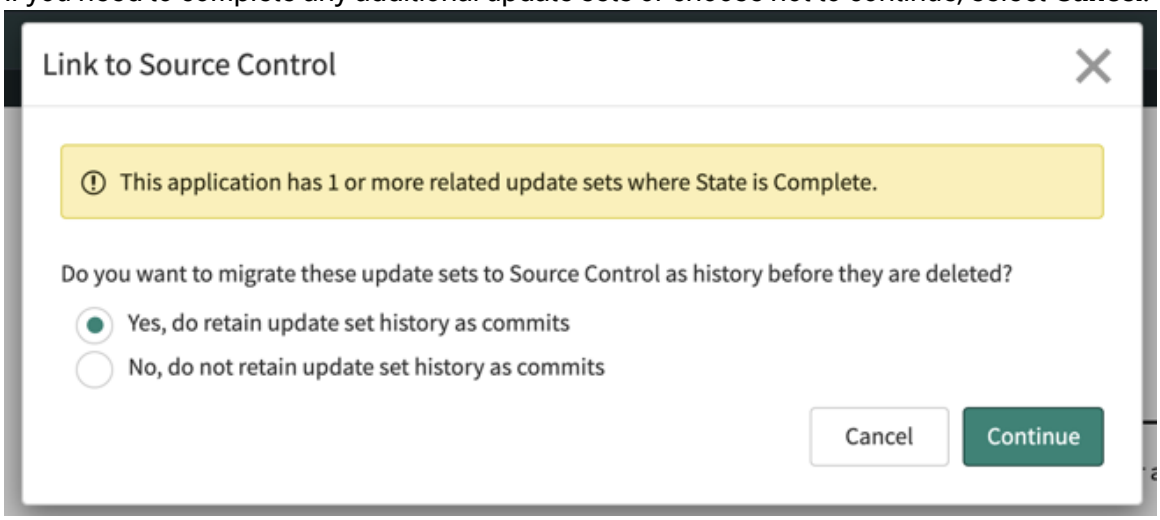
Make sure that you have fulfilled these criteria before attempting to migrate your update sets:

- Role required: admin
- Read the Legacy - Link an application or application-customization to source control topic
- Complete any update sets for your application that you want to export as Source Control history.
- Export the completed update set if you want to preserve it.

When you link an application to Source Control, the update sets and customer update records are deleted. After you link to Source Control, if the application has any completed update sets, you will be asked to make a choice in the dialog box below.

- If you select "Yes, do retain update set history as commits", the update set history is preserved as Source Control commits.
- If you select "No, do not retain update set history as commits," they are not preserved as commits.

Regardless of which option you select, if you select **Continue**, the **Link to Source Control** operation starts, and all completed update sets and all Customer Update records are deleted. If you need to complete any additional update sets or choose not to continue, select **Cancel**.



For every completed update set with updates to the application that you are linking to Source Control, commits are generated automatically by the system based on the sys_update_xml records in the update sets. The commits are ordered by the **sys_recorded_at** timestamp. For Global applications: Any **sys_update_xml** records that belong to the application and are part of a completed Global update set are captured as historical commits.

When the Link to Source Control operation is complete, the most recent commit is the current state of your application in its entirety. You can view historical commits in your Git repository or by clicking the Source Control menu option and selecting **View History**. Updates are separated into multiple commits:

- If there are updates for a file that are out of order between different update sets.
- If an update set contains multiple update records for a single file.

The commits for an update set are split into multiple commits ([Historical Commit 1], [Historical Commit 2]...) to represent each update. This is done so that each file has an ordered history of updates.

Warning:
Any commit prefixed by [Historical Commit] is generated solely to display its history. Do not attempt to check out these commits in the development process as they do not necessarily represent a stable snapshot of the application.

The **author_elective_update** folder is not created until the initial commit. That means that in the initial commit you might see files such as **sys_choice** files being renamed and moved from the update folder to the **author_elective_update** folder. Any files that are deleted from update sets in historical commits are deleted, and not moved to the **author_elective_update** folder as they would be for actual commits. During the initial commit, DELETE payloads are also created for any DELETE sys_update_xml records that were deleted as part of completed update sets.

Example commit message:

```
[Historical Commit 1] <Name of update set that this commit
 belongs to>
Description: <Description of update set that this commit belongs
 to>
Update Set was completed on / installed on <date>
Update Set was completed by <sys_user user_name > <sys_user
 email>
{
```

Additional values from sys_update_set record (see **Customization**section below)

```
}
{
```

Batch update set information (See the **Batch update sets** section below) }

Batch update sets

If an update set is part of a batch update set, that information is appended to the commit message in the following format, with the highest number being the Batch Base:

```
{
"1": {
"parent": "<name of parent update set>",
"description": "<description of parent update set>"
},
"2": {
"parent": " <name of parent 1's parent update set> ",
"description": " <description of parent 1's parent update set> "
}
}
```

Customization

You can add additional fields to include in the commit message by adding a **glide.source_control.historical_commit_fields** property. The value is a comma-separated list of fields the user wants to include from sys_update_set XML fields. Spaces and invalid or misspelled field names are ignored. This property is used for all applications that are linked to Source Control from the instance if the committer chooses to retain update set history.

Note:
If the value of a field references another table or sys_id, only the value of the field is added. For example: sys_id for a user instead of the name of the user.

XML example

```
▼<xml>
  ▼<sys_update_set>
      <application
      display_value="test_app2">a388a8735be720107c7899b8b681c7d8</application>
      <base_update_set/>
      <completed_by display_value="System
      Administrator">6816f79cc0a8016401c5a33be04be441</completed_by>
      <completed_on>2021-04-22 22:14:38</completed_on>
      <description>added a script include</description>
      <install_date/>
      <installed_from/>
      <is_default>false</is_default>
      <merged_to/>
      <name>update set 1</name>
      <origin_sys_id/>
      <parent/>
      <release_date/>
      <remote_sys_id/>
      <state>complete</state>
      <sys_created_by>admin</sys_created_by>
      <sys_created_on>2021-04-22 22:13:45</sys_created_on>
      <sys_id>6961f0f75be720107c7899b8b681c71f</sys_id>
      <sys_mod_count>1</sys_mod_count>
      <sys_updated_by>admin</sys_updated_by>
      <sys_updated_on>2021-04-22 22:14:38</sys_updated_on>
    </sys_update_set>
  </xml>
```

Value of the property

Result in commit message

[Historical Commit 1] update set 1

```
Description: added a script include
Update Set was completed on 2021-04-22 18:56:06
Update Set was completed by admin admin@example.com
{
  "sys_id": "5d278db35b2320107c7899b8b681c73d",
  "is_default": "false",
  "sys_created_by": "admin"
}
```

Legacy - Edit a Git repository configuration

You can edit a Git repository to change the network protocol selection, credentials or other field entries.

Before you begin

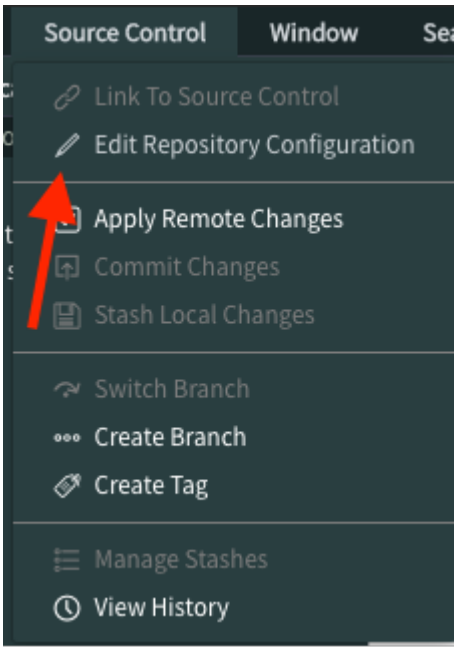
Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

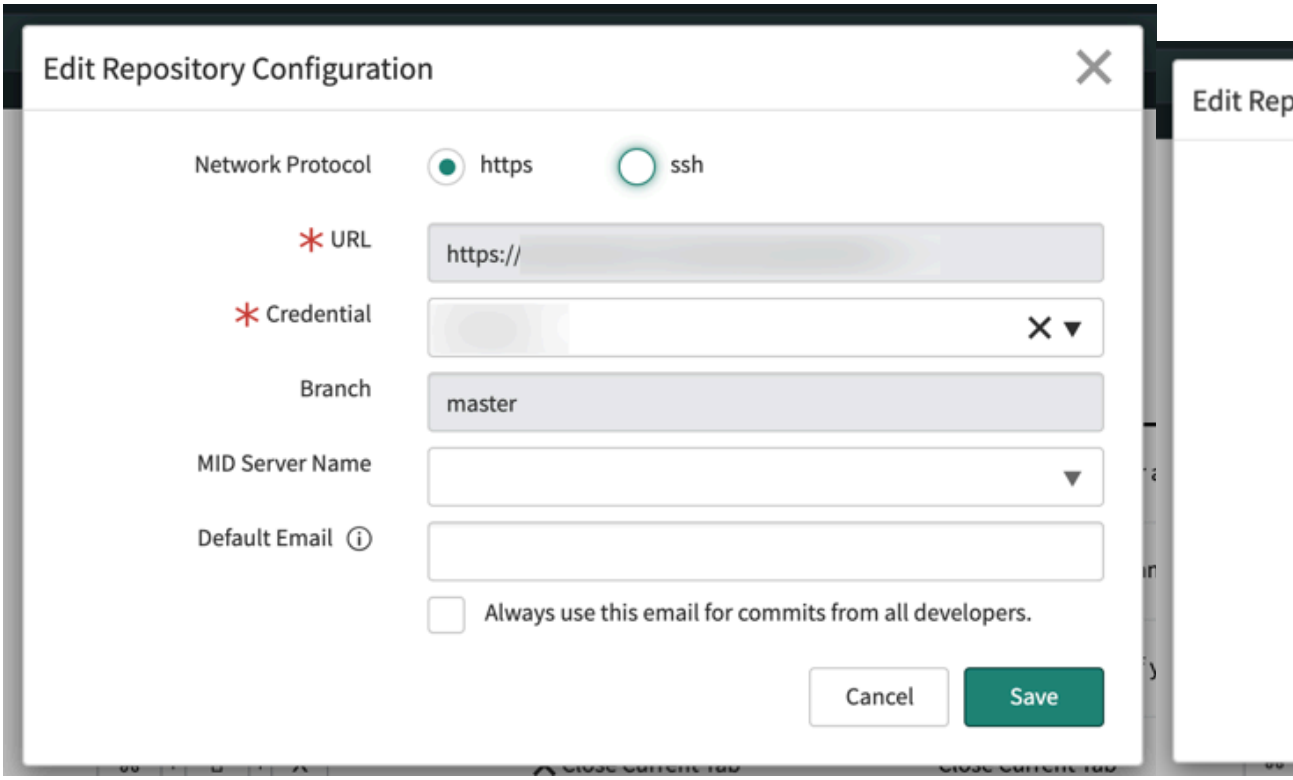
Role required: admin

Procedure

1. In Studio, select **Edit Repository Configuration**.



2. Choose **https** or **ssh** Network protocol and enter the URL address of your repository.



3. Change your credential or MID Server name if you wish.

   Note:
   If you have no MID server name, you can select a new one from the drop-down list. If you choose a new MID server, Legacy - Apply remote changes in the Source Control menu before making any further Source Control operations to avoid errors.

   For the default email field, the committer email address is defined by the sys_user record if available. But if a committer's sys_user record email field is empty, the system generates an alternate email (username@instancename.service-now.com). You can also enter a default email address and change it later. To use that default email address in all cases, select the check box.

4. Click **Save**.

Legacy - Apply remote changes

Application developers can pull changes from a linked GIT repository to apply remote changes to the local instance.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Role required: admin
- Legacy - Link an application or application-customization to source control

Procedure

Navigate to **All** > **Source Control** > **Apply Remote Changes**.
The following operations occur:

- The system fetches the most recent changes from the remote repository.
- The system applies the remote changes to the instance.
- The system identifies any change conflicts requiring resolution.

If there are conflicts, the system displays the **Resolve Conflicts** window.

Delta loading is enabled by default in sys properties so your data isn't removed. You can disable this feature if you want data automatically deleted.

Note:
After a Source Control **Apply Remote Changes** or other operation, you may encounter the error message: `Skipped loading file *.xml, cannot parse the xml document, found git merge conflicts tokens`. Make sure that the files in your application do not include values that the Git repository interprets as merge conflict tokens. While these aren't actual conflicts, you should remove them anyway. Examples of these values are:

```
>>>>>>>
```

```
=======
```

```
<<<<<<<
```

What to do next
Resolve any change conflicts.

Legacy - Commit changes

Application developers can commit their changes on the instance to the linked Git repository. You can either select a few changes to commit, or commit all changes on the instance at once.

Before you begin

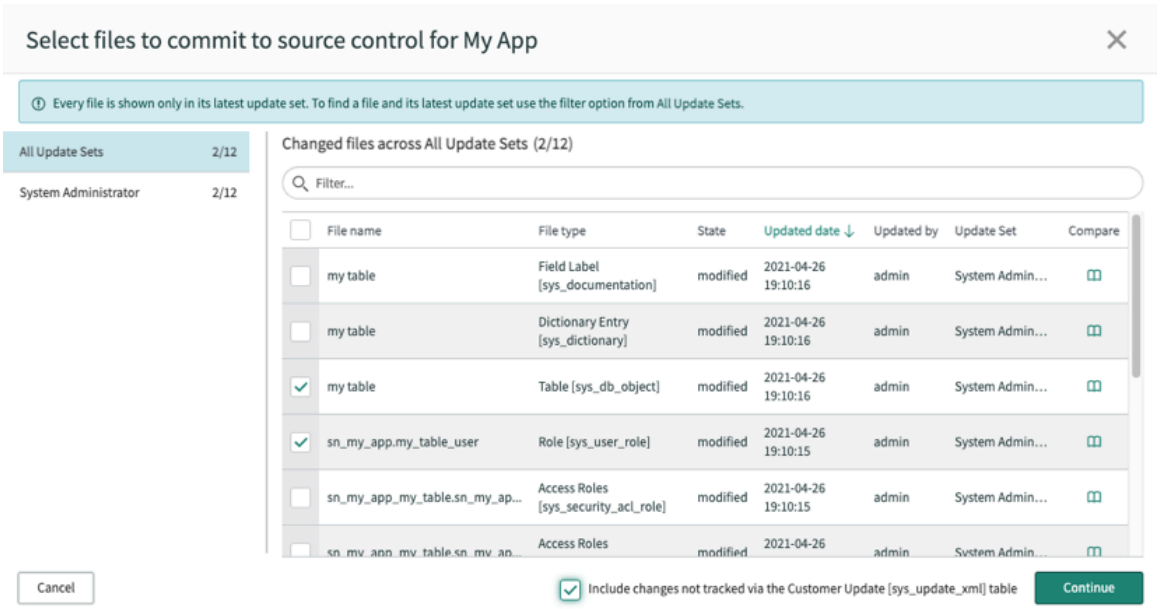
Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Role required: admin
- Legacy - Link an application or application-customization to source control

Procedure

1. Navigate to **All** > **Source Control** > **Commit Changes**.
   The system displays the **Select files to commit to source control** window. The file changes from all the updates sets display. By default, the file changes from the current update set display.



2. Select the file changes you wish to commit.
3. To include untracked changes, select the **Include changes not tracked via the Customer Update [sys_update_xml] table** check box.
   - The default for this check box is set via the **glide.sourcecontrol.default_commit_mode** property.
     - Property can be set to **include_untracked** or **exclude_untracked**.
     - The **include_untracked** mode commits the updates to the application that do not generate sys_update_xml records, as well as any user-selected updates.
     - The **exclude_untracked** mode commits only updates selected by the user in the **Select files to commit to source control** dialog.
   - The base system setting for the property is **exclude_untracked**.
   - Prior to the ServiceNow Rome release, only the **include_untracked** mode is used.

   To hide the check box and use the value of the **glide.sourcecontrol.default_commit_mode** property, create the **sn_devstudio.vcs.allow_commit_mode_selection** property and set it to false. Checking this check box may incur a performance penalty.

Note:

Commits always occur in **include_untracked** mode in the following cases:

- Linking to Source Control for the first time. (To learn more, see Legacy - Link an application or application-customization to source control.)
- Publishing an application that's linked to Source Control from ServiceNow Studio. (To learn more, see Legacy - Publish an application from ServiceNow Studio when linked to Source Control.)
- Selective commit mode is disabled.

4. Click **Continue**.
5. In **Commit comment**, enter a comment for the changes.
6. Click **Commit Files**.
   The following operations occur:
   - The system identifies all local changes.
   - The system commits all local changes to the remote repository.

Note:
For list of known files that don't have customer update records and are untracked, see Customer Updates table.

Legacy - Stash local changes

Application developers can remove and save changes locally to apply them later.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

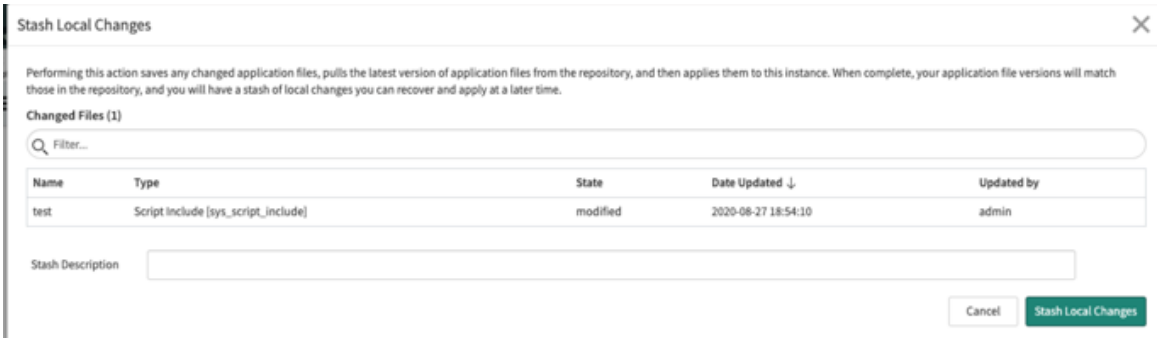
- Role required: admin
- Link an application to source control
- Change one or more application files

About this task

Stashing changes removes them from the current application and saves them for a developer to later apply or delete.

Procedure

1. From Studio, navigate to **Source Control** > **Stash Local Changes**.
   The system displays a list of locally changed files.
2. Enter your description.
3. Click **Stash Local Changes**.
   The system saves the current changes and displays a success message.



Note:
In the San Diego release onward, user information is preserved with each update during stashing. When the stash is applied, each update is restored in the respective user's in-progress local update set.

What to do next

- Close dialog
- Legacy - Manage stashes

To learn more see Getting started with credentials

Legacy - Switch branch

Application developers can switch to a different repository branch to work on another version of the application.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Role required: admin
- GIT repository with one or more available branches.

Procedure

1. Navigate to **All** > **Source Control** > **Switch Branch**.
   The system displays the Switch Branch window.
2. **Optional:** If there any local changes on the instance, you can save or discard them.

   Note:
   Use caution when discarding local changes. Since all application developers share repository credentials, there is no way to discard just one set of user changes. Note you cannot later restore discarded changes.

3. Select the branch you want to switch to.

4. Click **Switch Branch**.
   Studio updates the local application to match the branch version from repository.

Legacy - Create branch

Application developers can create a branch to work on a new version of an existing application.

Before you begin

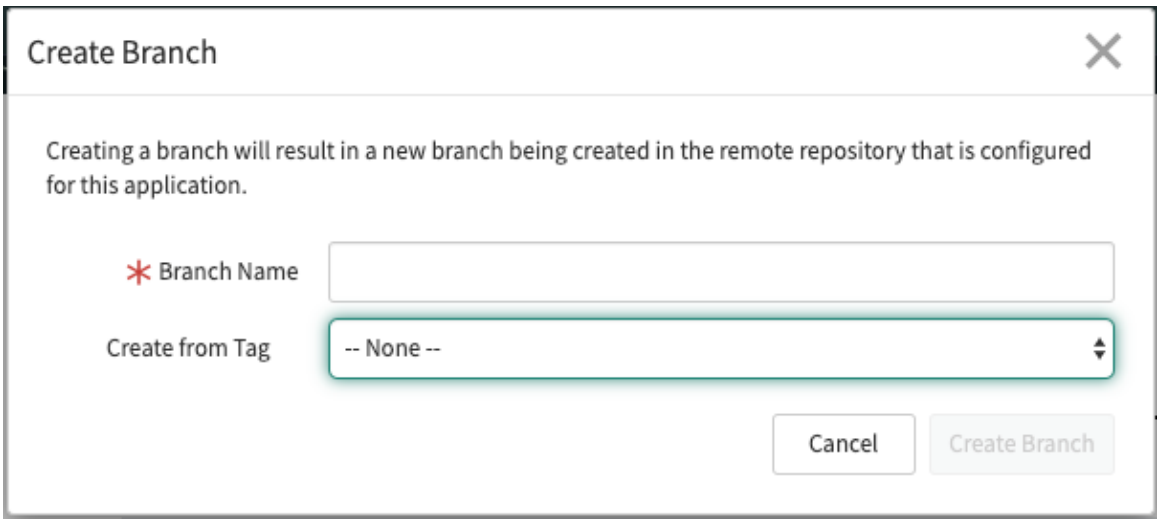
Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

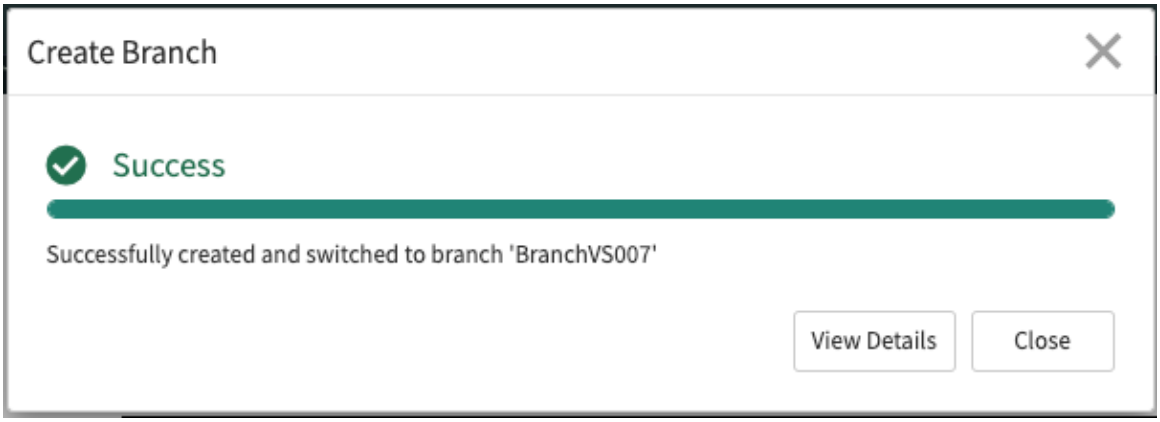
- Role required: admin or sn_group_creator.app_creator
- Legacy - Link an application or application-customization to source control

Procedure

1. Navigate to **All** > **Source Control** > **Create Branch**.
   Studio opens the Create Branch window.



2. Enter the **Branch Name**.
3. **Optional:** To create a branch from a tag, click the **Create from Tag**drop-down list and select an existing tag.
4. Click **Create Branch**.
   Studio creates the branch.



5. Click **Close**.

What to do next
Commit changes to the new branch.

Legacy - Set default branch

Set a default branch when you want to use a branch other than main for new changes or for your main development repository.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Role required: admin
- Legacy - Link an application or application-customization to source control

Procedure

1. Follow the steps to Add a system property .
2. Add the **glide.source_control.default_branch_name** property, and specify the default branch name of the GIT source control repository to work from (pull requests, code commits, etc.).

Result
Application developers' work is managed from and saved into the default branch if not otherwise specified. If not changed, this value defaults to sn_instances/<instance_name>.

Legacy - Manage stashes

Application developers can apply or delete stashed changes from Studio.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Role required: admin
- Link an application to source control.
- Stash one or more application file changes.

Procedure

1. From Studio, navigate to **Source Control** > **Manage Stashes**.
   The system displays a list of locally stashed changes.
2. Click the action next to the stash you want to manage.

Option	Description
Apply	Commits the stashed changes to the application and checks for conflicts.
Delete	Removes the stashed changes.
View	View the list of stashed changes.

Legacy - Resolve conflicts

Application developers can choose which application file version to use when applying remote or stashed changes.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Role required: admin
- Link an application to source control
- Apply a stashed change

About this task

Conflicts occur when there are multiple change versions of the same application file: one set of changes in the remote or stashed version and another set of changes in the local version. Studio requires developers resolve conflicts before applying remote or stashed changes.

Procedure

1. From Studio, apply remote or stash changes.
   If the system identifies a conflict, it displays the Resolve Conflicts dialog.
2. Select how to resolve the conflict.

Option	Description
Select an action	Apply or discard all stashed changes. Go to Step 3.
Manually merge changes	Individually select which changes to apply. Go to Step 6.

3. If you want to apply or discard all stashed changes, select an **Action**.

Option	Description
Take Stashed Changes	Applies the application file version from the stashed changes.
Discard Stashed Changes	Applies the application file version from the most recent pull from the repository.

4. Click **Apply Stashed Changes**.
   The system applies the selected changes.
5. Click **Close Dialog**.
6. If you want to merge the conflicting changes, click **Manually Apply**.
   The system displays a list of version differences by field.
7. Select the field values you want the merged application file to have.
8. Click **Save Merge**.
   The system applies the selected changes.

Legacy - View commit history

Application developers can view the commit history of applications linked to a source control repository.

Before you begin

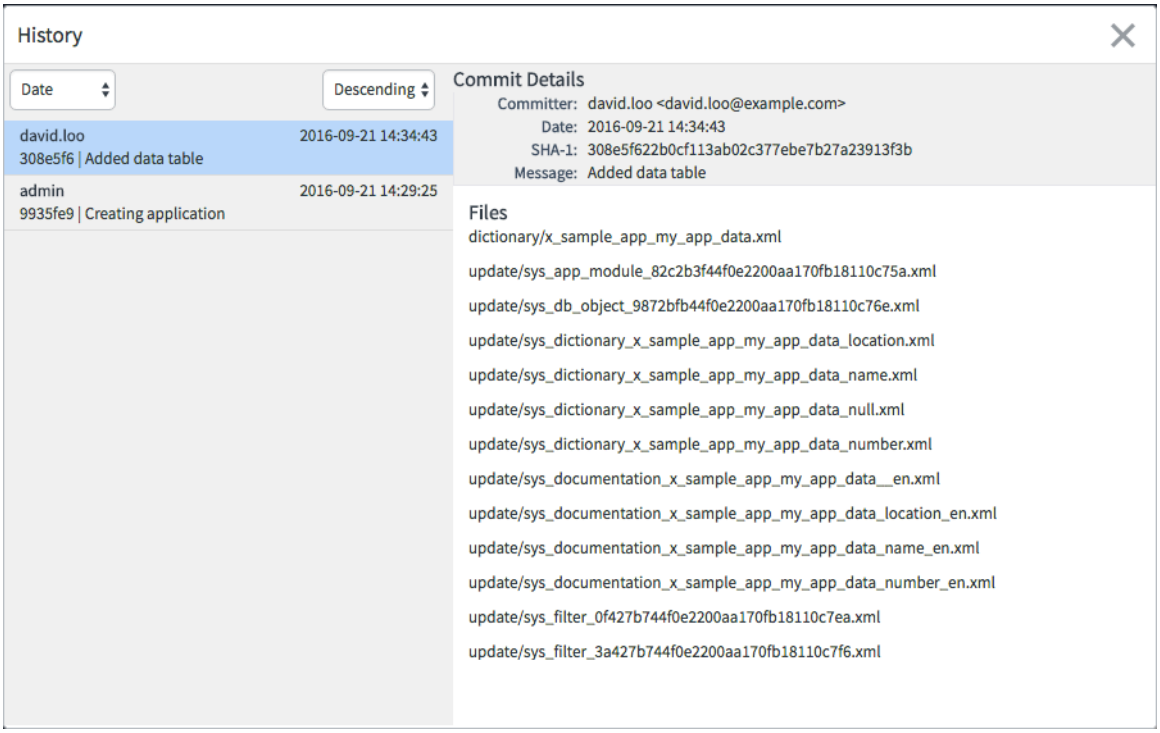
Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Role required: admin
- An existing link to a GIT repository

Procedure

1. Navigate to **All** > **Source Control** > **View History**.
   The system displays the History window.



2. Select the commit sort order type.

Option	Description
Date	Sort by commit date.
Committer	Sort by user name.

3. Select the sort order direction.

Option	Description
Descending	Sort dates from the most recent to oldest date. Sort user names reverse-alphabetically from Z to A.
Ascending	Sort dates from the oldest to most recent date. Sort user names alphabetically from A to Z.

   The system sorts commits by the selected sort order.

4. Select a commit.
   The system displays the commit details for the selected commit.
5. Review the commit details.

Commit Details

Field	Description
Committer	The user who committed the change.
Date	The date-time stamp of the commit.
SHA-1	The secure hash value identifying this commit in the repository.
Message	The commit message associated with this commit.
Files	The list of application files changed in this commit.

6. Close the History window.

Legacy - Move application files in a GIT repository

Move application files linked to source control to any folder of the repository. Allow application developers to store supporting content such as automated tests in the same repository as the applications they support.

Before you begin

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

- Legacy - Link an application or application-customization to source control
- Role required: Source control credentials with write access

About this task

Linking an application to source control generates a properties text file called `sn_source_control.properties` at the root level of the repository. The properties file specifies the folder containing the application files. The integration tracks changes to these application files by generating a `checksum.txt` file. When the checksum matches, the integration skips the validation and sanitization process. When the checksum does not match, the integration validates and sanitizes the application files as part of the source control operation. The integration ignores all repository content outside the application path.

Note:
You can set system properties **glide.source_control.checksum_required** to enable optional checksum validations and sanitizations and **glide.source_control.checksum_quick_install** to bypass sanitization steps on checksum matches. See Available system properties for more information.

Procedure

1. Login to source control repository linked to the application.
2. Create the destination folder where you will move the application files.

Example
For example, create the folders `src/app`.

3. Move the folder containing your application files to the destination folder.

Example
For example, move the folder `demo_my_app` to `src/app`.

4. Navigate to the root level of the repository.
5. Open the `sn_source_control.properties` text file in a text editor.
6. For the `path=` property, enter the folder path where you moved the application files.

Example
For example, enter `path=src/app`.

7. Save the properties file.

What to do next
Login to your instance and perform source control operations from Studio.

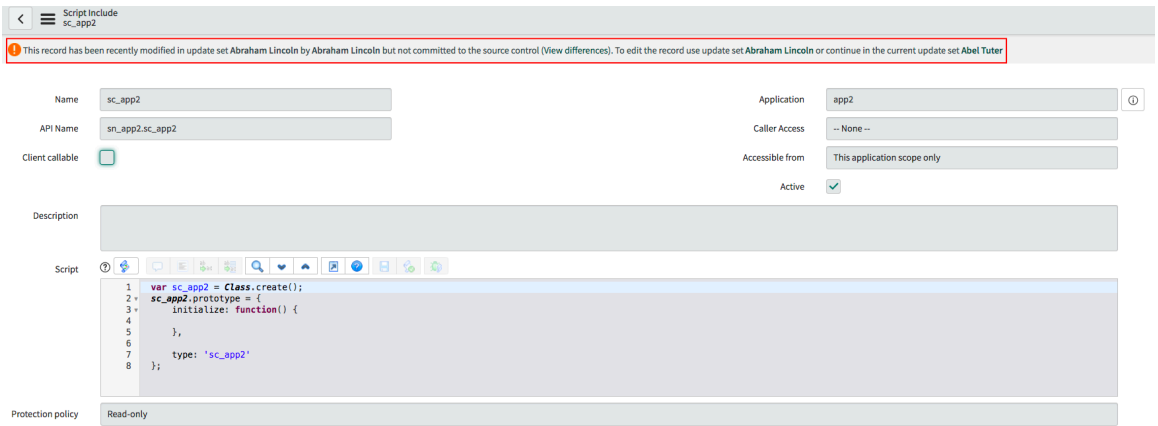
Legacy - Collision avoidance

Avoid modifying an application file across different update sets to ensure seamless experience during the commit process.

> **Important:**
> Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.
>
> Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

When the logged in user opens an application file that was modified in an update set different from the user's current update set for the corresponding application:

- Collision is detected in the application file and a warning message is displayed.
- Logged in user to prompted to choose an update set.
- Read-only protection policy is applied to the application file.



User can make changes to the application file only after selecting the required update set.

Note:
This feature is applicable to only those applications that are linked to GIT.

Enable or disable the collision avoidance feature using the *glide.ui.vcs.collision_avoidance* property in the System Property [sys_properties] table. By default, the feature is enabled. See Available system properties for more information.

When the feature is enabled, users can't work in the default update set of the application. If a user is assigned the default update set, a unique update set is created when the user logs in to the application for the first time after the collision avoidance feature is enabled. This new update set is specific to the user in current application.

Name of the update set specific to the logged in user is, User ID or user name based on the value specified for the *glide.ui.vcs.updateset_identifier* property in the System Property [sys_properties] table. See Available system properties for more information. By default, the update set name is set to user name of the logged in user. However, users can rename the update set.

Legacy - Delta loading

"Delta loading" is an optimized way to load an application from a Git repository.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

When you switch branches and apply remote changes, the Delta loading feature makes sure only changed updates load. Rather than having to perform a full uninstall and reinstall of the application, only the changes update. This speeds and eases the process for developers and does away with the need to use all branches during development. Data stored in tables is retained during these operations, lowering the need to load demo data back into the application after a reinstall or branch change operation.

Benefits of Delta loading

Time savings

Typically, the longest part of an application uninstall or reinstall is dropping and creating tables. Delta loading prevents that from occurring. Depending on the size of the application, not having to drop or create tables can significantly improve an operation's completion time.

Test and demo data retained

Tables that contain test and demo data are no longer deleted, so data is not lost. Developers can save time if they don't need to reimport demo data each time.

Cross-scope dependent references preserved

Items that are cross-scoped and share references are no longer lost when you apply changes. This used to occur when an item in scope B was associated to a parent item in scope A. Applying remote changes on scope A would delete the parent item and reinstall it, breaking the reference between both items, but not restoring that relationship. Delta loading prevents the uninstall, so the reference is not lost.

Recommended practice

Delta loading is enabled by default on all instances starting with the ServiceNow Paris release. The feature is designed to help in application development and receives continued support and upgrades.

Legacy - Denoise your source control commits

As a source control developer, you can merge the Git branches, without getting noise from the fields that are auto-updated by the system.

Important:
Starting with the Xanadu release, the legacy version of ServiceNow Studio is being prepared for future deprecation. It will be hidden and no longer activated on new instances but will continue to be supported. For details on the deprecation process, see the Deprecation Process [KB0867184] article in the Now Support Knowledge Base.

Try building and editing apps in the current version of ServiceNow Studio instead. For more information, see Building applications with ServiceNow Studio.

Overview

In Studio, the Source Control feature packages application files as XML payloads, when they are exported to Git repositories. When a user merges Git branches in a Git repository as part of the application development workflow, the user must resolve any conflicts in the XML files. These conflicts are typically in the fields that are system generated, like sys_updated_by, and represent non-user generated changes. The user must be careful when they are resolving conflicts in these fields because this process might create more noise.

Saver Exempt attribute

Features have been added in Source Control to de-noise the XML payloads, to assist the user in resolving the conflicts when the Git branches are merged.

By default, the system sets the saver_exempt attribute for certain fields in tables whose values are auto-generated by the system. For more information on the saver_exempt attribute, see Dictionary attributes.

The following table contains the saver_exempt values.

Saver Exempt attribute table

Fields	System Tables	saver_exempt values	Outcome expected
• sys_updated_on • sys_updated_by • sys_mod_count	• sys_metadata • sys_choice • sys_package • sys_app_customization • sys_claim • sys_package_dependency_m2m	exempt_vcs_only	The fields are not written to the corresponding XML representation of a record for the table during the packaging of the application for Git commits.
Sys_id	• sys_dictionary • sys_documentation • sys_choice	exempt_vcs_only	The sys_id field is not written in the XML representation of the record of three tables during the packaging

Saver Exempt attribute table (continued)

Fields	System Tables	saver_exempt values	Outcome expected
			of the application for Git commits.
Fields with loader_exempt=true		exempt_always	The fields that have loader_exempt set to **True** are not loaded in the instance. They are not written to the XML representation of the records in the tables during the packaging of the application for Git commits

Disabling the tracking property

An admin can disable the `glide.source_control.disable_tracking_of_update_fields` so that the fields do not display user-generated values.

Note:
The following fields are not written to XML during source control commits, and their values in the instance are loaded from a Git commit. If an admin wants the system to write these fields to XML and use the system-generated values for these fields, the admin can set the property `glide.source_control.disable_tracking_of_update_fields` to **False**.

The following fields are the system-generated values that the admin sees:

- sys_updated_by = commit user
- sys_updated_on = commit time
- sys_mod_count = zero

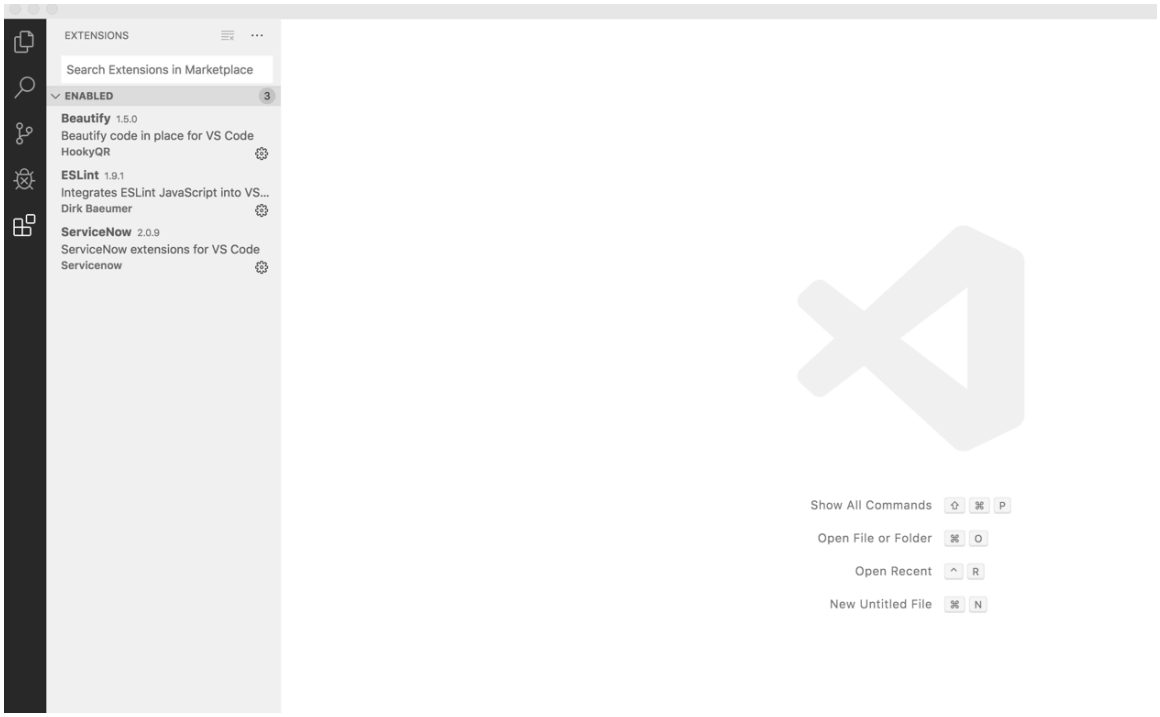
sys_id fields for tables

The sys_id field for tables are saved in the corresponding “dictionary/<tablename>.xml” so that they are not regenerated during the installation in the instance, other than when tables are created. The result is that the noise is reduced from all the XML files that reference the sys_id of the table or represent the sys_db_object record for this table.

ServiceNow Extensions for Visual Studio Code

Edit your ServiceNow applications in Visual Studio Code with the help of the ServiceNow Extensions for VS Code.

The ServiceNow Extensions for the VS Code editor enables you to edit applications within your ServiceNow instance.



Advantages of using Visual Studio Code

Edit your applications offline

Use Visual Studio Code to download and edit a local copy of your application. You can edit the application offline, and then synchronize again when your instance is available.

Use Visual Studio Code JavaScript features to reduce development time and improve code quality.

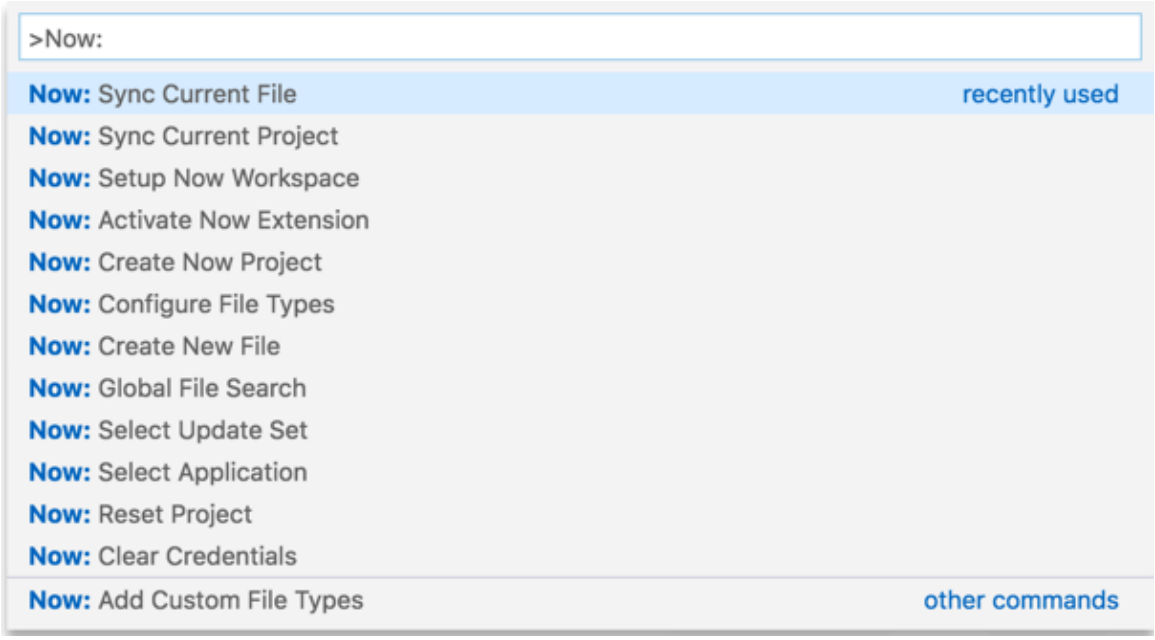
Visual Studio Code IntelliSense includes editing features such as code completion, code suggestions, and quick information. Use these tools to complete your coding tasks quickly and reduce errors. For more details on using IntelliSense in your applications, see IntelliSense in VS Code. The extension also supports Linting using ESLint. The same standard ESLint rules used in the ServiceNow instance are available.

Functions of ServiceNow Extensions for VS Code

The ServiceNow Extensions for VS Code extension includes tools for developing on the ServiceNow AI Platform.

The extension adds several functions to your Visual Studio Code (VS Code) implementation. You access these functions through the command palette.

Visual Studio Code command palette



Available functions in the ServiceNow extension for VS Code

Functionality	Description
Setup Now Workspace	Create the project folder to work with ServiceNow applications. For details on creating this workspace, see Set up a workspace in VS Code.
Activate Now Extension	Activates the ServiceNow extension. See Activate ServiceNow Extensions for VS Code.
Create Now Project	Import existing ServiceNow applications to a Visual Studio Code project. See Create a project in VS Code.
Sync Current Project	During development, synchronize all files between VS Code and the instance. See Synchronize the current project between a Visual Studio Code workspace and a ServiceNow instance.
Sync Current File	Synchronize the current file you are working on. See Sync the current file between a Visual Studio Code workspace and a ServiceNow instance.
Clear Credentials	Deletes the entered credentials in the Settings page. See Clear instance credentials in Visual Studio Code.
Reset Project	Sets project to its original state. See Reset a project in Visual Studio Code.
Configure File Types	Modify the options selected in the metadata. Metadata determines which elements of your application you want to synchronize with VS

Available functions in the ServiceNow extension for VS Code (continued)

Functionality	Description
	Code. See Import an application into Visual Studio Code.
Add Custom File Types	Add file types to your project using the Custom File Types wizard. See Add custom file types in Visual Studio Code.
Select Application	Switch between application within the workspace in VS code. See Import an application into Visual Studio Code.
Select Update Set	Select an update set. Changes synchronized to your instance are applied to the selected update set. For more information on update sets, see Import an application into Visual Studio Code.
Create New File	Create records in your application. For details, see Create a file in VS Code.
Global File Search	Find files within the instance. See Search files on your instance in VS Code.
Run Background Scripts	Run a background script on your instance. See Run background scripts using VS Code.
IntelliSense	An IntelliSense code-completion aid available for Glide APIs. See IntelliSense in VS Code.

Install ServiceNow Extensions for VS Code

Download and install ServiceNow extensions to begin editing your applications in VS Code.

Before you begin

Role required: none

In order to install ServiceNow Extensions for Visual Studio Code, you must have VS Code version 1.38.0 or later installed.

Procedure

1. Open VS Code and navigate to the **Extensions** tab by clicking the Extensions tab icon () or pressing Control+Shift+X on Windows or Command+Shift+X on MacOS.
2. In the Visual Studio Code Marketplace, enter `ServiceNow` or `ServiceNow Extensions` in the **Search Visual Studio Code Extensions** search box.
3. Select **ServiceNow Extensions for VS Code** in the search results and click **Install** to proceed with installation.
4. After the installation process completes, restart Visual Studio Code.
5. Navigate to **View** > **Command Palette** in Visual Studio Code.
   You can also use a keyboard shortcut (Control+Shift+P on Windows and Command+Shift+P on MacOS) to open the command palette.
6. Choose **Developer: Reload Window** from the list.

What to do next
Once you reload VS Code activate the ServiceNow extension.

Activate ServiceNow Extensions for VS Code

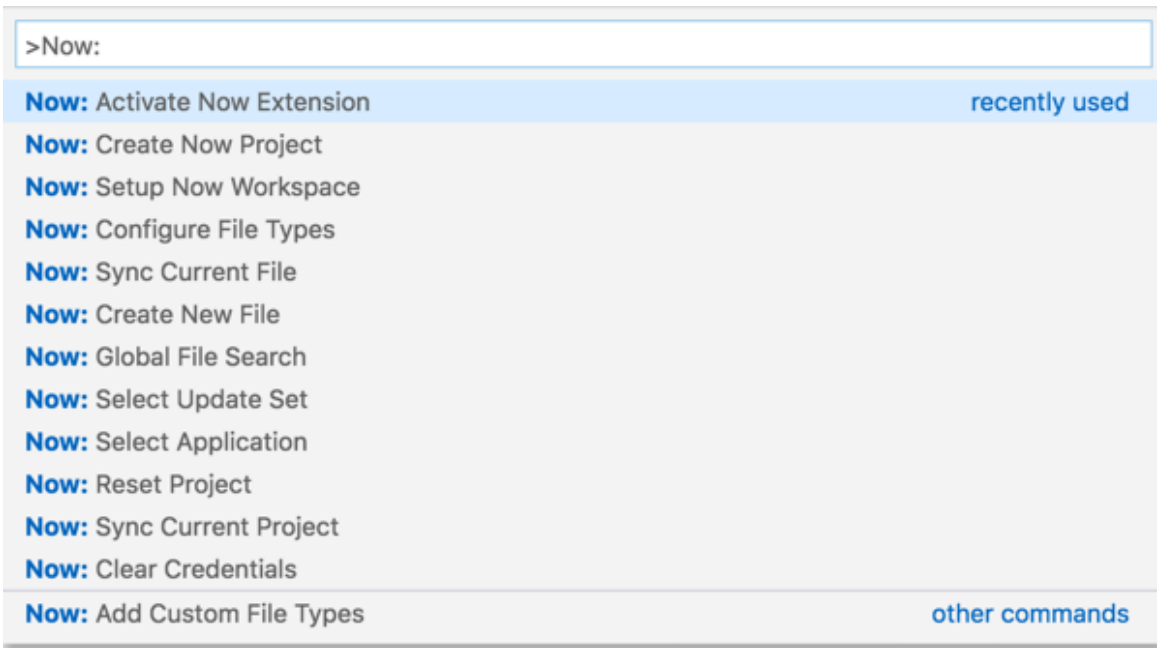
Activate the ServiceNow Extensions for VS Code to be able to edit applications within your ServiceNow instance.

Before you begin
Install ServiceNow Extensions for VS Code to activate the extension and begin editing your applications in VS code.

About this task
Activating the extension is the first step you must perform after installation of the extension to access its functionalities.

Procedure
1. Navigate to **All** > **View** > **Command Palette** to open the command palette.
   You can also use a keyboard shortcut, Control+Shift+P on Windows or Command+Shift+P on MacOS, to open the command palette.
2. Choose **Now: Activate Now Extension** from the list.

Activate Now extension



The **Setup Workspace** link appears at the bottom of the VS Code IDE. After the extension is activated, Set up your workspace in VS Code to begin editing ServiceNow applications.

Set up a workspace in VS Code

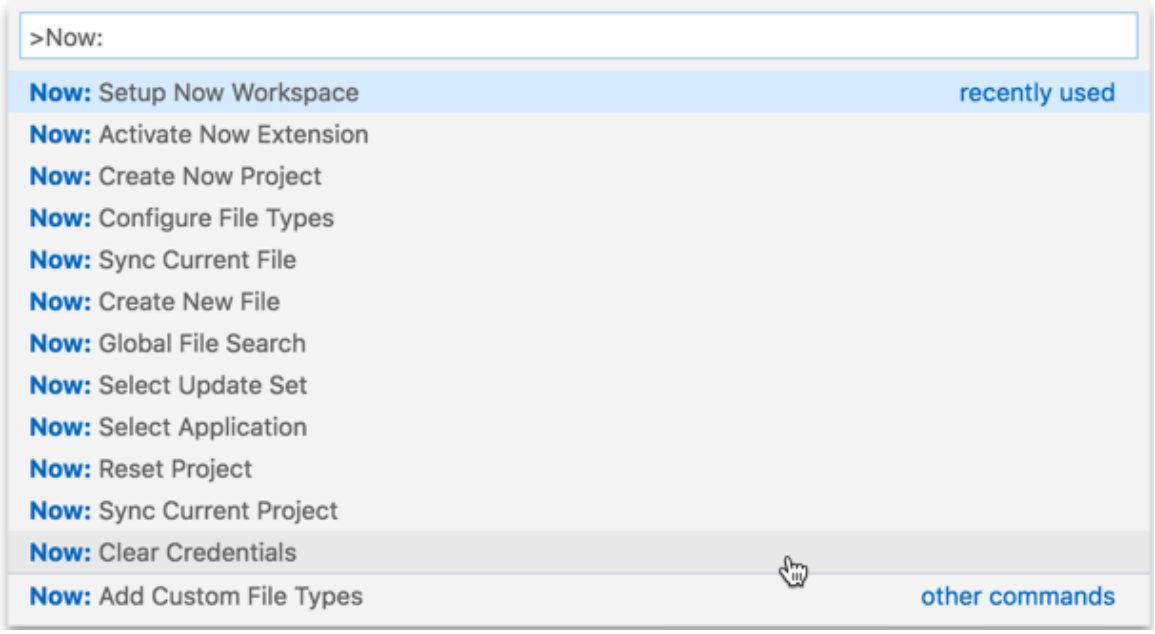
Using the ServiceNow Extensions for VS Code, create a project work folder to use as a workspace for ServiceNow applications.

Before you begin
Activate the workspace to access the functionalities of ServiceNow Extensions for VS Code. Role required: admin

Procedure

1. Click the **Setup Workspace** on the status bar at the bottom of the VS Code IDE.
   You can also use a keyboard shortcut, Control+Shift+P on Windows or Command+Shift+P on MacOS, to open the command palette and choose **Now: Setup Now Workspace** from the list.

Setup Now Workspace command



2. Do any of the following actions.
   The workspace is created from the selected folder.

What to do next

Create a project in your workspace. You can create multiple projects of different project types within a workspace.

Create a project in VS Code

ServiceNow applications are contained within a project in the VS Code IDE. Import an application from your ServiceNow instance with the help of ServiceNow Extensions for VS Code.

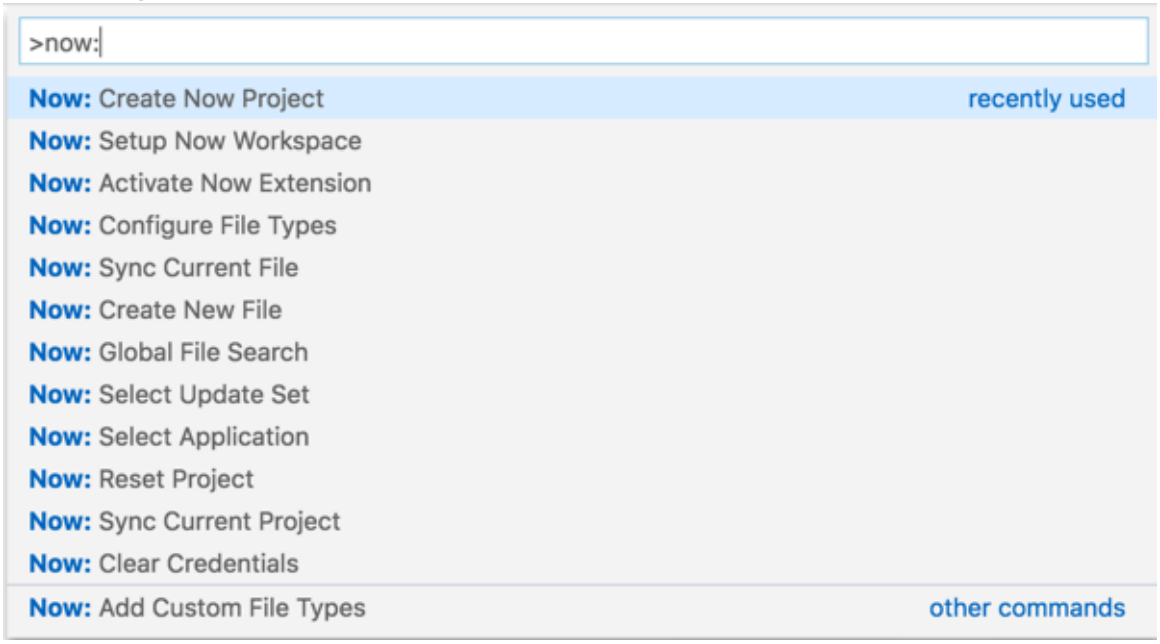
Before you begin

Setup your workspace to create a project and start editing your ServiceNow applications.

Procedure

1. Click **Create Project**in the status bar on the bottom of the VS Code IDE.

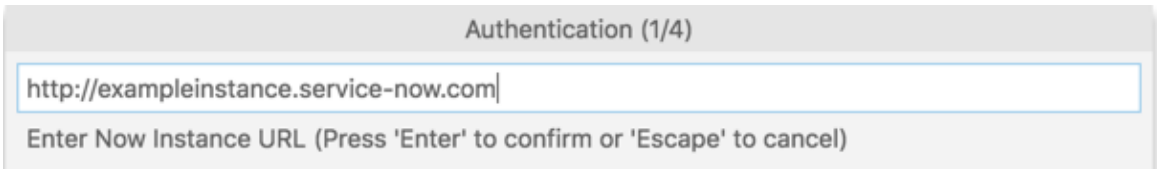
Create Project



You can also use a keyboard shortcut, Control+Shift+P on Windows or Command+Shift+P on MacOS, to open the command palette and choose **Now: Create Now Project** from the list.

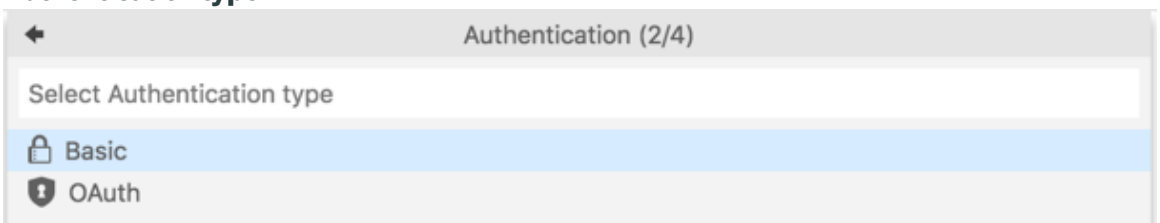
The **Create Project** wizard opens.

2. Enter the URL of the ServiceNow instance and press Enter to confirm.
Include the complete URL for your instance, for example, `https://example.service-now.com`



3. When prompted, select the Authentication type.

Authentication type



- To create a basic application, choose **Basic** and fill in the following fields, when prompted.

Basic application

Field	Description
User Name	Enter the User name of the ServiceNow instance URL.
Password	Enter the password of the ServiceNow instance URL.

Note:
The user account being used for authenticating must not contain the admin role.

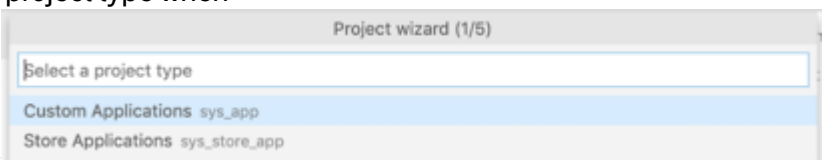
Project type

- To create an OAuth application, choose **OAuth** application and fill in the following fields, when prompted.

OAuth application

User Name	Enter the Username of the ServiceNow instance URL.
Password	Enter the password of the ServiceNow instance URL.
Client ID	Client ID is automatically generated by ServiceNow OAuth server.
Client secret	TClient secret for the OAuth application.

4. Select the project type when prompted.



Note:
You can enable the Packages and Plugins project types by enabling them from the Settings. Navigate to **Preferences** > **Settings** > **Extensions** > **Now** > **Project Settings** to enable them.

5. Select **Import Existing**.

Import an application into Visual Studio Code

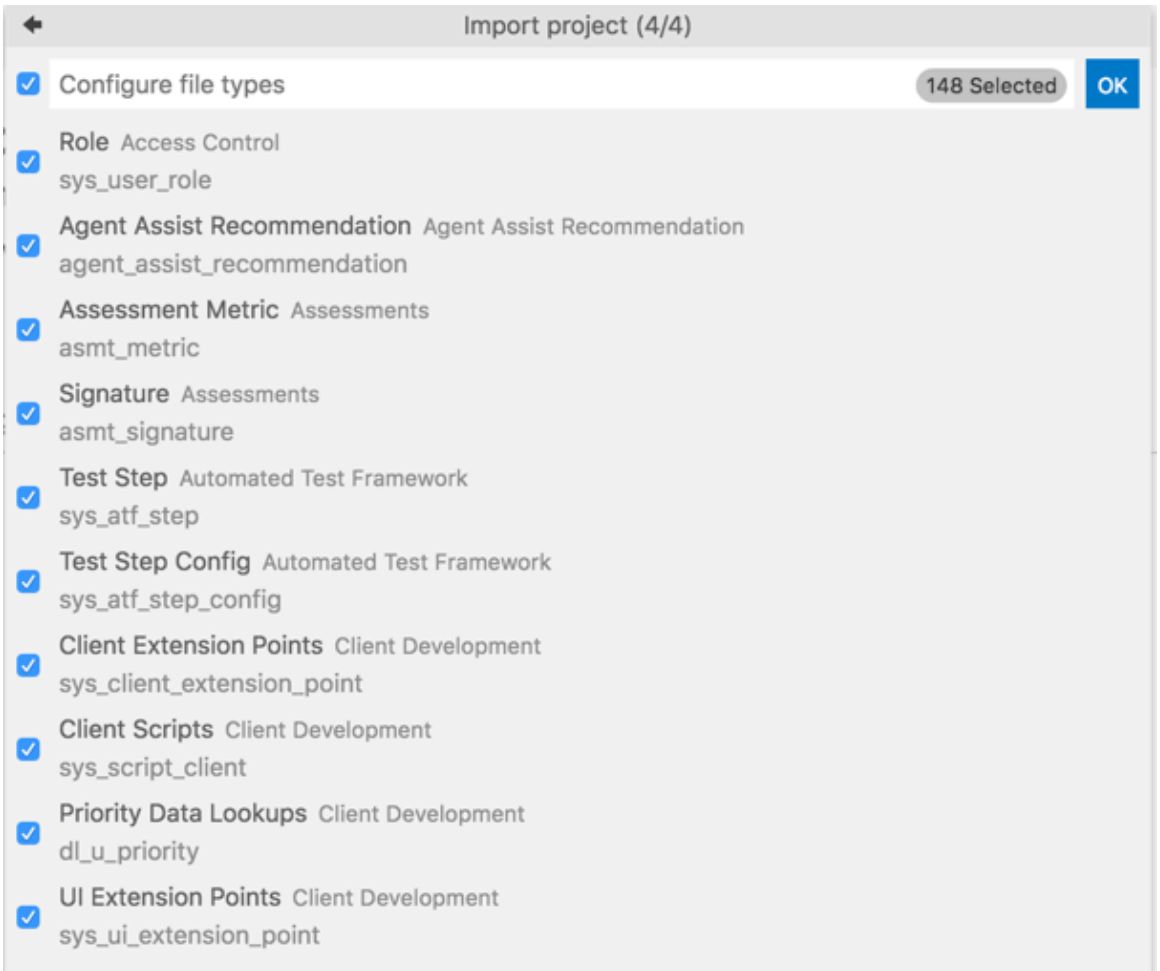
After you create a project, import an application from your instance into the project to begin editing.

Procedure

1. Visual Studio Code displays the list of applications available in your instance for the selected application type.

   The list of applications shows once the Manage Developers settings have been applied for the custom application intended to import. If the settings are left unchecked, applications will not be listed in the drop-down.

2. Select an application.
3. At the **Configure file types** prompt, click **OK** to select all file types.



By default, Visual Studio Code imports all the file types of your application. You can modify the selection of the file types while editing the application, by choosing **Now: Configure File Types** from the command palette and choosing the file types from the list.
Your ServiceNow application is imported into your workspace.

> **Note:**
> Even if your application exists on your instance, you must create a project for it in Visual Studio Code.

You can switch applications within the workspace by clicking the name of the application, for example, EmployeeApp in this case, in the status bar at the bottom of the VS code IDE or choosing **Now: Select Application** from the command palette and selecting the application name from the list. Similarly, you can modify the update set by clicking the current update set

icon, for example, EmployeeApp in this case, in the status bar at the bottom of the VS code IDE or by choosing **Now: Select Update Set** from the command pallet and selecting the name of the updateset.

Synchronization between Visual Studio Code and a ServiceNow instance

You can synchronize your applications between Visual Studio Code and your ServiceNow instance.

Synchronization

Changes made to your applications in Visual Studio Code are stored on your local file system only until you synchronise with your instance.

Synchronization accounts for the following changes:

- File modifications on the instance or in the local file system
- File creation on the instance or in the local file system
- File deletion on the instance or in the local file system

You can sync your currently selected record or all records in the current application.

Sync the current file between a Visual Studio Code workspace and a ServiceNow instance

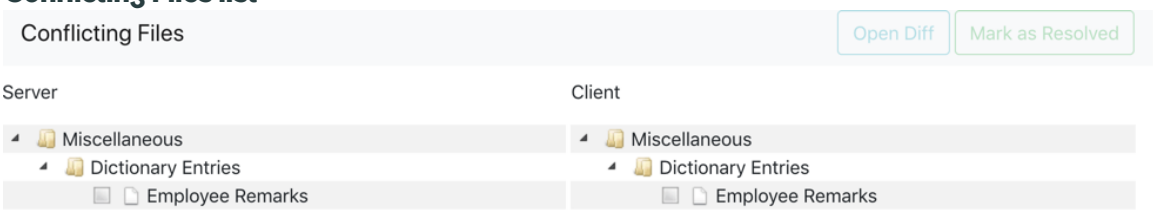
Identify conflicting files and merge or overwrite the changes, between the current file in your Visual Studio Code workspace and your instance.

Before you begin
Role required: admin

Procedure

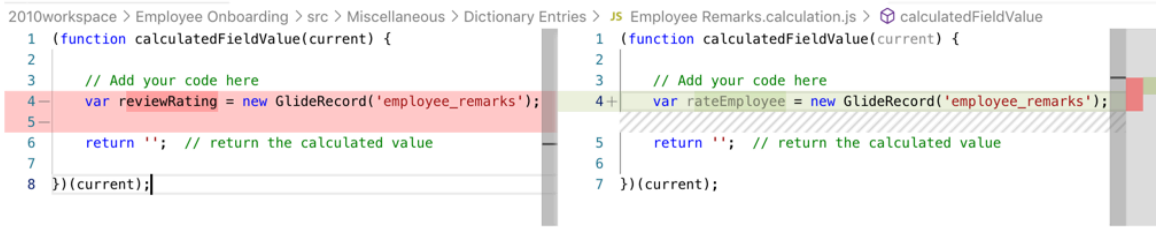
1. Click the **Sync** icon from the menu on the top-right corner of the Visual Studio Code IDE. You can also use a keyboard shortcut Control+Shift+P in Windows and Command+Shift+P in MacOS to open the command palette and choose **Now: Sync Current File** from the list.
   - If the changes occur either on the server or on the client, the extension synchronizes the file successfully without showing up any errors.
   - If there are no differences between the local and server versions, then the message **No changes detected** appears on the bottom of the page.
   - If the same file is modified both on the client and the server, then the system displays the conflicts in a **Conflicting Files** dialog box.
2. If a conflict list is displayed in a **Conflicting Files** dialog box, check the conflicts.
   - Compare the **Server** and **Client** version using the **Diff** window, and evaluate the changes.
   - Override the **Server** or **Client** changes.

Conflicting Files list



3. Check for the differences in the **Conflicting Files** list.

Conflicting changes display

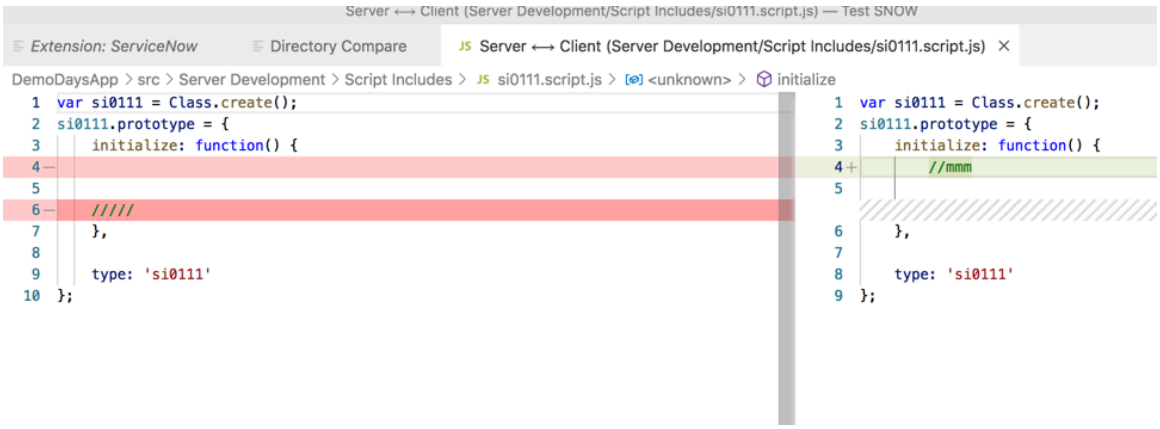


A dialog box displays the following options:

- Open Diff: Check for the differences between two versions and make the necessary changes.
- Mark Resolved: Resolve the differences and sync the versions.
- Overwrite Server: Choose if you want to override the server version with the client version.
- Overwrite Local: Choose **Override Local** to override the local version with the server version.

4. Click **Open Diff**.
   You can see two versions of the file with the differences highlighted.

Differences in client and server



5. Make changes to the server version from the client version and save the file.
6. Click **Mark as Resolved** under **Conflicting Files**.
   The changes are successfully synchronized upon the next synchronization. When the changes are merged successfully, the message **Sync completed successfully** appears at the bottom of the page.

Synchronize the current project between a Visual Studio Code workspace and a ServiceNow instance

Synchronize files between the Visual Studio Code workspace (client/local) and the instance (server) bi-directionally.

Before you begin

Create your projectand start editing your applications.

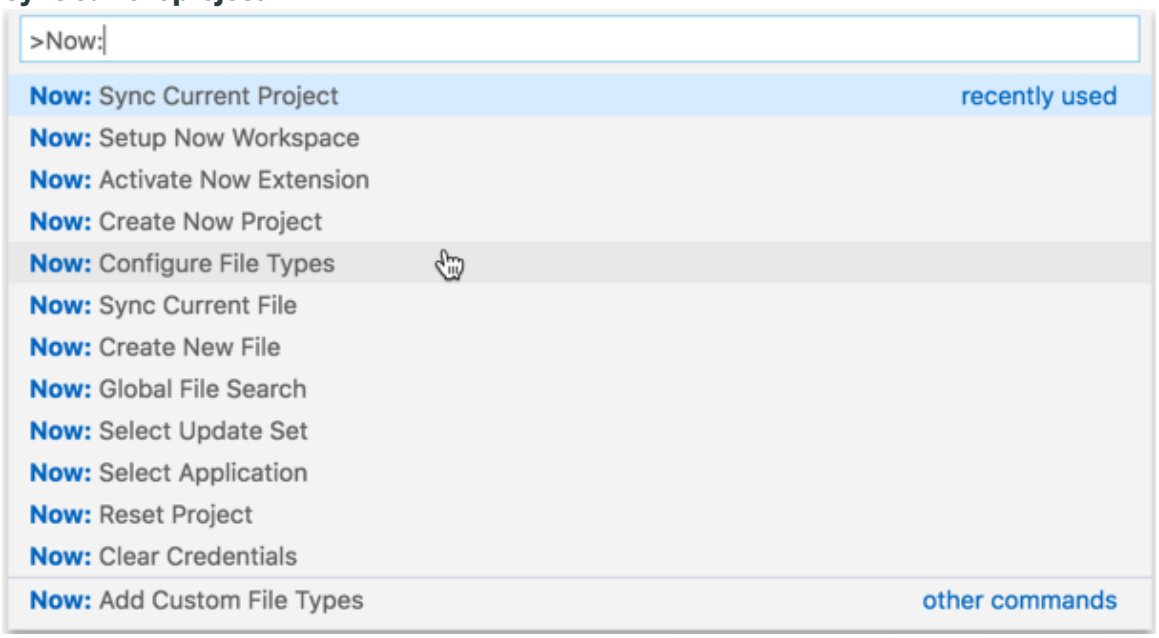
About this task

The changes made to the application files in the Visual Studio Code workspace are initially stored in the local file system. When you run the **Sync** command, the local changes made to the application files are sent to the server for persistence.

Procedure

1. Click **Sync** on the status bar on the bottom of the Visual Studio Code IDE.
   You can also use a keyboard shortcut, Control+Shift+P on Windows or Command+Shift+P on MacOS, to open the command palette and choose **Now: Sync Current Project** from the list.

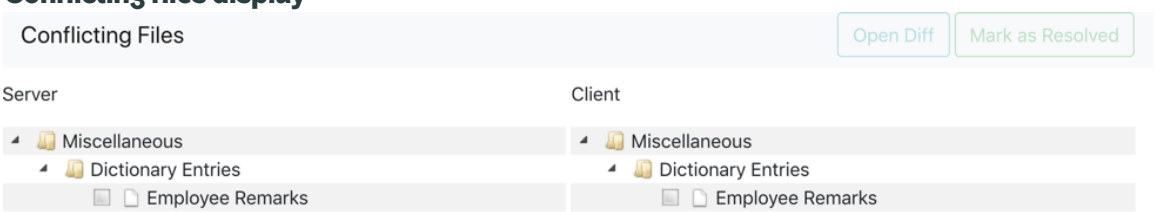
Sync current project



   - If the changes occur either on the server or on the client, the extension synchronizes the file successfully without showing up any errors.
   - If there are no differences between local and server versions, then the message **No changes detected** appears on the bottom of the page.
   - If the same file is modified both on client and server, then the system shows up the conflicts.

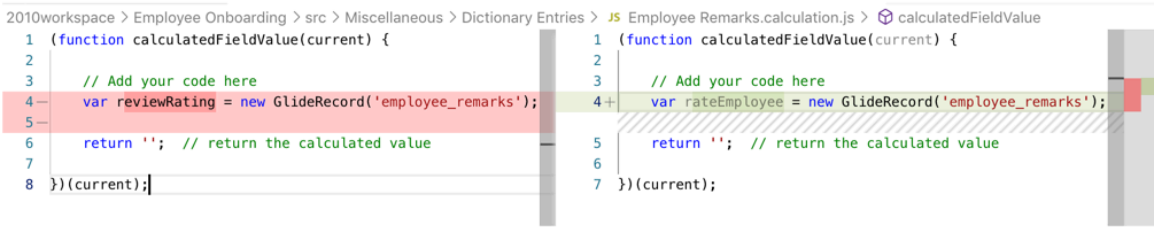
2. If a conflict list is displayed, right-click on the conflicting file to open the context menu and select from the context menu.
   Alternatively, select the file and click **Open Diff** menu from the top of the page.

Conflicting files display



   - Choose **Override Server** if you want to overwrite the server version with the client version
   - Choose **Override Local** to overwrite the client version with the server version.
   - Choose **Mark as resolved** to remove the flag indicating a conflict. The server version will be overwritten in the next synchronization.

Differences in client and server versions



3. Retain the desired changes in the client file and save.
4. Once you have resolved all file conflicts, click **Mark as Resolved** under **Conflicting Files**.
   The changes are synchronized on the next sync. When the changes are merged successfully, the message **Sync completed successfully** appears in the bottom of the page.

Note:
The files are checked for syntax errors at the beginning of synchronization process. Any errors found are presented in the problem tab at the bottom of the screen.

What to do next

All the changes are tagged to the update set in the status bar. When a project is loaded or selected, the default update set is displayed in the status bar. Click the update set picker to choose the current update set (valid until the IDE or project is closed). If you do not explicitly select an update set, all writes happen to the default update set.

Clear instance credentials in Visual Studio Code

Clear the stored credentials of the current project from the system. User credentials (instance URL, user name, password) of a project are stored in the operating system credentials vault so that the user does not need to log in each time.

About this task

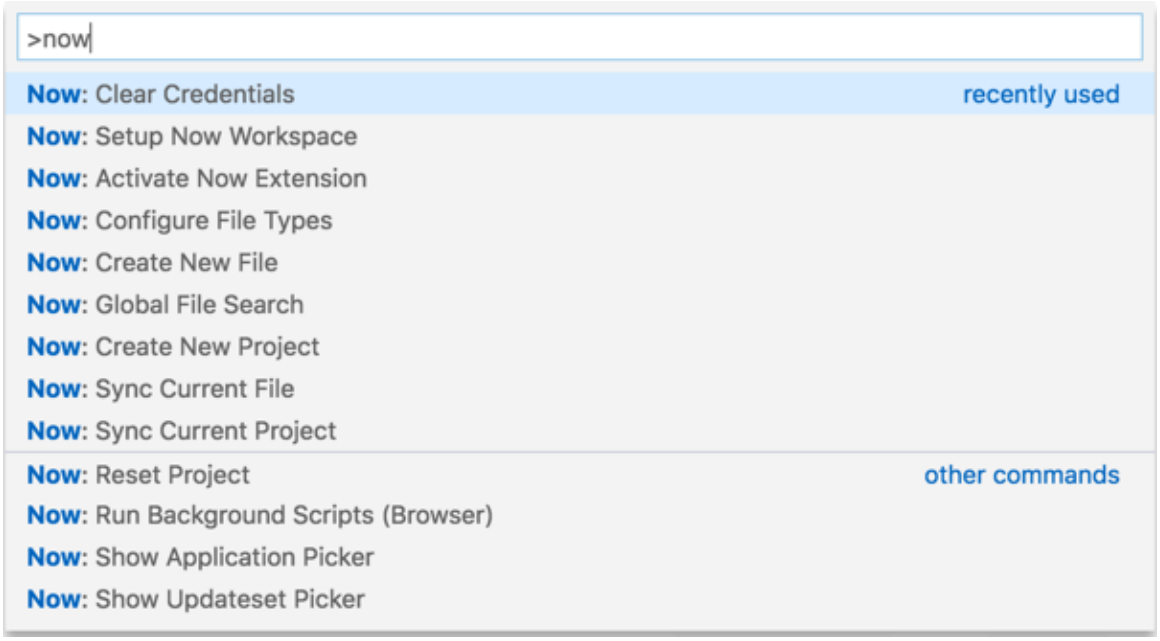
Once you clear credentials from the current instance, the ServiceNow Extensions for Visual Studio Code extension prompts for authentication the next time you open VS Code. You can continue working in the current session.

Note:
The password is not stored in the Visual Studio Code settings.

Procedure

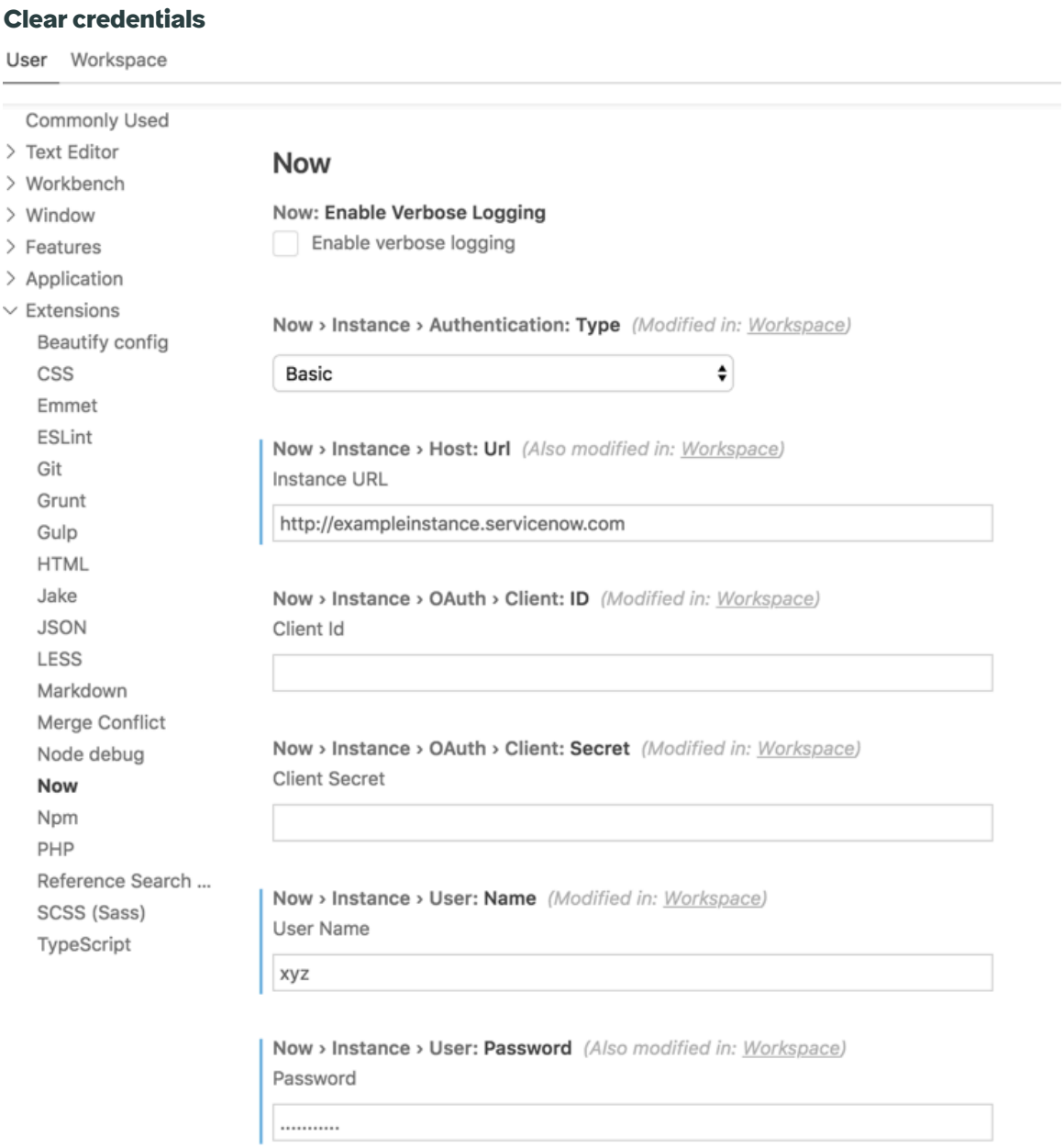
1. Navigate to **All** > **View** > **Command Palette**.
   You can also use a keyboard shortcut, Control+Shift+P on Windows or Command+Shift+P on MacOS to open the command palette.
2. Choose **Now: Clear Credentials** from the command palette.

Now: Clear credentials command



The credentials of the current project are cleared.

3. **Optional:** Navigate to **Code** > **Preferences** > **Settings** > **Workspace** > **Extensions** > **ServiceNow** to check whether the credentials and the instance URL have been deleted.



Note:
To re-enter your credentials, navigate to **Code Preferences Settings Workspace Extensions Now**. Enter your credentials in **Username** and **Password** fields.

Reset a project in Visual Studio Code

Reset the project to the state on the server by discarding all the local changes if you encounter any serious sync issues.

About this task

When you reset a project to its server state, all unsynchronized changes are lost.

Procedure

1. Navigate to **All** > **View** > **Command Palette** in.
   You can also use a keyboard shortcut, Control+Shift+P on Windows or Command+Shift+P on MacOS, to open the command palette.

2. Choose **Now: Reset Project** from the drop-down list.
3. Click **Yes** in the dialog box.

Result
The project is reset to its original state.

Note:
This option should be used only when something goes wrong, for example, if the project is corrupted or you are facing sync issues.

Related topics

Create a project in VS Code

Add custom file types in Visual Studio Code

If you have work with file types other than the default types provided, you can add additional file types to your instance and edit them in ServiceNow extension for Visual Studio Code.

Before you begin
Role required: admin

Procedure
1. Create a new file type in your ServiceNow instance and inherit it from the application file.
2. Add custom columns of the type *`Script`* and *`String`*, and enter some data.
3. Navigate to **View** > **Command Palette** in Visual Studio Code.
   You can also use a keyboard shortcut, Control+Shift+P on Windows or Command+Shift+P on MacOS, to open the command palette.
4. Choose **Now: Add Custom File Types** from the command palette.
   The command fetches all the existing tables in the current application and opens the **Add new file type** wizard.
5. Select the file type you created in your ServiceNow instance from the list.
6. At the Select tags prompt, select the tags you created in the instance.

   All tags are selected by default. Click a tag's check box to deselect it.

Select tags dialog box

The new file type is successfully added to the `app.config.json` file. You can also add the desired file types manually to the `app.config.json` file under your project. See the example below.

7. Absorb this new file type to the current project using the **Now: Configure File Types** command from the command palette.
8. Select the file type you created from the list of file types and click **OK**.
   The selected file type is ready for editing in the ServiceNow Extensions for Visual Studio Code extension.

Example:
The following is an example for app.config.json

```
"CustomFileTypes": {
        "sc_ic_aprvl_type_defn": {
            "superCoverName": "Miscellaneous",
            "coverName": "Approval Type Definition",
            "tags": {
                "approver_script": "js",
                "approver_html”: "html",
            }
        }
    }
}
```

- superCoverName is a name of the super parent directory (Should always pointtoMiscellaneous).
- coverName is a descriptive name of the table.
- sc_ic_aprvl_type_defn is a table identifier.
- tags represent the set of scriptable columns in the table
- approver_script is a name of the table column / xml tag
- js is an extension of the file (js | html | css | json)

Create a file in VS Code

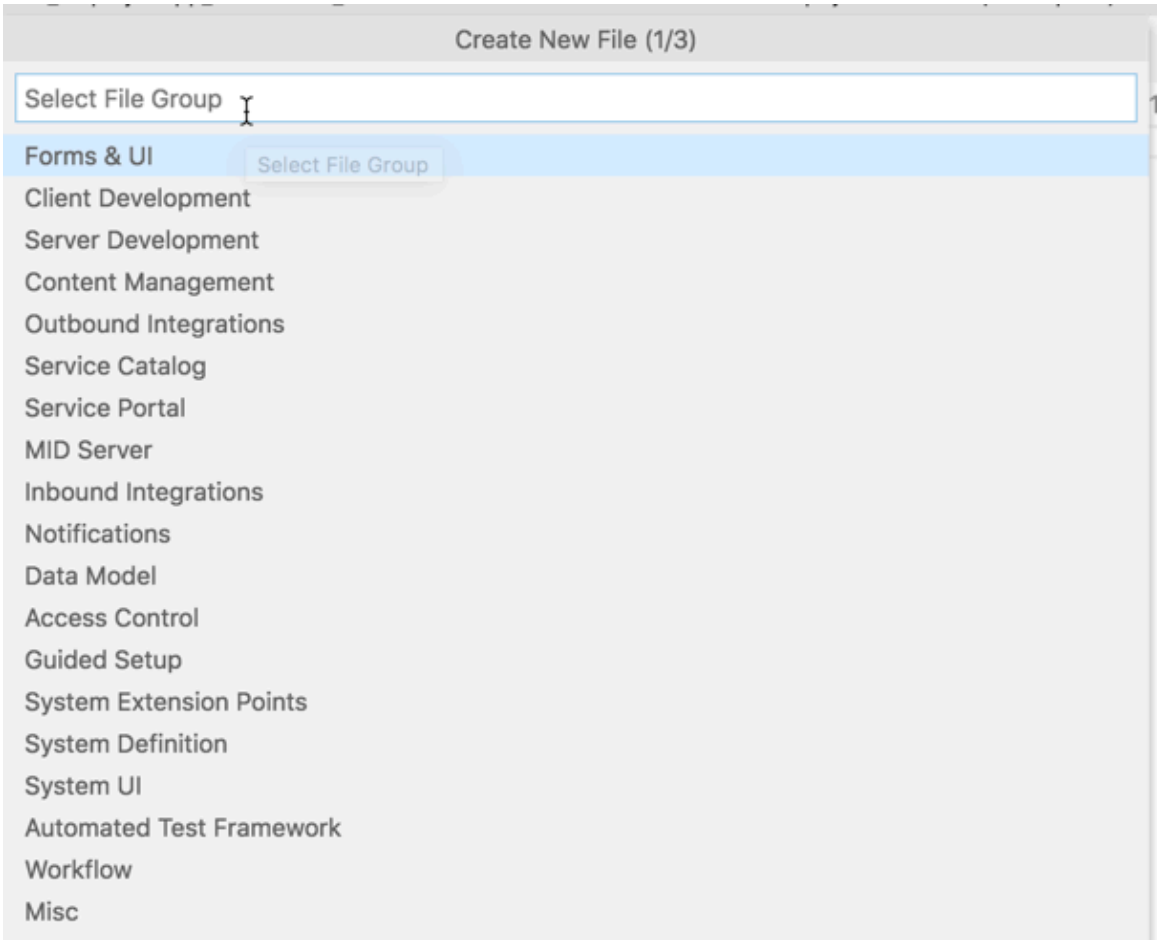
Use the ServiceNow extension to create a new file for your application with your project.

Before you begin
Import an application in your VS Code project to create a file within your project.

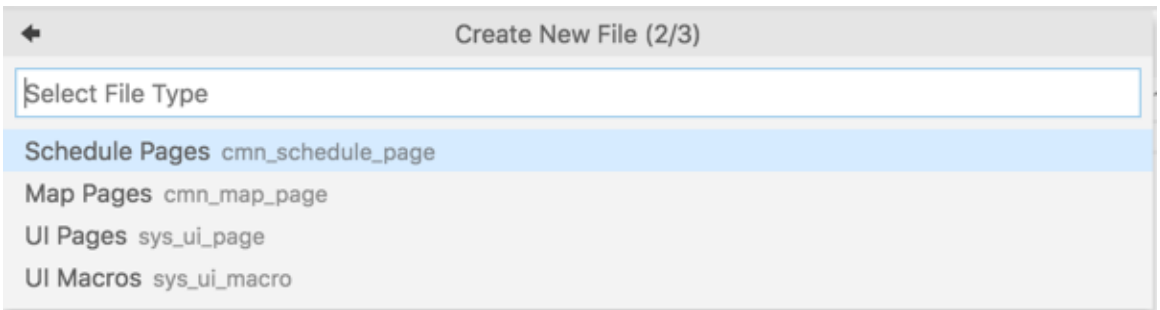
Procedure

1. Open the VS Code project containing your application.
2. Click the **Create File** (

) on the status bar on the bottom of the VSCode IDE.
   You can also use a keyboard shortcut, Control+Shift+P on Windows or Command+Shift+P on MacOS, to open the command palette and choose **Now: Create New File** from the list.
3. Select the file group from Select File Group list.



4. Select the file type from the **Select File Type** list from the command palette at the top of the screen.



5. Enter a name for the file in the Create New File dialog box.
   The extension creates a file in your application on your local file system. The new file is added to your instance when you synchronize your workspace.

Search files on your instance in VS Code

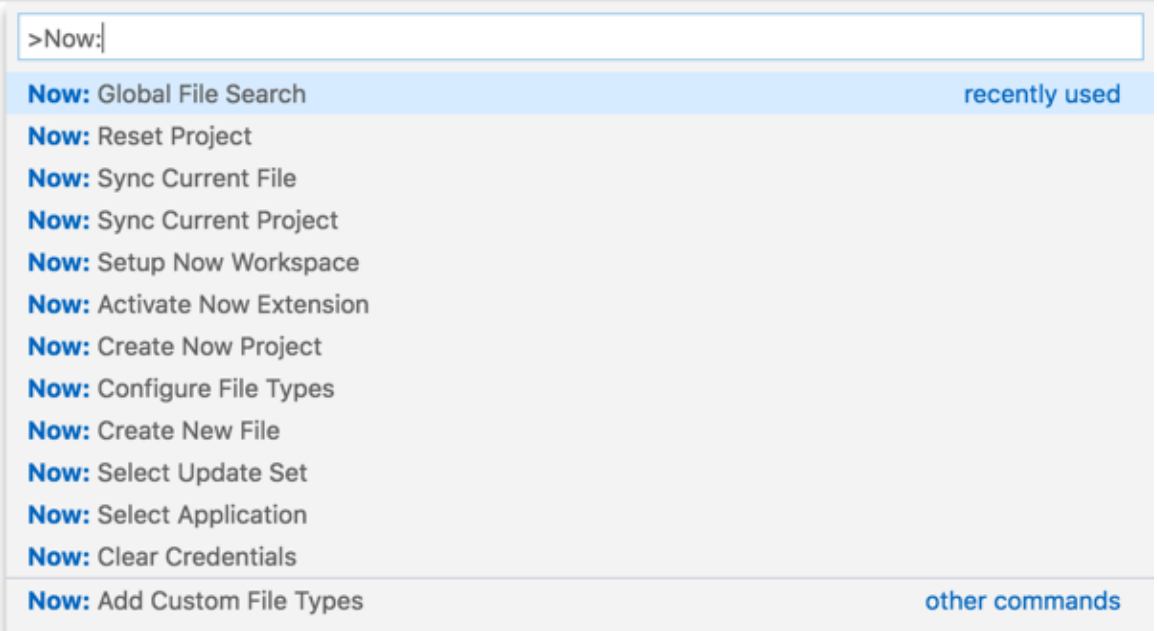
Search and download any script files on your instance using the ServiceNow Extensions for VS Code extension.

Before you begin
Role required: none

Procedure

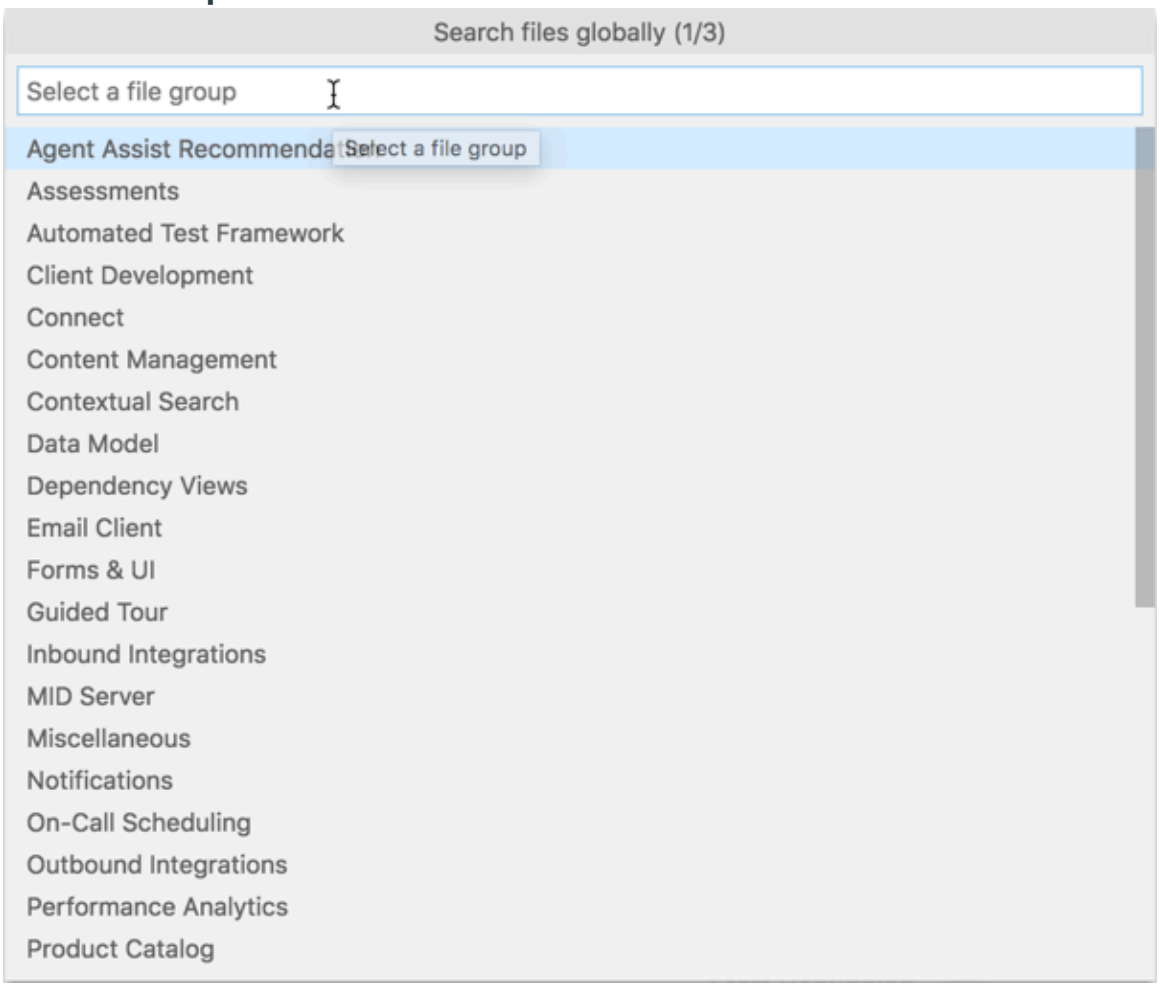
1. Open a project in VS Code containing your application.
2. Click the **File Search** on the status bar on the bottom of the VSCode IDE.
   You can also use a keyboard shortcut, Control+Shift+P in Windows or Command+Shift+P on MacOS, to open the command palette and choose **Now: Global Search** from the list.

Global File Search command



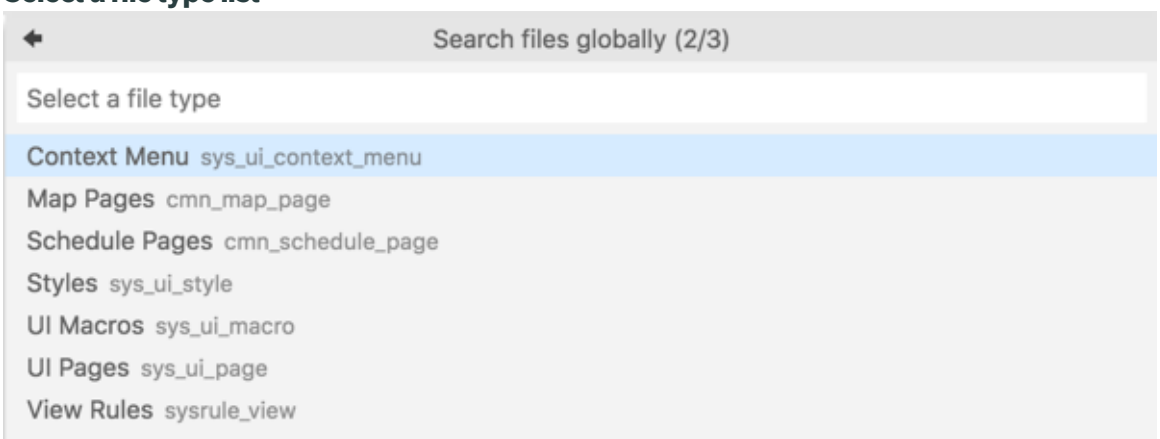
3. Select the file group from the **Select File Group** list in the command palette on the top of the screen.

Select File Group list



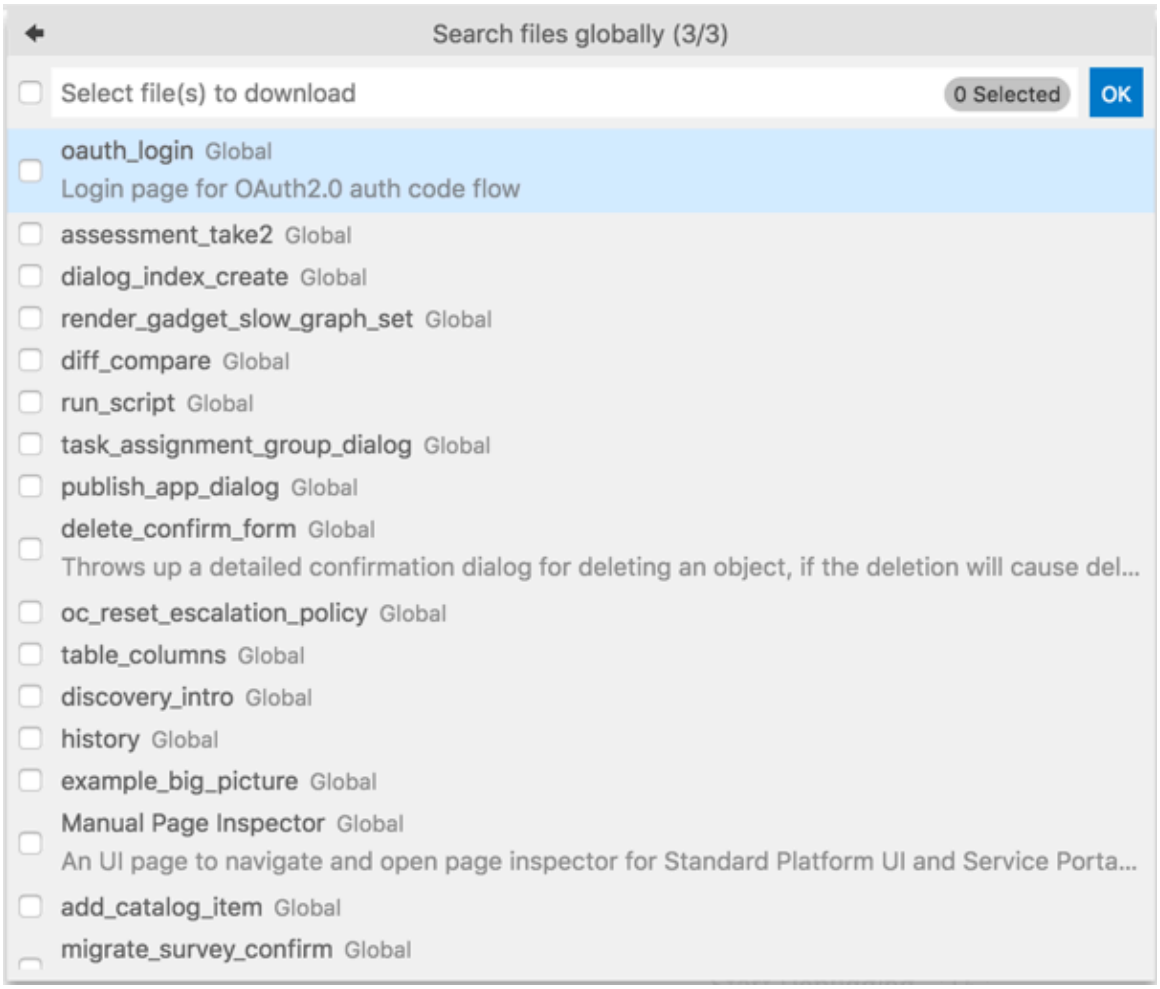
4. Select the file type from the Select a File Type list in the command palette on the top of the screen.

Select a file type list



5. Select files from the **Select files to download** list.

Select files to download list



You can select and download multiple files at a time.

The selected files are downloaded to the `scratch` folder of your application. You can edit the selected files.

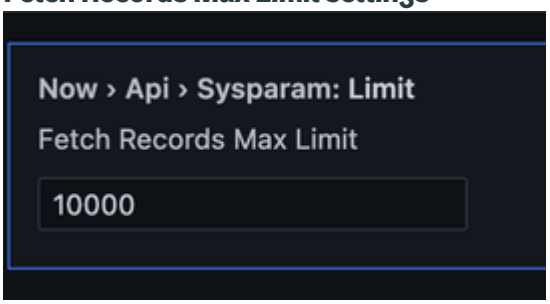
6. View more than 10,000 results from your search:

   a. Open settings by selecting Control+, in Windows or Command+, on MacOS.

   b. Enter `now` into the search bar.

   c. Enter the number of records you want to see in the Fetch Records Max Limit settings.

Fetch Records Max Limit settings



Related topics

Create a file in VS Code

Run background scripts using VS Code

Run scripts from your ServiceNow instance in VS Code using the ServiceNow Extensions for VS Code.

Before you begin

Role required: admin

Procedure

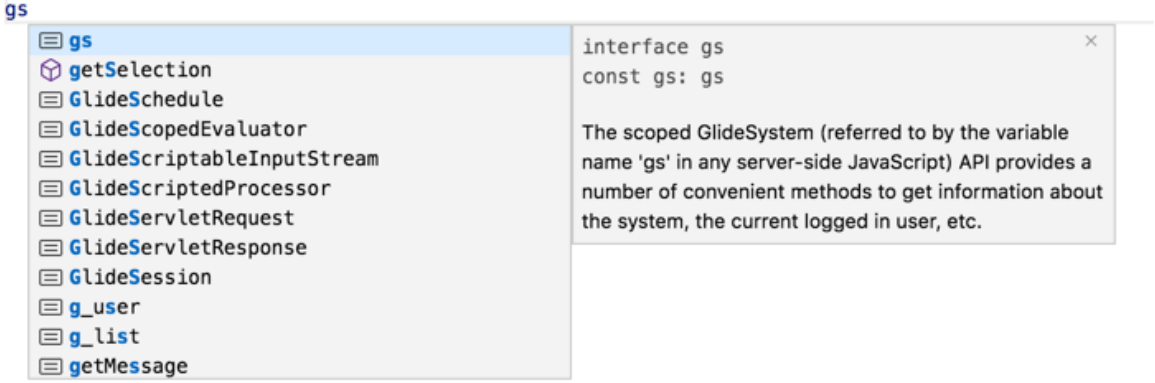
1. While writing the script in the VS Code IDE, select the script in the text file and right-click on the script.
2. Select **Run Background Script- Current** or **Run Background Script- Global** from the context menu.
   The background scripts runner tab opens in VS Code and shows the location where the script is executed in the server.

IntelliSense in VS Code

The IntelliSense code editing feature in the ServiceNow Extensions for VS Code extension provides intelligent code completions which provide auto-completion options to make writing code faster and more efficient.

When you start scripting in the VS Code, the IntelliSense feature shows auto-completion options suitable for your code context at the top of the list.

IntelliSense for GlideSystem



The auto-completion options available are:

- Auto-completion of methods and properties for client and server scripts.
- Auto-completion of methods and properties for standard JS APIs
- Auto-completion of Jelly tags and attributes.
- Code completion in commonly used standards.

The extension supports snippets (templates) for commonly used code patterns and loads snippets defined by the user in Macros table of the Syntax Editor on an instance. Snippets appear in IntelliSense when you type `#Space` along with other suggestions. Alternatively, snippets can also be inserted using the **Insert Snippet** command from the command palette in VS Code.

Telemetry in ServiceNow Extensions for VS Code

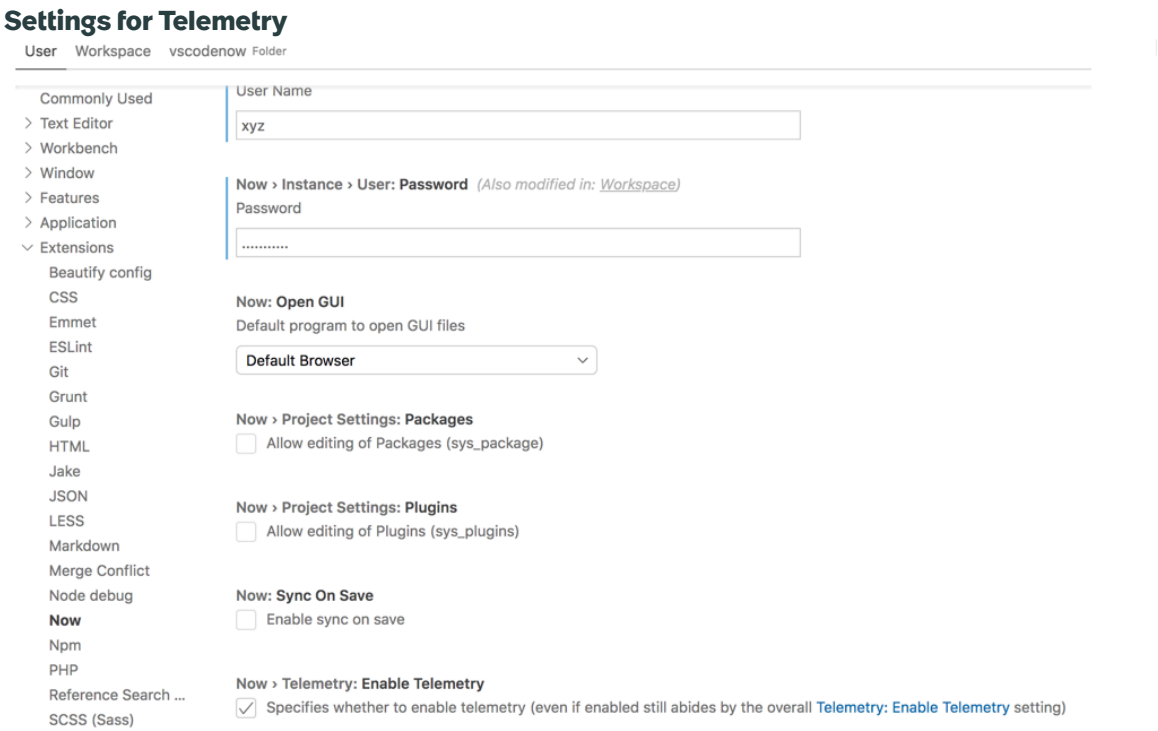
ServiceNow Extensions for VS Code collects information on its various usage patterns such as the commands run with the help of Now Telemetry.

Now Telemetry helps in the following

- Debug issues
- Analyse most used functions or features
- Capture user actions
- Improve the extension

Note:
Telemetry collects anonymous information related to usage and does not capture any personal information such as name and email ID of the users. Data is accessible to ServiceNow only and is not shared with anyone

Telemetry is enabled by default for the ServiceNow extension. If you do not wish to send usage data, you can set the `now.telemetry.enableTelemetry` user setting to false. Alternatively, select **File** (Windows) or **Code** (Mac) and navigate to **Preferences** > **Settings** > **Extensions** > **Now** and clear **Now Telemetry** check box to disable the feature.



Note:
The usage data will be sent until you disable the setting. The extension abides by the global telemetry setting `telemetry.enableTelemetry`. If that is set to false, Now telemetry is disabled.

Exploring professional development

ServiceNow provides a single mobile and web application development platform to quickly build business applications to power digital transformation. Learn how anyone can automate, extend, and build digital workflow apps across the enterprise with the ServiceNow AI Platform.

This content covers the application development stages to help developers easily build applications on the ServiceNow AI Platform.

- Plan your application development: Think through the application before building it.
- Build your application: Create and configure the application and its components.
- Validate app functionality: Perform functional testing and write test cases.
- Deploy your app: Move the application into production.

Plan your application development

The planning process is the most important step in building an application. Thinking through design and configuration before building has both immediate and long-term benefits for the application.

Fit for the ServiceNow AI Platform

Not every application idea is a good fit for the ServiceNow AI Platform. During the planning stage, consider what makes an application a good or a poor fit for the ServiceNow AI Platform. Keep the good fit characteristics in mind while planning the application.

Good fit	Poor fit
• Simple forms • Task management • Request management • Spreadsheet-driven processes • Repeatable processes • Third-party integrations • Orchestration of multiple systems • Single experience from functions in multiple systems • Web and mobile access to the same apps and data simultaneously	• Unstructured data • Unrepeatable processes • Content that needs graphics processing • Streaming audio or video • Highly customized user interface

Plan before you build

Essentially, an application is a digital program that supports user tasks. Some actions you take when building an application might be irreversible. Be aware of these actions and plan for them in advance.

Application scope

One of the first major decisions to make when creating an application is: Should the application be in a private scope or a global scope?

By default, applications are created in their own private application scope. Applications in a private application scope restrict access to their application artifacts so only application artifacts in the same scope have full access to create, modify, remove, or run application data. Scoped applications can use source control integration and delegated development. Globally scoped applications cannot use delegated development.

Create custom business applications in scope unless:

- The application has to delete global data.
- The application needs to change application access settings on multiple default tables to function.
- The application needs to access APIs only available in the global scope. Creating a globally scoped passthrough script include would not be enough for this requirement.

Note:
Globally scoped passthrough is a script include created in a global scope that is accessible from the private scope. The passthrough gives access to a global API that is not accessible by default from a private application scope.

For more information, see Application scope and Understanding Application Scope on the ServiceNow AI Platform (Whitepaper).

Instance selection

Proof of Concept (PoC) application builds can and should be built in a separate instance from a regular development instance. The instance can be a sandbox instance or a Personal Developer Instance (PDI) from the Developer Site. The PDI naming format is **dev12345.service-now.com**.

If using an instance with a different scope namespace, rebuild the PoC applications in the organization's development instance. Do not import the applications into the organization's development instance. The scoped namespace for the applications will not match the scoped namespace for the company's development instance.

Applications the organization intends to use (i.e. production apps) should be created in the organization's developer instance, so the application can follow the organization's testing and deployment process.

Naming decisions

The application name matters. ServiceNow suggests a scope based on the application's name. Application file names are appended to the scope to uniquely identify application resources in an instance. Scope is in the format: x_[company code]_[application_name] with a maximum of 18 characters. For example, an application name **Legal Request** has a suggested scope of **x_acme_legal_reque**.

All application files within the application inherit the scope, so carefully consider what the value should be. The application name can always be changed.

Manage app development

Govern app development on the ServiceNow AI Platform.

Source control

To enable simple, future proof collaboration for applications, use source control integration. Source control allows an organization to adopt industry standard tool sets while following modern application development and team management paradigms. See the Source Control Integration documentation for more information.

Source control platforms, such as GitLab and GitHub, align an organization with current and future best practice trends in ServiceNow application development and management. However, some organizations may not be ready to adopt the current source control integration.

- Branch merging is not currently supported.
- Corporate Git repositories behind a firewall cannot be directly integrated without additional firewall configuration.

Delegated development

Delegated Development allows designated users without the admin role to develop or deploy applications on the ServiceNow AI Platform. Users with the application-specific admin role or the system-level admin role can delegate application development to designated developers at the application level. Delegated developer roles have fewer privileges than admin roles but are still allowed to develop applications.

Note:
Delegated development is only available to privately scoped applications.

For more information about building applications, see Build My First Application.

Build your application

Build and define the data, design elements, and logic that make up your application.

Define and build the data model

After planning is complete, define and build the data model. Create one or more tables with fields, load the table with demo data, and verify access controls to the data.

Create an application

Create or open an application record.

- Create: If creating an application directly, use Guided App Creator to create the application. Guided App Creater allows you to create tables from spreadsheets, roles, and security rules. Design options are also available.
- Open: If developers do not have the admin role, the ServiceNow System Administrator needs to create the application and grant developers a delegated development role. Developers then use Studio to open the application for editing.

Note:
The application scope and table name are sometimes referred to as the internal names for these objects and cannot be changed once they are created. However, the application name and table label can be changed. Application users see the internal names in the URL only. If possible, internal names should always be consistent with what the user sees.

Build the data model

Create tables and fields on the tables to support the application's data model.

ServiceNow automatically adds five fields to each new table. The new fields contain auto-populated information about the table.

Fields added to every table

Field name	Database name	Description
Created by	sys_created_by	User who created the record.
Created	sys_created_on	Date/time when the record was created.

Fields added to every table (continued)

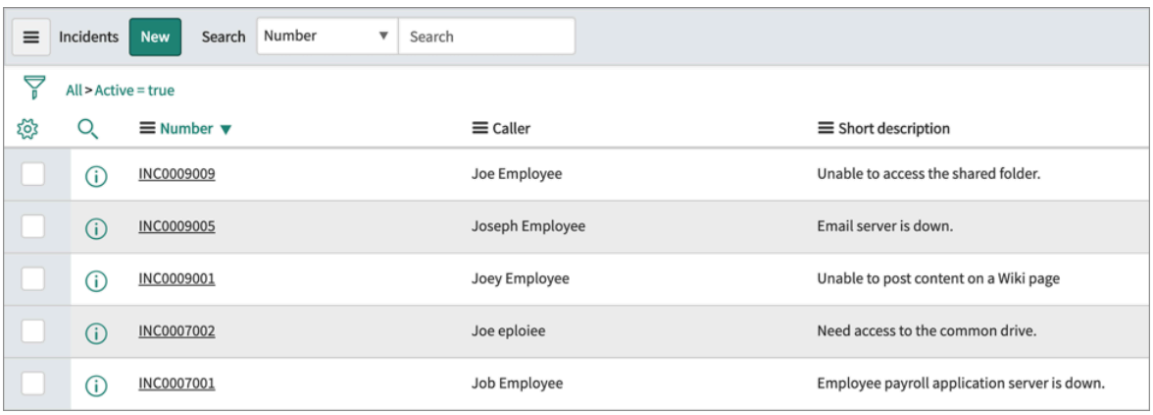
Field name	Database name	Description
Updated by	sys_updated_by	User who last updated the record.
Sys ID	sys_id	Unique identifier for the record. It is unique throughout the instance.
Updates	sys_mod_count	Numeric field that counts the number of updates to the record since record creation.

New tables can extend an existing table to inherit fields and functionality from the table being extended. Add to and modify the components of the extended table. The most commonly extended ServiceNow table is the task table. For more information, see When to create a new table vs. when to extend and Exploring ServiceNow AI Platform® tables.

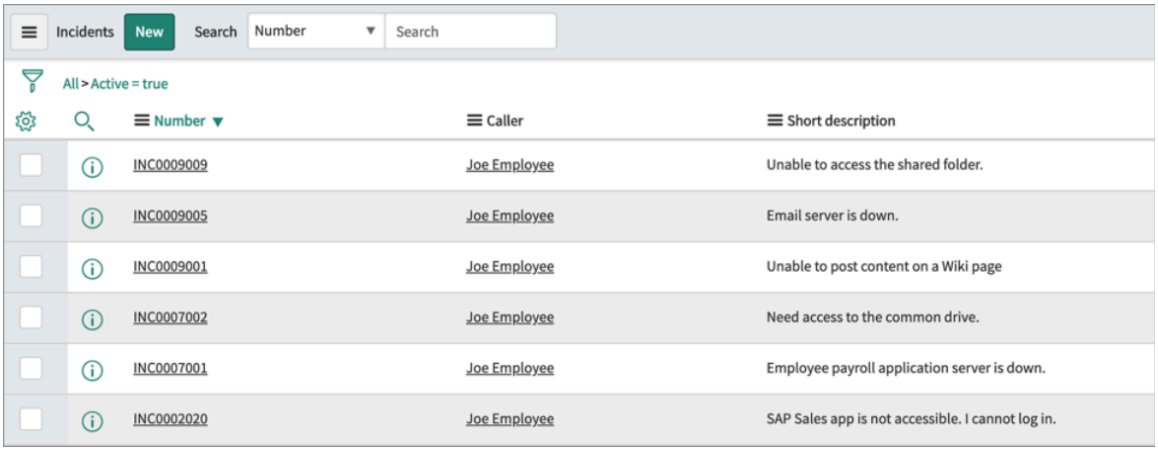
Add fields to the table to support the data model required by the application. ServiceNow has many different field types with built-in validation. Select the field type that best fits the field's data.

Note:
String (plain text) fields are the easiest to configure. However, because users can enter anything, string fields can result in bad and inconsistent data that is difficult to use.

In the example, a string field type is used for a user's name. Notice the Caller field is different for each Incident record, but the caller may be the same person. Do not use a string field type for a user's name in tables.



Instead, use a reference field type that references the User table instead of a String field. Users then need to select a single consistent record in the Caller field.



Reference fields ensure consistent data by normalizing date in another table in ServiceNow. ServiceNow has over 2000 baseline tables available to reference. The Appendix lists some commonly used tables for building an app.

While a reference field can normalize data, other fields can be used for specific types of data. The ServiceNow Documentation site has the complete list of Field types. Some common field types are:

Field type	Descriptions
Integer	Stores number values and can be used in calculations.
Currency	Holds a currency value and will show values in the currency of the logged in user.
Phone number	Includes validation and formatting for E164-compliant phone numbers.
Reference	Displays a record from another table and helps to normalize data.
Choice	Displays a select box with a predefined list of choices. Choice lists should include fewer than ten items.
Date	Stores a date value selected with a date picker. Use Date if you do not need a specific time.
Date/Time	Stores date and time values selected with a date and time picker. Use Date/Time to compare specific times or if the exact time is important.
String	Holds freeform text. Use String if no other field type matches the values stored in the field.

Note:
Field types should not be changed after a field is created.

Choice lists or reference fields

Choice lists and Reference fields both offer users a way to choose a value from a list. Choice lists are name/value pairs. Users select from the names and the field stores the value of the selected

choice. Scripts use the value. Add and remove name/value pairs from the choices to manage the list of options.

Reference fields point to a table. Manage choices in the table. The value stored in the reference field is the sys_id of the referenced record.

Choice lists do not require a reference table and are easier to configure than reference fields. Use Choice lists when the field has ten or fewer options and the options will not change. Consider using a reference field and table when:

- The field requires more than ten choices.
- The choices will regularly change.
- Someone other than an administrator needs to manage the choices.
- The value of the field has an impact on decision logic. For example, decision tables in Flow Designer.
- The data has multi-level dependencies between different fields that can lead to complex and unwieldy choice field combinations.
- The choices require more than a name/value pair. For example, referencing a user record gives the referencing table access to other user details, such as email and department.
- A table already exists that includes the data needed for the field.

When using reference fields, review the tables available in the instance to reference before creating a table. If creating a new table, check the list of exempt tables in section 2 of the Custom Table Guide. If appropriate, extend the new table from one of these.

Note:
Before creating new fields on an extended table, check for an existing field inherited from the base table that has a similar purpose. If a field is found, override the extended table's label.

Secure data

Data security is one of the most important and overlooked aspects of creating an application. ServiceNow automatically configures access control for a new or selected role during the table creation process. Only users with the role can access the table to read, create, write, and delete.

Use access control rules to configure table and column-level security in the ServiceNow AI Platform. To properly configure access to an application, developers should understand how access controls work and the order in which access controls are evaluated. Apply multiple access controls that together make an Access Control List (ACL).

Self-Paced Training: Securing Applications

Documentation: Access control list rules

When considering security:

- Protect tables, UI pages, property pages, and other content with the appropriate access controls and roles.
- Limit the use of GlideRecord queries in access control scripts. GlideRecord queries can affect performance.

Beginning with the Orlando Platform Subscription model, customers are charged by how many tables a user can access, regardless of whether the user does access the table. Configure ACLs to restrict access to a table to ensure that only the users that need access to a table can access the table.

Note:
Consider making any auto-populated fields read only. If the system is populating the data, a user should not be able to.

Alternately, secure data on the ServiceNow AI Platform with before-query Business Rules. Before-query Business Rules run before the database query and are limited to controlling read access to a record. Only use before-query Business Rules when necessary. Some considerations when deciding to use Access Controls or before-query Business Rules:

- GlideRecord queries will bypass read access controls on a table and will be restricted by before-query Business Rules on a table.
- When access controls restrict read access to records in a list, ServiceNow shows a message saying that access has been restricted for the records. With before-query Business Rules, the number of records in the list total matches the number of records shown to the user. The user receives no indication that some records have been hidden from the list.

Review the user query Business Rule on the User [sys_user] table for reference.

Note:
Before-query Business Rules do not take the place of ACLs. Denying users access to a table via before-query Business Rule will still count the table against the subscription model. Use Access Controls to prevent the table from being counted for the users in the Platform Subscription model.

Encryption

The ServiceNow AI Platform also provides various encryption solutions at the application tier, database tier, and hardware tier. Learn more in the Data Encryption Whitepaper.

Note:
Set up security before configuring any interfaces or business logic. Since security affects the data available to interfaces and business logic, waiting until the end of the application build process may cause rework and issues.

Manage data

With the data model (tables and fields) created and security set up, add data into the application's table(s).

Import sets

To populate tables in ServiceNow from an external platform, use import sets and transform maps. To retrieve data from an external source on a regular basis, use a scheduled import.

Self-Paced Training: Importing Data into ServiceNow

Guidelines for Import Set Performance:

- Break up large amounts of data into smaller sets for faster imports. Consider 100,000 rows to be a baseline and break up anything larger than that into sets of 100,000. For example, importing 10 sets of 100,000 will finish quicker than one set of 1,000,000 records. Also consider using Concurrent Imports with larger numbers of records.
- Importing large data sets simultaneously can put load on an instance. Stagger large imports so the imports do not overlap.
- If possible, deselect the Run business rules check box on the transform map table to avoid running Business Rules and other logic during an import. Consider using an onComplete

transform script to run business logic, such as calculations, at the end of an import rather than on each record as Business Rules do.

- Use default functionality for imports. Whenever possible, avoid writing custom scripts. For example, use the coalesce functionality rather than writing a custom coalesce script.
- Avoid GlideRecord queries in an import set.
- Make sure any fields configured to coalesce are indexed.
- If replacing a system with a requirement to import historical data, import only the historical data required for the application. Consider storing historical data in a data lake.

Inbound integrations

In addition to Import Sets, ServiceNow includes APIs to accept data from external platforms.

To push data directly into an application table from another system, use Web Service Import Sets instead of writing directly to application tables using the REST Table API or SOAP APIs.

To handle data transactions that are more complex than writing data to a table, such as sending an attachment or ordering a catalog item, review the available APIs to see if one exists that supports the required logic.

Use a Scripted REST API or Scripted SOAP Web Service to build REST or SOAP endpoints, respectively. Scripted REST API and Scripted SOAP Web Service endpoints can execute ServiceNow server-side code when the endpoint is consumed by an external system.

Self-Paced Training: REST Integrations

Create design elements

After the application's data model is created, secured, and populated with data, create the design elements to access that data.

Primary interfaces

The primary way for users to interact with a data model is through forms and lists or through mobile.

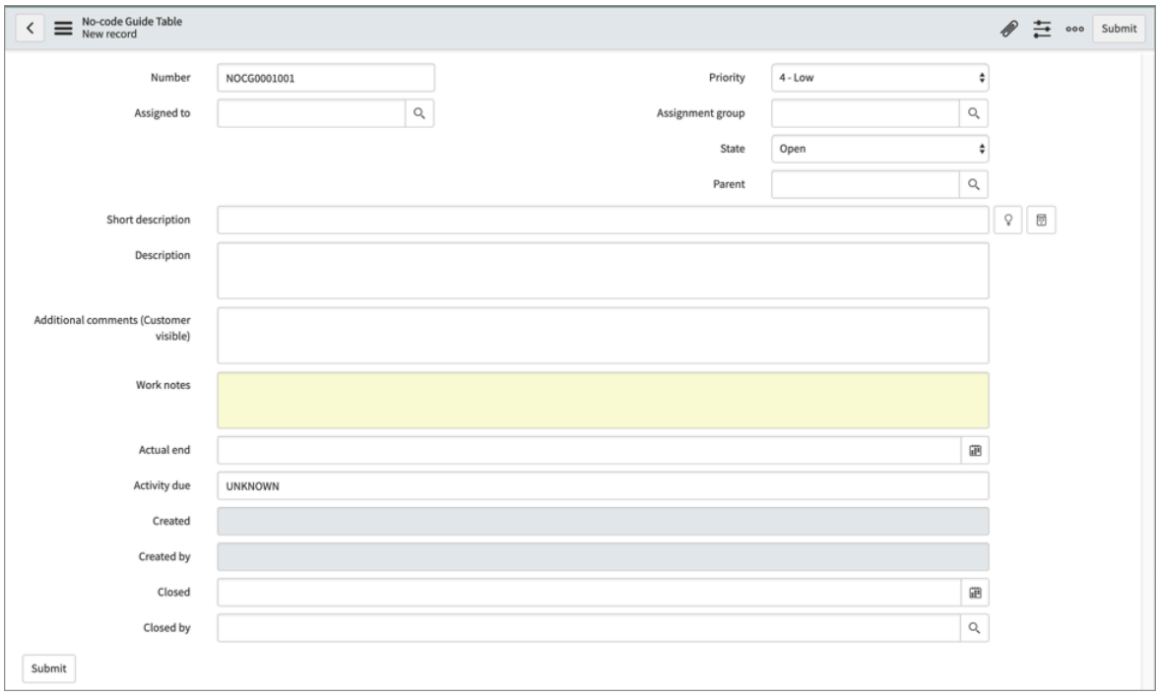
Forms and lists

The standard method of accessing data in ServiceNow is through the default forms and lists. A form displays information from one record in a data table and a list displays a set of records from a table. When configuring forms and lists:

1. Keep the number of fields on a form to a minimum. The more fields on a form, the longer the form takes to load resulting in a poor user experience. Use form views to create different sets of fields for different situations.
2. Use form sections to logically group fields together and to keep users from scrolling. The top section of the form should contain the fields that are always needed or used, while the other form sections contain less frequently utilized fields.
3. Make sure fields appear in the right order. For example, the start date field should always come right before an end date field.
4. Use seven or fewer columns in a default list. Users can add more by personalizing their lists.
5. Avoid using a reference field as the first item in the list view, because it is shown as hyperlinked text. Clicking on the reference field will redirect the user to the referenced record instead of the list record, resulting in a poor user experience.

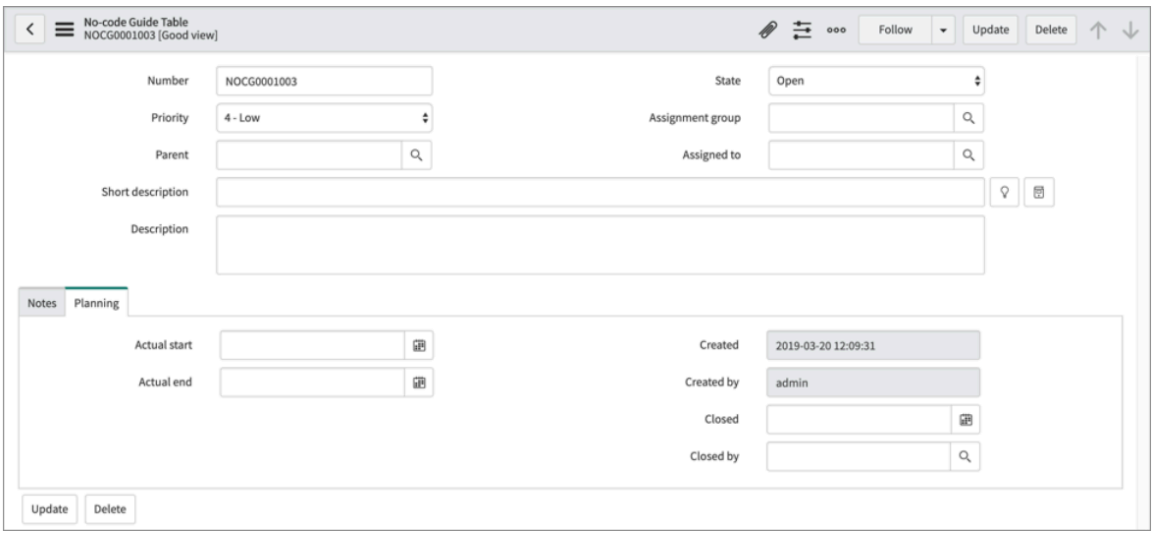
This example shows a poorly designed form.

- The form has no sections. Users need to scroll through the entire form to see all the fields.
- Similar fields are not grouped together. For example, Assignment group and Assigned to are on different sides of the form.



This example shows a well-designed form.

- Fields are grouped together logically.
- The form has been broken into sections for easier viewing and data entry.



Mobile

If users will interact with the application on their mobile devices and will need native iOS or Android functionality, such as geolocation or offline access to application data, use the ServiceNow Agent app. Use Studio to create a mobile application for the application.

Create applets in the mobile application. Applets are tiles within an application. Include a main screen, such as a map or a list, and various details screens, such as an activity stream or related lists. Applets should have focused experiences.

Note:
Individual applets can be secured by role as well as made available in offline mode.

Self-paced training: Mobile Applications

Other resources: Mobile Resources

Self service

Your application may need a way for end users to be able to access your data model, so there are self-service options available.

Service Portal

If the application has Requestor or Self-Service users, use Service Portal to provide a friendly web experience.

To give self-service users the ability to easily create application records from the Service Portal, create a record producer. A record producer can provide a better end-user experience than a regular form. Talk to your ServiceNow Administrator about the appropriate catalog and categorization to make the record producer accessible through the Service Portal.

Alternatively, create a Service Portal for your app if the following is true:

The application needs different branding, navigation, or user experience than an organization's current Service Portal.

OR

The organization does not have an existing Service Portal.

AND

The application needs more functionality than the default portals provide.

AND

The application requires a more customized user experience than the default forms and lists can provide.

OR

The application needs more control over branding and themes than the default interface provides.

Tip:
Do not try to reuse any existing service portal pages in an application. Create new pages and then reuse components in your pages, such as widgets and headers.

Widgets

Widgets are what define the portal content. The base system widgets provided with Service Portal can be used, or developers can build custom widgets to fit business needs.

Considerations for creating custom Service Portal widgets for an application:

- Start from an existing widget instead of creating a widget from scratch. To protect existing widgets from accidental modifications, all baseline widgets are read-only.
- When developing a widget, use the preview pane to quickly test the widget's behavior. Always test the widget on a portal page before releasing a widget to production.
- Use third-party debugging tools when debugging browser-based applications. For example, the ng-inspector Chrome extension for Angular JS.
- Avoid the use of $rootScope.$broadcast(). Instead, use $rootScope.$emit() to publish an event to the rootscope.
- Use widget options to make widgets more easily reusable. The widget option schema defines the user-configurable fields.
- For field types not supported in the option schema, create an extension table to store a custom widget option schema.
- Make use of Angular Providers, which are reusable components that can be injected into multiple widgets. To ensure quick loading widgets and a high performing portal, create Angular Providers instead of overloading your client controllers with persistent data and additional logic. With Angular Providers, you can maintain data for the lifetime of your Service Portal and reuse components and data objects across multiple widgets.

For more information, see Service Portal training.

Virtual Agent as an application design element

Consider Virtual Agent, ServiceNow's conversational bot platform. Use ServiceNow Virtual Agent (VA) to build and design bot conversations to help users quickly obtain information, make decisions, and perform common work tasks.

Some considerations when building VA conversation topics:

- If adding VA to a portal, create topics that focus on particular use cases rather than a broad topic.
- Read topics out loud to keep the topic from sounding too robotic or being too verbose.
- Do not try to design the whole interaction at once. Instead take an iterative approach when building the bot and test the conversation frequently.

For more information, see Virtual Agent.

Notifications

After the data model is defined and users are able to interact with the application, determine how the application should communicate with users. Configure notifications to alert users to important application related events, share application information in the knowledge base, and add translations to allow users to interact with the application in their native language.

The ServiceNow AI Platform can notify users by email, SMS text message, or push notifications.

Triggers: Event Driven Notifications vs Table Updates

Configure notifications to trigger based on updates to a table or based on an event. Use event-based triggering when triggering requirements cannot be easily implemented in the notification conditions or when the notification is triggered from a Workflow. Creating specific events also enables easier notification debugging.

Use the Notification activity in a workflow or the Send an email action in Flow Designer for simple notification use cases. For complex or critical notifications, trigger an event from a workflow or a Flow Designer flow and configure a notification to fire off that event.

Notification Content

ServiceNow notifications support static and dynamic content using email templates, email scripts, and notification variables.

Notification variables add dynamic information to the body of a notification, such as field values from the base record. The variables also support dot-walking, which allows field values from any related records to be included in the content without scripting.

For example, use the URI_REF variable to point to the record that originated the email.

Use email scripts or dot-walk from the base record to include dynamic content that is not available in the record. Use the mail script API to set notification details, such as the recipient and sender addresses, etc.

Note:
Scripts entered in the notification body using <mail_script> tags may not always work correctly. Always create email scripts at System Notifications > Email > Notification Email Script and include them in the notification body with ${mail_script:script_name}.

Create email templates for content used in multiple notifications. Adding the content to an email template enables administrators to create reusable content for the subject line and message body of email notifications.

Configure recipients

ServiceNow emails can be sent to users, groups, or individual email addresses. When sending to groups, check the **Include members** field on the group record for the notification to be sent to all members of the group in addition to the group email.

Subscription-based notifications - Select the **Subscribable** option on the notification to allow recipients to pick and choose the emails they want to receive.

For a subscription-based notification, the Mandatory option can be set to true for the recipients to receive the notification regardless of their individual preferences. Optionally, configure unsubscribe links in the outgoing email to allow users to remove themselves from the notification.

ServiceNow uses email watermarks to correctly process user responses to notifications. Email notifications automatically include watermarks unless the **Omit watermark** option is selected in the notification. Omit watermarks only when no email response is expected from the recipients.

The Troubleshooting Outbound Email knowledge base article provides troubleshooting steps for the most common notification issues. Log in to the HI portal (https://hi.service-now.com) to access the article.

Translations

When using one of the Internationalization plugins, most of the fields in the instance are automatically translated. However, customizations are not translated automatically, and need to be translated by hand. In this case, it is best to locate the individual untranslated strings, and insert those translations manually.

To implement a new language, activate the plugin for the new language. Then export the existing values from any of the Translation tables.

For example, export all **sys_choice** records. Provide the export to a translator to provide translations for the **Label** field in the provided file. The **Value** field should not change. Update the

Language value to the new language. Then, upload the translated values back to the **sys_choice** table.

> **Note:**
> To insert values, make sure you are setting all the necessary fields; table, element, language, label, value, and sequence. Set the dependent value and hint where applicable. This logic is the same for all translation tables.

For more information, see Language Internationalization Support

Overview of reporting and analytics for developers

Most applications that you create have some level of reporting requirements. Reports should be actionable to drive change.

"Reporting" generally refers to showing the data inside facts tables like Incident [incident]. You can also create key performance indicators (KPIs) to track changes in this data over time, through the Performance Analytics application.

Currently the ServiceNow AI Platform® is in transition between two user interfaces for showing this information:

- The older Core UI technology. This UI includes the Reporting application, which only shows data directly from tables, and PA Widgets, which show data from Performance Analytics indicators. Both reports and PA widgets can be placed on Core UI responsive dashboards. For more information, see Reporting, dashboards, and Performance Analytics in the Core UI.
- The newer Platform Analytics technology. This UI includes Data Visualizations, which let you report on data from any source. These visualizations can be placed on Platform Analytics dashboards, along with Platform Analytics filters. For more information, see Platform Analytics experience.

  All Platform Analytics objects are rooted in the Next Experience UI Framework and are available to developers. However, a non-developer can still build their own objects through the Platform Analytics experience, without using UI Builder. For more information, see Platform Analytics experience.

> **Note:**
> Although "reporting" is a general term, this documentation usually uses "report" and "reporting" to refer to the Core UI Reporting application and "visualization" or "visualize" to refer to Platform Analytics data visualizations, for disambiguation.

Follow these guidelines when building reports or data visualizations:

- Creating reports or visualizations on large tables can negatively impact performance. Be sure to filter by a date range or other limiting criteria rather than showing all records on the table.
- Grouping by fields that contain many possible values can negatively impact performance.
- If loading a report or visualization gives a Long running transaction timer message, consider adding more data filters to reduce the load time.
- If you need to export a report, data visualization, or dashboard on a regular basis, schedule the export and email.

In Platform Analytics, you have several possibilities for showing multiple data visualizations on one page:

- Create the data visualizations and the dashboards entirely inside the Platform Analytics experience. This approach does not require a developer role or special technical knowledge, and probably should be explored before you try a more complex solution.
- Create data visualization components in a generic UI Builder page, along with filters and other components. This approach gives you the most freedom as a developer, but also requires the most configuration.
- Create the data visualizations and the dashboards inside the Platform Analytics experience, but then place the dashboards inside UI Builder pages using the Dashboard page template. This approach lets you use the convenience of the Platform Analytics experience to create dashboards, data visualizations, and filters in your own experiences/workspaces. It also gives you the freedom to customize the page configuration partially. For more information, see Creating Platform Analytics pages in your own workspace.
- Create a technical dashboard and populate it inside UI Builder. This approach is almost the same as creating your own UI Builder page from scratch, but the dashboard is available in the dashboard library, has dashboard details, and can be shared like other dashboards. For more information, see Technical dashboards.

Build form and business logic

The next step in designing an application is to build logic. Logic includes form logic (what people can and cannot see/use on a form) and business logic (rules that govern what happens to data when it is entered).

Scripting and modifications

Before writing any code, be aware of the impact on upgrades and the adoption of new ServiceNow features. Particular care should be taken when modifying baseline artifacts and processes.

Consider the following before scripting:

- Assess the requirement. Is the logic critical to the functioning of the app?
- Determine if ServiceNow can be configured to fulfill the requirement without code.
- Leverage options, such as Flow Designer, Virtual Agent, and UI Policies to take advantage of platform capabilities without writing code.
- Low and no-code approaches to logic are easier to debug and upgrade.

Examples of when scripting is appropriate:

- Building Flow Designer actions
- Creating a Scripted REST API
- Creating logic for scoped applications in Script Includes
- Customizing and creating widgets for Service Portal

Evaluate the business requirement and consider a no-code route before using a scripted solution.

Be aware of ServiceNow enhancements. For example, in the orlando release, Virtual Agent conversations have more no-code options than London. Read release notes and other publications. Get certified and stay current with your certifications.

To better understand when to customize, review the Innovate at Scale Success Playbook on the Customer Success Center.

Modifying default behavior

In the past, one of the strategies used was to copy the artifact to update and to deactivate the original. The copy/deactivate approach is no longer recommended due to the following issues:

- Developers cannot tell if a deactivated artifact was upgraded without research.
- Two files, the original and the copy, need to be maintained. Maintenance doubles each time a customization is made.
- With each release, the customized record becomes older.
  - Customers do not receive the enhancements included in a new release.
  - A new release may rely on the original record being updated.
  - Developers may make more changes to compensate for the original record being inactive.

A script where only the **Active** flag is changed will be updated, but the script does not appear on the skipped list. With the copy and deactivate strategy, a developer has less visibility into customizations and cannot easily assess or revert to the baseline version.

Rather than copying and deactivating the original artifact, edit the artifact directly. The ServiceNow Upgrade Engine will add the latest version to the version history and report that the artifact was skipped. Developers can see a new version is available with the upgrade.

Form logic

Controlling what users see when they visit a form can increase productivity and responsiveness. For example, users should only see fields that are useful to them. Users may only need to see certain fields based on what is configured on the form. Apply form logic to control what is visible, read-only, and mandatory on a form.

The following question will help direct you to the right decision for when to control user access to information: Is this a suggestion or enforcement? A suggestion makes the form easier to complete whereas enforcement forces the user to do something in order to complete the form.

UI Policies are useful for conditional suggestionslike showing and hiding fields or adding field messages based on another field's value, while Data Policies and Business Rules are better suited for doing conditional enforcementlike making a field mandatory.

The best user experience is to utilize both suggestion and enforcement together.

For more information, see the UI Policy article in the Client-side Scripting Module.

Build UI Policies and Data Policies to handle client-side activities before scripting any client-side logic. Use of Client Scripts to validate user input and provide feedback while the user is completing the form.

Some general practices for client scripting are:

- Optimize for performance by using asynchronous GlideAjaxover client-side GlideRecordor multiple getReference()calls.
- Keep the isLoadingcheck in onChangeclient scripts.
- Keep the newValuecheck and add a newValue != oldValuecheck.
- Use all client-side scripts possible before making a server call with GlideAjax. Server roundtrips can impact performance.

Some client scripting practices to avoid are:

- Global Client Scripts or Global UI Scripts: Global scripts will run on every page load and introduce browser load delay.
- DOM Manipulation: Using document object model manipulation against default UI elements introduces upgrade risk and maintainability issues. The exception is using DOM manipulation against the DOM in pages authored in the same scoped application, like UI Pages or Service Portal widgets.

Business rules and script includes

Business rules are server-side actions that can be run during CRUD (Create, Read, Update, Delete) operations on instance records.

Some good practices when using Business Rules are:

- Keep Business Rules small and specific.
- Avoid modifying base system Business Rules.
- Use Script Includes instead of global Business Rules.
- Use scripting only when necessary.
- Store reusable script logic in a script include.
- Use queries to limit records processed within a Business rule.
- Avoid client-callable Business Rules to improve efficiency when running client scripts.
- Always use a condition with Business Rules to control when the Business Rule runs. Running Business Rules with conditions can also aid in debugging. Business Rules rarely run with no conditions.

Business Rules can be configured to run before or after a database operation. They can also be configured to run asynchronously and also before displaying a form or executing a query.

Value	Runs	When to Use	Example
Before	Synchronously before the database operation	Set or update values on the current object as part of the save operation. Validate and abort execution if required.	A developer wants to set the state of the current record based on another input in that record.
After	Synchronously after the database operation	Trigger events and notifications after the database update to access the previous object or to make something occur in sequence. Update related records other than the base table being updated to access the previous object or to make something occur in sequence.	A developer wants to cascade values from the current record down to child records.
Async	Asynchronously executed as a separate process after the database operation is completed	The process triggered by the rule may take a while to run. When the user who triggered the operation does not need the output right away. Trigger events, notifications, or related record updates when access to the previous values of the record or a specific sequence of actions is not required.	A developer needs to trigger an external process that may take a while or update a large number of records.
Display	Executed every time the	Used to make server-side objects available to client-side scripts.	A developer wants to write information

Value	Runs	When to Use	Example
	corresponding form is displayed		about a user associated with the current record to the g_scratchpad object to use in a client-side script.

Note:
current.update()should not be used in any Business Rules. Using current.update()triggers an additional database operation, which could cause duplicate notifications, recursive loops, etc.

Use Script Includes to store JavaScript functions and classes for use by server scripts. Each Script Include defines either an object class or a function that can be reused among any server-side scripts. For more information, see Script includes.

Store any code that might need to be used elsewhere in a Script Include. Call the Script Include from a Business Rule, UI Action, workflow script, Scripted REST API, etc. Instead of calling a Business Rule from a UI Action or a UI Action from a Scripted REST API, put the code in a Script Include and call the Script Include from both places.

Keeping functions in a Script Include allows testing of the function before deploying the function in other scripted areas, thus reducing overall development and testing time.

For more information, see Classic Business rules.

Flow Designer

Flow Designer is a ServiceNow AI Platform feature that enables rich process automation capabilities in a consolidated design environment. Flow Designer enables process owners to use natural language to automate approvals, tasks, notifications, and record operations without having to code.

Flow Designer and IntegrationHub

For any new process flow requirements, ServiceNow recommends using Flow Designer over the legacy workflow for almost all circumstances.

Flow Designer and Business Rules

You should use Flow Designer instead of Business Rules unless:

- Business logic needs to run in a specific sequence with other Business Rules. For example, new business logic needs to run after one Business Rule but before another.
- Logic needs to execute immediately before or after writing to the database in the same thread.
- The logic only calls a Script Include.

When designing a flow, follow these design principles:

- Single Purpose: Each flow should have a singular goal.
- Reusability: Design with reusable sub-flows in mind (approval is a great example).
- Clarity: The language and layout of a flow should make each action's purpose clear.

Start with a white board design of a business flow. Then build the flow action by action to align with the process. More than one flow may be required for a for a single process to keep to the design principles.

Use the following practices when working with Flow Designer:

- Use records, not SysIDs. Provide a guided experience with inline documentation.
- Learn how to use template objects to work with both static and dynamic inputs.
- Avoid passing around blobs of data unless absolutely necessary.
- Only pass information to a flow that the flow is going to use.

Use the following practices when working with Flow Designer Actions:

- Always create actions under the scope of the application's spoke, if applicable.
- Set access to Accessible from all scopes in actions to be able to reuse actions across other apps and scopes in the future.
- Set **Protection** to Read-only to avoid any unwanted edits to the actions by users.
- Make sure inputs have a specific type.
- Ensure that **Mandatory** is selected where required.
- If using a **choice** input type, use a default value.

Use the following practices when working with IntegrationHub:

- Create one spoke per integration system. Only put actions for a single system in a spoke.
- When creating the scoped app for the spoke, use a version naming convention that makes sense.
- Use a connection alias instead of an inline connection. The Base URL will be automatically extracted.
- Use connection attributes under the Alias to pass the version in a REST step giving future flexibility for versioning in the resource path.
- Use Save as Attachment to save the content in the response instead of creating another step to save the data.
- If the Alias is dynamic, make Alias one of the inputs and use the data pill to provide the Alias.

Use the following practices in Flow Designer and IntegrationHub for Error Handling:

- Create a Script Include to handle errors.
- Write short and understandable error messages.
- Incorporate all of the possible error messages the API returns.
- Ensure that the outputs from the integration step are validated before using them.
- Fail Early: If the inputs are not available, do not call the integration.

Self-Paced Training: Flow Designer

Self-Paced Training: IntegrationHub

Validate app functionality

As the application is built, validate that it works as expected.

Unit testing

Unit/Story testing ensures requirements specified in a story are validated before closing the story. A Story/Unit is a smallest testable portion of system or application that can be configured and executed.

When the configuration of the story is complete, developers need to unit test the features not only in the context of that particular story, but also other related stories that share components with the current story.

As a good practice, developers need to assign the story to process owner or designated stakeholder to validate the story configuration meets expected outcomes before closing the story.

ServiceNow's Automated Test Framework (ATF) is primarily meant for automating functional testing of applications but in few cases can be used to automate unit testing of configurations that involve Script Includes and Business Rules.

System testing

System testing is performed on a complete system when development is completed. Test the overall interaction of components and integrations with other applications within scope. System testing is performed by the QA/Testing team, but developers need to collaborate with the QA team and process owners to ensure test cases provide comprehensive coverage. Developers will be responsible for remediation of issues found during System Testing.

Automated Test Framework

Automated Test Framework (ATF) should be leveraged for automating functional system testing of ServiceNow applications to reduce testing time and costs and make testing repeatable and UI independent. When creating test cases, follow these guidelines.

When creating tests:

- Use parameterized testing to avoid duplicate test cases.
- Follow a Test naming standard.
  - `<app initial>: <functionality that is being tested>`
  - CSM: Resolve case
- Describe each test's use case in its description. For example: Sample that tests use case.
- Develop tests on a Development instance and promote/run the test on a Test instance.
- Clones wipe out tests. Use one of these options to preserve tests:
  - Bundle tests in a scoped app and upload the app to GIT.
  - Save tests before the clone.
  - Promote tests to prod instance, but DO NOT EXECUTE THE TESTS IN PROD.
- Create self-contained tests.
- Create new server-side or REST test steps any test steps are missing. For example: Email body verification.
- Use server-side test step whenever possible and when screenshots are not important.
- Start with the Impersonatestep.
- Be aware of browser throttling.
- Use the Test Logs and Test Transactions to troubleshoot test errors.

When creating test suites:

- Follow a test suite naming standard. For example: ITSM INT: Use cases.
- Describe the suite.
  - Test suite description: "This is a sample test suite to test plugin/application".
  - Provide any additional information possible in the description.
- Organize test suites by feature areas.

User acceptance testing

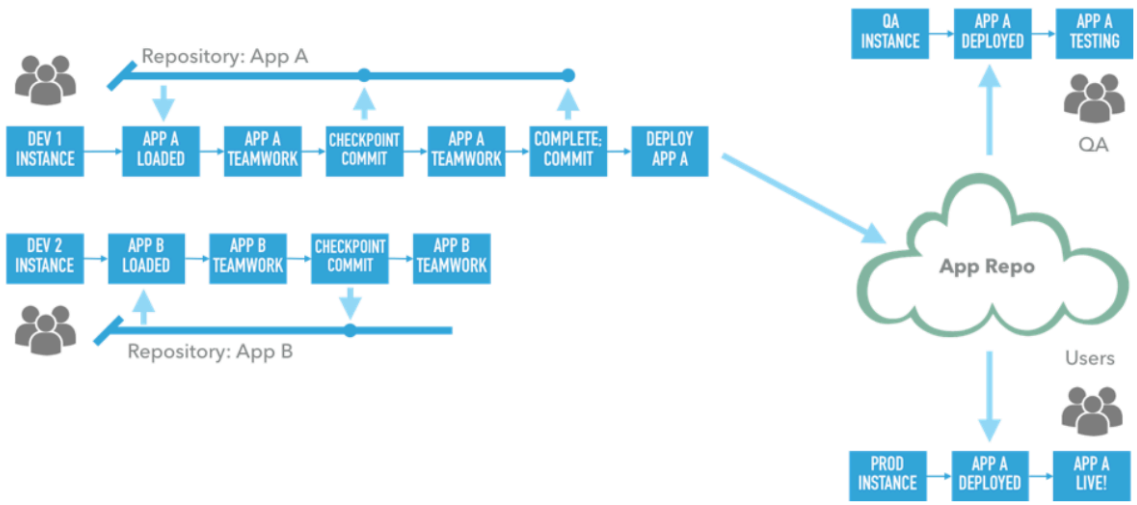
User Acceptance Testing (UAT) is a test conducted to evaluate the application's compliance with the business requirements and assess whether the application is acceptable for delivery. Users, customers, or other authorized stakeholders perform acceptance testing. Developers will be responsible for remediation of issues found during System Testing.

Deploy your app

Once the application is built and validated, the application needs to be moved to the production environment. Applications can be moved through an application repository or by using Update Sets. Applications should be deployed to test environments prior to moving to production.

Application Repository (App Repo)

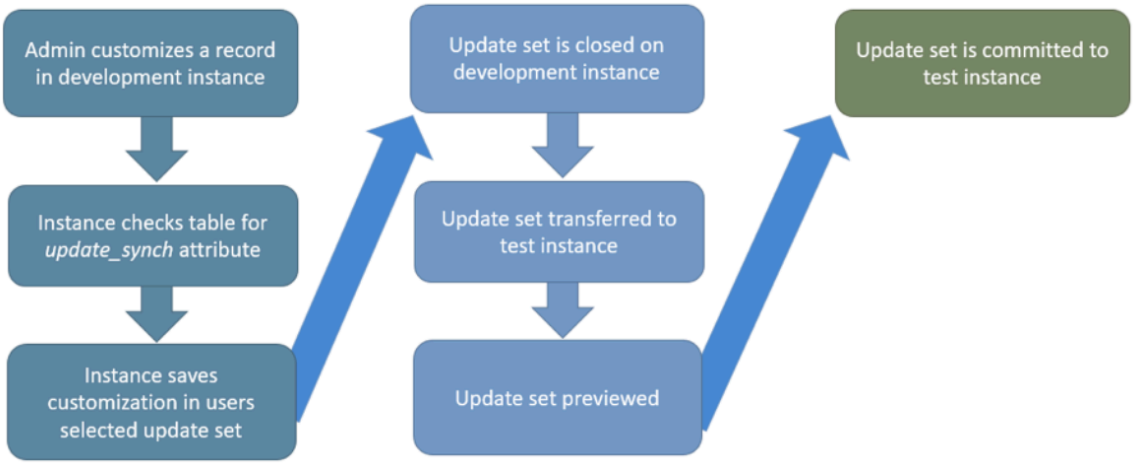
Publishing an application to the App Repo makes this version of the application available to all of an organization's ServiceNow instances. Use the App Repo to deploy an application to QA / Test instances (for testing) and finally to Production (Prod) instances.



For more information, see Publishing an application to the application repo, Install an application.

Update Sets

If the application repository cannot be used to deploy applications, use Update Sets instead. The diagram shows the best practice lifecycle of an Update Set to deploy a customization from the development instance to the test instance.



Practices that lead to a quality development and release process:

- Always move customizations from the bottom of the stack up.
  - Ensures down-stack instances match up-stack instances.
  - Customizations introduced mid-stack can be overwritten by future pushes from down-stack.
  - Common scenarios include:
    - Fixes need in test or prod – always push them from dev up
    - Common Prod admin customization, such as choice lists – always push updates from dev up
- Always review updates contained in an Update Set before transferring.
  - Look for updates associated with other development efforts and updates associated with testing.
  - Watch for system properties and integration end-points changes. Ex: pushing sys_properties change that directs all email to test email account
  - Move updates to a “scrap” Update Set rather than deleting the update.
- Always test after pushing to ensure that all desired customizations are captured and applied as expected.
- In situations with multiple, parallel releases, ensure communication and coordination between the development teams.
- Avoid experimenting on the development instance, because customizations can be captured accidentally and migrated by other team members.
- Do not capture development in the Default Update Set.

List all user Story numbers with a short description in the **Description** field of an Update Set. Include all manual steps that are required to deploy the Update Set.

Some typical examples of manual steps that are needed for a deployment that are not captured in an Update Set:

- Plugin activation.
- Transfer of tables that are not tracked in the Update Set (typically, starting with “x_” or “u_”).
- Creation of database indexes on the tables. The index creation is not tracked via Update Set and needs to be done manually.

Update Set management

Be sure the correct Update Set is selected when working on a story or a defect and check the records in the Update Set daily.

Do not manually move sys_update_xml records between Update Sets. The only exception is to move a record to the Default Update Set.

Update Sets capture configuration information but not task or process data. For example, Update Sets track Service Catalog item definitions and related configuration data like variables and variable choices. However, the orders (requests, items, catalog tasks) placed in testing are not tracked by Update Sets.

Be aware of Update Set DOs and DON'Ts:

- To remove a specific sys_update_xml record from the current Update Set, move the record to the Default Update Set and populate the **sys_update_set.comments** field of the record with the reason for moving the record to the Default Update Set.
- Never move customization records from one Update Set to another Update Set.
- Never delete an Update Set unless the Update Set has been merged successfully into a new Update Set.
- Always use data extracts or Import Sets to move data from one instance to other (and not Update Sets).

Update Set batching

Batch Update Sets to preview and commit Update Sets in bulk.

Dealing with multiple Update Sets can lead to problems, including committing Update Sets in the wrong order or inadvertently leaving out one or more sets. Avoid these problems by grouping completed Update Sets into a batch.

The system organizes Update Set batches into a hierarchy. One Update Set can act as the parent for multiple child Update Sets. A given set can be both a child and parent, enabling multiple-level hierarchies. One Update Set at the top level of the hierarchy acts as the base Update Set.

Previewing or committing the base Update Set previews or commits the entire batch. The system determines the processing order, and checks for collisions based on the dates the changes were recorded and on their sequential ancestry. Their ancestries are the specific instances in which the changes in the Update Sets took place.

Update set batching can be applied to releases, where an empty parent Update Set is created for the release, and actual Update Sets are included in the release as children.

Advantages of using Update Set batching are:

- Individual Update Sets can be removed from the release at the last moment.
- Batching is similar to merging, except batching allows updates to be removed.
- Batch Update Sets are easy to deploy. Only the Parent Update Set needs to be processed.

For more information, see System update sets.

What to do next

Now that the app is deployed, think about how to improve and enhance it. Here are some suggestions for determining where to go next:

- The people using the application day-to-day will be the best source of feedback. Talk to them about what new features or changes they would like to see.
- Determine if additional related process flows can be automated through Flow Designer.
- Determine if new IntegrationHub spokes can be leveraged for new integrations.

Builder library

The ServiceNow AI Platform provides a robust set of builder tools that enable you to quickly develop applications for your organization.

The following builder tools enable you to quickly design and build user experiences, interface components, data models, and more when developing applications on the ServiceNow AI Platform.

Builder tools

Tool	Description	Use case(s)
Exploring decision tables	**Configure application decision logic** Use decision tables to help resolve a complex decision that depends on multiple factors. Decision Builder enables developers to decouple decision logic from their code by creating and maintaining decision rules.	Determine a customer discount rate for a product given two inputs (number of units ordered, and whether or not it was paid for in cash).
Flow Template Builder	**Configure a standard template for specific app workflows** Flow Templates guide flow authors to create flows for common use cases. Use this builder to define flows, actions, and flow template variables.	Create a flow template to use as the basis for all request approvals within an IT application.
Integration with Performance Analytics	**Optimize processes through integration with Performance Analytics** Use Process Optimization with Performance Analytics to perform data extraction from an indicator and analyze processes associated with the KPIs, such as time to resolve.	Visualize the execution of processes or a customer journey, show insight into adoption, activity transitions, performance, and user experiences or identify bottlenecks and see their impact on your key performance indicators, enabling you to resolve root-cause issues.
Mobile App Builder	**Configure Mobile apps** The Mobile App Builder is a configuration tool to build and manage screens and records that make up workflows within	Streamline mobile configurations by organizing records in a hierarchical layout for quick access and reference, supporting hierarchical tree navigation, and enabling users to visualize how records relate to

Builder tools (continued)

Tool	Description	Use case(s)
	ServiceNow mobile apps. The organizational layout and navigation options in the Mobile App Builder facilitate a faster and more intuitive creation of ServiceNow mobile applications.	each other in the overall mobile app configuration.
Mobile Card Builder	**Configure and manage card templates to display information for mobile interfaces** Use Mobile Card Builder to create or modify card templates used in the ServiceNow mobile applications for iOS and Android. The ServiceNow mobile interface uses cards to display information about records on your instance.	Create cards for launcher screen record sections, calendar screens, form screens, Genius search results, list screens, and map screens.
NLU Workbench	**Configure Natural Language Understanding (NLU) models to interpret user-expressed intent** ServiceNow® Natural Language Understanding (NLU) provides an NLU Workbench and an NLU inference service that you can use to enable the system to learn and respond to human-expressed intent. By entering natural language examples into the system, you help it understand word meanings and contexts so it can infer user or system actions. NLU workbench is where you create, manage, and test NLU models. Use NLU models to apply ServiceNow® Natural Language Understanding on your instances. Create, manage, test, and publish NLU models with the NLU Workbench.	Configure a Virtual Agent Designer conversation flow to consume an NLU model so that agent chatbots can better understand user statements in the conversation.
Platform analytics for workspaces	**Configure performance analytics that capture historical trends in data through snapshots** Integrate data visualization and analytical functionality into the workspace experience. Subscribe to notifications about behavioral process changes. Explore KPIs and	Configure the ability for users to ask natural language questions and be shown the relevant analytics in a visualization, view any Next Experience dashboard that are owned by a user or shared with a user, create dashboards and data visualizations from workspace runtime, or drill down into KPIs,

Builder tools (continued)

Tool	Description	Use case(s)
	get answers and insights on Platform Analytics.	apply filters to slice data, or apply aggregations and statistical tools.
Reports	**Build and distribute reports that show the current state of instance data** ServiceNow® Reporting enables you to create and distribute reports that show the current state of instance data. Reporting functionality is available by default for all tables, except for system tables.	Show the number of open incidents at an organization that also displays the priority of each incident at a current point in time. A variety of different types of reports could be configured to show this information, including bar, pie, time series, multidimensional, scores, statistical, etc.
Service Portal Designer	**Create a self-service portal for your users** Service Portal enables you to build a mobile-friendly self-service experience for your users. It interacts with parts of the ServiceNow AI Platform, so users can access specific ServiceNow AI Platform features using Service Portal. This tool is used to create service portal pages to organize content, ensure responsive mobile optimization, and design meaningful portal user experiences for your customers. A page houses containers and rows, which then contain widgets. By manipulating the layout of the page, and the widgets within it, you can construct your desired user experience. Widgets are what define the content in your portal. You can use the base system widgets provided with Service Portal, clone and modify widgets, or develop custom widgets to fit your own needs.	Configure a mobile-friendly experience for a company's employees to engage with IT and request services like new emails, new employee computer setup, etc.
Table Builder	**Create application tables and table-related forms and configure display logic** Table Builder enables you to create, design, and administer tables and forms visually from a single user interface.	Create an online food order form in an food delivery application for customers. The form might capture the food ordered and related delivery options. The form would store and capture the information entered in a designated application data table that you configure for this purpose.

Builder tools (continued)

Tool	Description	Use case(s)
Themes in Theme Builder	**Customize themes and branding** Theme Builder enables you to tailor the visual experience for your users, helping to update the look and feel to be more like your brand. Next Experience extends the theming model used in UI Builder.	Configure a dark theme for your company's brand that makes use of a primary color and secondary color, specifies what color to use for screen elements like divider lines, and contains a new brand logo.
UI Builder	**Design application user Interfaces** UI Builder is a web user interface builder. Use UI Builder to build pages for workspaces or custom web experiences (generated by App Engine Studio) that use Next Experience Components and custom web components..	Create a new standard record page variant to show specific incident records, edit the standard record page to show a list of problems for a user, or build a landing page for a Next Experience app you are designing.
Virtual Agent Designer	**Design virtual agent conversation blueprints** The Virtual Agent Designer is a diagram tool for creating and managing topics, which are blueprints for conversations between a virtual agent and user. You can design topics that help your users resolve common work issues or guide them through self-service tasks.	Create and configure a conversation flow that enables end users to see the status of incidents they submitted and leave a comment in the incident.
Workspace Builder	**Create custom workspaces for your application** Workspace Builder is a streamlined, no-code environment that enables you to create a custom workspace from within App Engine Studio (AES) quickly and efficiently.	Configure a Service Desk workspace that will handle request and fulfillment of IT services within your organization.

Create custom components using ServiceNow CLI

Develop custom components using the Next Experience UI Framework and the ui-component extension.

Components are reusable building blocks that you use to create a custom user interface. The Next Experience Design System comes with a set of customizable components that you can drag into your custom UI. Develop your own components if you can't find what you need in the Next Experience Design System library.

To see the Next Experience Design System library, visit the ServiceNow® Developer Site.

Benefits of creating custom components

Developing custom components lets you:

- Personalize a UI according to your agent, customer, and company needs.
- Make your employees more effective and reduce context switching with quick access to important data and information.
- Accommodate your company's unique omni-channel environment using APIs to consolidate your data.

For example, you might want to create a component that displays the cases associated with an SLA, or that tracks the active chats in a particular queue. You can use the Next Experience UI Framework and the ui-component extension to develop the component you need, and access data from your platform using the Http Effect API. You can also query platform data using GraphQL by creating a custom schema. For more information, see Scripted GraphQL.

What to know before you begin

Before you start designing and building your component, make sure you have:

- Some general knowledge of web component concepts, development, and design.
- JavaScript knowledge to define the component behavior.
- Knowledge of Node Package Manager (npm).
- The most recent version of Node.js installed on your local machine. For more information, see, Node.js.
- The ServiceNow CLI installed on your machine.

Next Experience UI Framework

The Next Experience UI Framework is a JavaScript framework that lets you extend your apps and build web components that are reusable throughout your applications. Using the Next Experience UI Framework lets you:

- Create a single component to use in multiple places across your applications.
- Encapsulate the component's scope to prevent conflicts with other code.
- Add properties, slots, and actions to your component, allowing users to customize the component every time they use it in a Workspace.

For more information, see the ServiceNow® Developer Site.

ui-component extension and development flow

The ui-component extension is an extension to the ServiceNow CLI that lets you develop custom components using the Next Experience UI Framework. With the ui-component extension, you can:

- Create the set of files required to develop a component, or project scaffolding.
- Start a local development server to test your component.
- Build a component project and deploy it to a ServiceNow instance.

Application scope

When you deploy a Next Experience UI Framework component, it deploys into a scoped application on the instance. You can provide an application scope for the component to use as a namespace identifier. Use the namespace identifier guidelines for application development on the instance. For more information, see Application scope.

When reserving an application scope, follow these requirements:

- Maximum: 18 characters.
- Case: snake case.
- Format: `x_customerprefix_componentname`, where:
  - `customerprefix` is the value in the *`glide.appcreator.company.code`* system property on your instance.
  - `componentname` is the value provided in the component's `name` parameter when you created the project.

If you do not provide an application scope when creating your component project, the Now CLI creates one for you.

Alternatively, you can add a value to the `scopeName` parameter in the `now-ui.json` file. For more information, see Change a component's application scope.

Reference guide

To see the Now CLI reference guide, visit the Developer Site.

Set up your environment

Prepare your local environment by installing the Node.js JavaScript runtime environment, Node package manager (npm), and the ui-component extension. These tools enable you to develop and build components locally and deploy them to your instance.

Before you begin

The ui-component extension only supports these operating systems.

- Windows 10
- MacOS Yosemite and later
- Linux CentOS v7.5

Role required: none

Procedure

1. Install Node.js version 12.16.1 or greater.
   npm automatically installs as part of the Node.js package. Node.js is a JavaScript runtime environment that enables you to execute JavaScript locally. npm is the default JavaScript package manager for Node.js. For more information, see Node.js and npmjs.com.

2. Add the ui-component extension to the ServiceNow CLI.

   a. Open your system's command-line tool and execute this command.

   ```
   $ snc extension add --name ui-component
   ```

3. Verify the installation by running an extension command with the `--help` argument.

   ```
   $ snc ui-component --help
   ```

   The CLI displays a list of available commands. See Get help with ServiceNow CLI.

What to do next

Set up your project.

Related topics

Create custom components using ServiceNow CLI

ServiceNow CLI

Set up your project

Create the component project and the set of files required to develop a component. You can connect to your instance and create an application scope for your component, or you can reserve a scope to verify later.

Before you begin

Set up your environment.

Procedure

1. Create a folder for your project and point your system's command-line tool to the folder.
2. Create the component project and all the files required to build a component.

   a. From the folder you created, execute this command.

   ```
   $ snc ui-component project [--name name --description
    description --scope scope --offline]
   ```

   Pass in values for these arguments.

Name	Description
name	Required. The project name. Must be a valid and unique npm package name.
description	The project description to be available in the npm registry and the plugins list in your instance.
scope	Suggested application scope to assign to this project and its components. If provided, the instance validates the name. Use the namespace identifier guidelines for application development on the instance. For more information, see Application scope. Maximum: 18 characters. Case: snake case.

Name	Description
	Default: `x_customerprefix_componentname`, where: ▪ `customerprefix` is the value in the *`glide.appcreator.company.code`* system property on your instance. ▪ `componentname` is the value provided in the component's `name` parameter when you created the project. Alternatively, you can add a value to the `scopeName` parameter in the `now-ui.json` file. For more information, see Change a component's application scope.
offline	When true, creates and scaffolds a component while disconnected from your instance. Skips validation of the given scope name. Default: `false`.

Example

```
$ snc ui-component project --name @myorg/movie-quotes
 --description 'A web component that prints movie quotes.'
```

Example

```
$ snc ui-component project --name @myorg/hello-world --scope
 x_myorg_helloworld --offline
```

3. Run the following command to install the npm dependencies.

```
npm install
```

What to do next

Develop a component.

Related topics

Create custom components using ServiceNow CLI

Use a proxy server with ui-component extension

Use a proxy server with the Now CLI to define a separate data source for your component. When using a proxy server, the instance forwards Now CLI requests to the proxy server.

Before you begin

Set up your environment.

Procedure

1. In your component directory, open the `now-cli.json` file.
2. Update the `proxy` object.

```
{
 development: {
  proxy: {
   headers: {
    Authorization: "Basic <username:password>"
```

```
    },
    origin: "http://myinstance.servicenow.com",
    port: 3000,
    proxies: []
   }
  }
}
```

Update the following parameters with your proxy information.

Parameter	Description
development.proxy.headers	Headers to append to your proxied request.
development.proxy.headers.Authorization	Authentication header to add to the proxied request. Enter your Base 64-encoded username:password here.
development.proxy.origin	Host to open a proxy with (your instance host address).
development.proxy.port	The port the proxy runs on.
development.proxy.proxies	URL glob patterns to pass to the proxy server.

3. Save the file.

What to do next

Develop a component.

Related topics

Create custom components using ServiceNow CLI

Develop a component

Add your component code and test it using a local development server.

Before you begin

- Set up your environment
- Set up your project

About this task

For a tutorial on developing a counter component, visit the Developer Site.

Procedure

1. In your project directory, navigate to `src/<component-name>/index.js`.
   This is the primary code file for a component.
2. Add your component code.

Example

This example shows a sample Hello World component.

```
import {createCustomElement} from '@servicenow/ui-core';
import snabbdom from '@servicenow/ui-renderer-snabbdom';
import styles from './styles.scss';

const view = (state, {updateState}) => {
```

```
 return (
  <div>Hello World!</div>
 );
};

createCustomElement('hello-world', {
 renderer: {type: snabbdom},
 view,
 styles
});
```

To develop a component for Virtual Agent, add specific properties and actions to interact with the Virtual Agent client interface. The properties required depend on the type of component you are creating. For more information, see Develop a component for Virtual Agent for more information.

3. **Optional:** Add inner components.
   An inner component is a component included in another component. For example, a card component might include a separate button component. The button component becomes an inner component that you must import, install, and define in the card component project.

   a. Install the inner component package by executing the install command using your system's command-line tool:

   ```
   npm install @servicenow/<now-inner-button>
   ```

   b. Add an import statement to the top of your `index.js` file:

   ```
   import '@servicenow/<now-inner-button>';
   ```

   c. Use the inner component in your `index.js` file:

   ```
   <now-inner-button label="Click Me" />
   ```

   d. List the inner component in the `components.[component-name].innerComponents` array in the `now-ui.json` configuration file.

   Example

   ```
   {
     "components": {
       "now-chart-timeseries": {
         "innerComponents": [
           "now-inner-button"
         ],
   ```

4. Run the development server command to view your component in a test browser.

   ```
   $ snc ui-component develop [--entry entry --open --port port
    --host host]
   ```

   Pass in values for these arguments.

Name	Description
entry	Path to the test module in your component project.

Name	Description
	Default: `example/index.js`.
open	Opens the default browser and navigates to the test page. Default: false.
port	Port where the development server runs. Default: 8081.
host	Host address to use if you want your local development server to be accessible externally by others. Typically set to 0.0.0.0

Example

```
$ snc ui-component develop --entry example/hello.js --open
 --port 3000
```

What to do next

Deploy a component to an instance.

Related topics

Create custom components using ServiceNow CLI

Develop a component for Virtual Agent

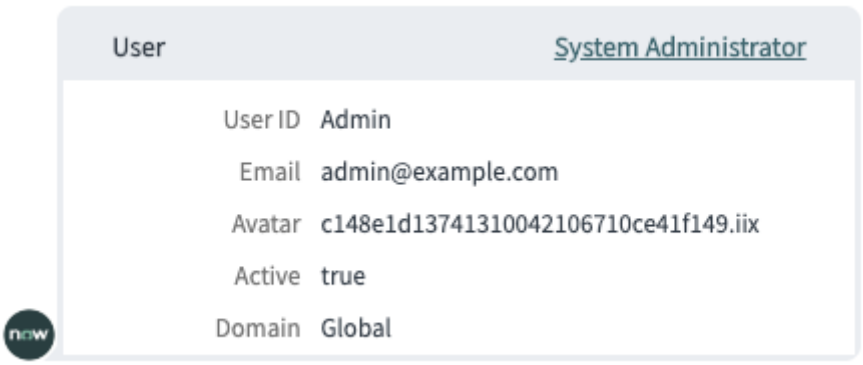
Create a custom Virtual Agent component to gather input or display information in the Virtual Agent client interface.

Types of Virtual Agent components

To develop a component for Virtual Agent, add specific properties and actions to interact with the Virtual Agent client interface. The properties required depend on the type of component you are creating.

Response component

A response component only provides information to the user, and does not gather input or handle user interaction. For example, a card control that does not require user input and is only in the conversation



once.

Add a property to your response component to handle the data sent by the Virtual Agent server.

Output component properties

Property	Description
controlData	Initial data that the Virtual Agent server sends to your component as the topic runs. Data type: JSON Object

Input component

An input component displays information and/or gathers user input. It includes the same property as the output component to handle data sent by the server, but has more possible states and requires user interaction.

Input component properties

Property	Description
controlData	Initial data that the Virtual Agent server sends to your component as the topic runs. Data type: JSON Object
responseValue	Data sent to the component from the user's response, either from the client directly, or from the server if there is a refresh. Only use in components that require user input. Data type: JSON Object
forceCloseControl	Flag that indicates whether the component can accept input. When true, the control closes and the user cannot interact with it. Monitor changes on the client to update this value. Only use in components that require user input. Data type: Boolean

Use this action to emit data from user interaction.

Input component actions

Action	Description
VA_CONTROL#ValueSent	Payload data from the client to send to the server. Only use in components that require user input. Data type: JSON Object

Input component states

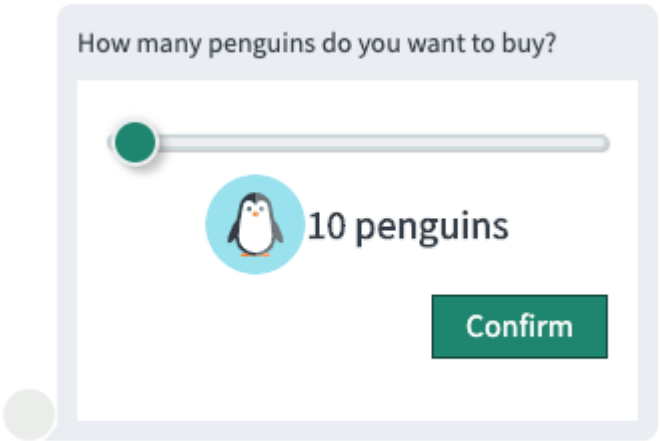
Because they accept data, input components must handle multiple states. The state flow is generally as follows:

1. Virtual Agent shows the custom component in the waiting for input state.
2. The user interacts with the component to provide input.

3. The component closes and sends the `responseValue` property to the server.
4. The server runs the server-side logic and sends the component with the user's input back to the client.

Waiting for Input

The initial state of a component waiting for user interaction. The `controlData` property is set, but the `forceControlClosed` property is false. This example shows a slider component in the waiting for input state.



In this example, if the user has not provided a `responseValue` and the control is not closed, the slider and the input button display.

```
const {controlClosed, sliderVal, sliderMin, sliderMax}
 = state;
return (<div class={{"slider-chat": true}}>
    {responseValue ? null :  <Fragment>
    <div class={{"slider-label": true}}>{label}</div>
    {controlClosed ? null : <Fragment>
        <div class={{"slider-container": true}}>
                    <input on-change={onSliderChange}
 type="range" min={sliderMin} max={sliderMax}
 value={sliderVal} class={{"slider": true}} />
            <div class={{"slider-value": true}}>
                    {unitIcon &&
                     <div class={{"unit-icon":
 true}}><img src={unitIcon} /></div>
                     }
                     {sliderVal} {unitName}
                </div>
        <div class={{"button-container":true}}>
            <now-button variant="primary"
 label={buttonText || 'Confirm'} />
         </div></div></Fragment>}
```

Handling Input

In the previous example, the slider uses a `now-button` component, which the user clicks to confirm input and send it to the server. When the user clicks the button, the VA_CONTROL#VALUE_SENT action fires with the `responseValue` payload.

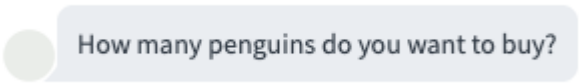
```
'NOW_BUTTON#CLICKED': (data) => {
    const {updateState, dispatch, state: {sliderVal}} =
 data;
    updateState({controlClosed: true});
    dispatch('VA_CONTROL#VALUE_SENT', {
        value: {
            sliderVal
        }
    });
}
```

Closed

A component that is closed can no longer accept user input. Components are generally closed because:

- The user responded to the component. The component closes and the conversation continues.
- The user ended the chat. The server does not wait for a response.

For example, the slider component only renders the original prompt when in the closed state.



Sending response

After the user responds, the control is rendered again on the user's side of the conversation with the value of the `responseValue` property.

For example, the slider control uses this snippet to render the response.

```
{responseValue && <div className="response-container
 slider-value">
   {unitIcon &&
   <div className="unit-icon"><img
 src={unitIcon} /></div>
   }
   {returnVal} {unitName}
</div>}
```

Adding the component to Virtual Agent Designer

After developing the component and deploying it to your instance, add it to Virtual Agent Designer using a custom control and definition. For more information, see Virtual Agent custom controls.

Add properties to communicate with Virtual Agent

To develop a component for Virtual Agent, add specific properties and actions to interact with the Virtual Agent client interface. The properties required depend on the type of component you are creating.

Before you begin

- Set up your environment
- Set up your project
- Develop a component

Procedure

1. In your component code, add the Virtual Agent properties to interact with the Virtual Agent client interface.

Virtual Agent component properties

Property	Description
controlData	Initial data that the Virtual Agent server sends to your component as the topic runs. Data type: JSON Object
responseValue	Data sent to the component from the user's response, either from the client directly, or from the server if there is a refresh. Only use in components that require user input. Data type: JSON Object
forceCloseControl	Flag that indicates whether the component can accept input. When true, the control closes and the user cannot interact with it. Monitor changes on the client to update this value. Only use in components that require user input. Data type: Boolean

2. If creating an input component, use the `VA_CONTROL#VALUE_SENT` action to send user values to the server.

Virtual Agent component actions

Action	Description
VA_CONTROL#VALUE_SENT	Send data from the client to send to the server. Only use in components that require user input. Data type: JSON Object

What to do next

Test a component for Virtual Agent and Deploy a component to an instance.

Related topics

Create custom components using ServiceNow CLI

Test a component for Virtual Agent

Test your Virtual Agent custom component before deploying it to your instance.

Before you begin

- Set up your environment
- Set up your project
- Develop a component
- Add properties to communicate with Virtual Agent

Role required: virtual_agent_admin or admin

About this task

Virtual Agent components must be tested within the Virtual Agent client chat to ensure that the component responds correctly to user input. You can set properties in your component project to test your component in a mock Virtual Agent test client tool.

Procedure

1. Add a dependency on the test client tool to your component project.
   a. Open the `<component-name>/package.json` file in your component project.
   b. Add `"@servicenow/sdk-ci": "1.0.8"` and `"@servicenow/library-translate": "^18.0.0"` to the `devDependencies` object.

   Example

   Here is an example `package.json` file with the correct dependencies.

```
"dependencies": {
    "@servicenow/ui-renderer-snabbdom": "xanadu",
    "@servicenow/library-translate": "^18.0.0",
    "@servicenow/now-button": "xanadu",
    "@servicenow/now-dropdown": "xanadu",
    "@servicenow/sass-generic": "xanadu",
    "@servicenow/cli-archetype": "xanadu",
    "@servicenow/sdk-ci": "1.0.8"
}
```

2. Add sample properties to use as the initial state of the component in your test.
   a. Add a `<component-name>/example/sampleProps.json` file with initial properties to render in your test.

   Example

   Here is an example `sampleProps.json` file with initial properties set for a slider component.

```
{
 "label": "How many penguins do you want to buy?",
 "defaultValue": 10,
 "sliderMin": 20,
```

```
 "sliderMax": 80,
 "unitName": "penguins",
 "unitIcon":
 "https://image.flaticon.com/icons/svg/141/141836.svg"
}
```

3. Update the `example.js` file to open the component through the test client tool using the sample properties that you created.

   a. Open the `<component-name>/example/element.js` file in your component project.

   b. Add statements to import `@servicenow/sdk-ci` and the sample properties file you created.

   c. Add the following statement, replacing `<component-name>` with your component's name to create the test tool with initial data from your sample properties.

```
const el =
 document.createElement('tool-ci-custom-control-tester');
el.componentTagName="<component-name>";
el.initialExampleData=sampleProps;
document.body.appendChild(el);
```

Example

Here is an example `example.js` file that opens the component using the test client tool.

```
import '../src/now-chat-control-slider';

import '@servicenow/sdk-ci';

import sampleProps from './sampleProps.json';

const el =
 document.createElement('tool-ci-custom-control-tester');
el.componentTagName="<component-name>";
el.initialExampleData=sampleProps;
document.body.appendChild(el);
```

4. Run the development server command to view your component in a test browser.

```
$ snc ui-component develop [--entry entry --open --port port
 --host host]
```

Pass in values for these parameters.

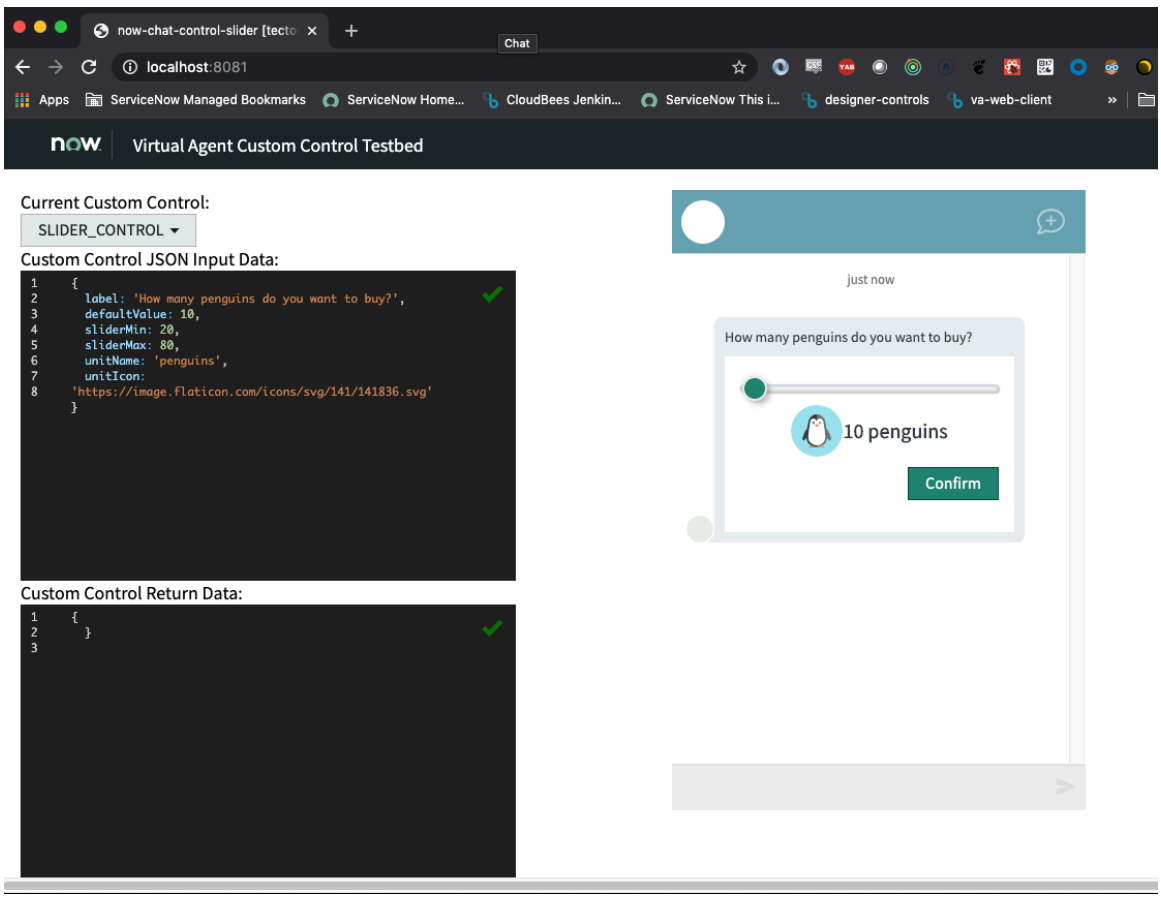
Name	Description
entry	Path to the test module in your component project. Default: `example/index.js`.
open	Opens the default browser and navigates to the test page. Default: false.
port	Port where the development server runs.

Name	Description
	Default: 8081.
host	Host address to use if you want your local development server to be accessible externally by others. Typically set to 0 . 0 . 0 . 0

Example

```
$ snc ui-component develop --entry example/hello.js --open
 --port 3000
```

The component opens in the test client tool. You can see the initial data provided in the **Custom Control JSON Input Data** field and the component's output in the **Custom Control Return Data** field.



What to do next

If your component is working as expected, deploy it to your instance. See Deploy a component to an instance.

After developing the component and deploying it to your instance, add it to Virtual Agent Designer using a custom control and definition. For more information, see Virtual Agent custom controls.

Related topics

Create custom components using ServiceNow CLI

Add a component to Agent Workspace

Use custom components to create a custom Workspace interface to fulfill the specific need of your company's agents.

Communicating with customers through multiple channels can be time consuming. To be efficient in these omni-channel interactions, your agents need a single view of customer information to reduce context switching between multiple tools. By developing custom components for Workspace, your team can bring communications from multiple channels into one interface.

Adding components to a Workspace

Once deployed to your instance, you can add components to Workspace in these ways.

Add a component to a Workspace modal

Use a UI action to launch a custom component in a modal so an agent doesn't have to navigate to a different screen to accomplish a task. For more information, see Render a component in a modal.

Add a component to a Workspace landing page using UI Builder

Use the UI Builder to create custom landing pages for your agents. UI Builder is a drag-and-drop tool that lets you visually arrange workspace components. For more information, see Configuring Configurable Workspace.

Configure properties in the `now-ui.json` file to use deployed components in the UI Builder. For instructions, see the ServiceNow Developer Site.

Add a component to a Workspace record view

You can add custom or standard components to the component area in the Workspace record view. For more information, see Forms.

Add a component to UI Builder

Set properties in a configuration file to add your component to the UI Builder in your instance.

Before you begin

- Set up your environment
- Set up your project
- Develop a component

Procedure

1. In your project directory, open `now-ui.json`.
2. Add the `components.[component-name].uiBuilder` object to the file.
   This object adds the component to the UI Builder.

   This object includes these key-value pairs:

Key	Data type	Description
components.[component-name].uiBuilder	Object	Object that adds the component to UI Builder.
components.[component-name].uiBuilder.label	String	Required. The display name of the component in UI Builder.

Key	Data type	Description
components.[component-name].uiBuilder.icon	String	Required. The name of the icon that appears in UI Builder.
components.[component-name].uiBuilder.description	String	Required. A short description of the functionality of the component.

Example

```
{
  "components": {
    "card": {
      "uiBuilder": {
        "label" : "Card",
        "icon" : "chat-fill",
        "description" : "A visual card format for a record.",
        "associatedTypes": ["global.core"]
      },
```

3. If your component includes properties, add the `components.[component-name].properties` array to the file.
This adds the properties as configuration options for the component in UI Builder.

   This object includes these key-value pairs:

Key	Data type	Description
components.[component-name].properties	Array <Object>	An array of objects that includes all the properties of the component and all relevant information about those properties.
components.[component-name].properties[].name	String	Name of the property in your component's code.
components.[component-name].properties[].label	String	Display name of the property to display in UI Builder, if applicable.
components.[component-name].properties[].description	String	A short description of what the property does or how to use it.
components.[component-name].properties[].readOnly	Boolean	When true, prevents a user from configuring the property in UI Builder. Default: false.
components.[component-name].properties[].required	Boolean	When true, the user must configure the property. Default: false.
components.[component-name].properties[].defaultValue	String	The default value when the user does not provide one.
components.[component-name].properties[].associatedTypes	Array	Describes where in the UI Builder toolbox the component appears. Value must be `"global.core"`.

Key	Data type	Description
components.[component-name].properties[].typeMetadata	Object	Extra configuration data that some data types require, such as reference properties and choice lists.

Example

```
{
  "components": {
    "properties": [
      {
   "name": "backgroundColor",
   "label": "Background Color",
   "description": "Background Color",
   "readOnly": false,
   "required": false,
   "defaultValue": "lightgray"
      },
      {
         "name": "cardType",
   "label": "Type of card",
   "description": "Type of card",
   "readOnly": false,
   "required": false,
   "defaultValue": ""
      }
   ]
```

4. Save the file.

What to do next

Deploy a component to an instance.

Related topics

Create custom components using ServiceNow CLI

Deploy a component to an instance

Deploy your component to display in your instance as an application plugin.

Before you begin

- Set up your environment
- Set up your project
- Develop a component

Procedure

Deploy the component to your instance.

a. Open your system's command-line tool and execute this command.

```
$ snc ui-component deploy [--open --force]
```

Pass in values for these arguments.

Name	Description
open	When true, opens the default browser and navigates to UI Builder in your instance. Default: false
force	Deploys component changes and overwrites any existing component records. Default: false.

Example

```
$ snc ui-component deploy --open
```

What to do next

Navigate to your instance and add your component to a modal or a landing page in Agent Workspace, or to Virtual Agent. For Agent Workspace, see rendering a Now component in a modal or creating custom landing pages for workspaces. For Virtual Agent, see Virtual Agent custom controls.

Related topics

Create custom components using ServiceNow CLI

Change a component's application scope

Change a component's application scope if you encounter scope issues when deploying your component to an instance. For example, you may have set an application scope that is already in use or invalid while creating a component in offline mode.

Before you begin

- Set up your environment
- Set up your project
- Role required: none

About this task

Procedure

1. In your project directory, open `now-ui.json`.
2. Update the `scopeName` key with the desired scope name.

   Use the namespace identifier guidelines for application development on the instance. For more information, see Application scope.

   When reserving an application scope, follow these requirements:

   - Maximum: 18 characters.
   - Case: snake case.

Example

```
"scopeName": "x_customerprefix_componentname"
```

3. Deploy your component.
   See Deploy a component to an instance.

Related topics

Create custom components using ServiceNow CLI

UI generation

Use ServiceNow® Now Assist for UI generation to quickly create UI's by describing what you want using natural language.

Get started

Explore UI generation  Learn how you can create experiences with the power of AI.	Configure UI generation  Configure experience generation.
Using experience generation  Quickly generate an experience with natural language.	Reference  Get details about properties, roles, and more.

Important:

- Not all model providers are available for customers with in-country SKUs, and some Now Assist products/features are currently unavailable for in-country customers. For more information, see the KB1584492 article in the Now Support Knowledge Base. Be sure to check for model provider availability updates in future releases.
- Some Now Assist products/features are currently unavailable for customers in the FedRAMP, NSC DOD IL5, or Australia IRAP-Protected data centers, self-hosted customers, or in other restricted environments. For more information, see the KB0743854 article in the Now Support Knowledge Base . Be sure to check for availability updates in future releases.
- Some Now Assist products/features are currently available only for customers in some regions. Be sure to check for availability updates in future releases.
- Some AI products and skills are not available in Regulated Markets. For more information, see KB2593939: Regulated Markets AI Products/Skills Not Available. Be sure to check for availability updates in future releases.

Troubleshoot and get help

- ServiceNow Community
- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

AI limitations

This application uses artificial intelligence (AI) and machine learning, which are rapidly evolving fields of study that generate predictions based on patterns in data. As a result, this application may not always produce accurate, complete, or appropriate information. Further, there is no guarantee that this application has been fully trained or tested for your use case. To mitigate these issues, it is your responsibility to test and evaluate your use of this application for accuracy, harm, and appropriateness for your use case, employ human oversight of output, and refrain from relying solely on AI-generated outputs for decision-making purposes. This is especially important if you choose to deploy this application in areas with consequential impacts such as healthcare, finance, legal, employment, security, or infrastructure. You agree to abide by ServiceNow's AI Acceptable Use Policy, which may be updated by ServiceNow.

Data processing

This application requires data to be transferred from ServiceNow customers' individual instances to a centralized ServiceNow environment, which may be located in a different data center region from the one where your instance is, and potentially to a third-party cloud provider, such as Microsoft Azure. This data is handled per ServiceNow's internal policies and procedures, including our policies available through our CORE Compliance Portal.

Data collection

ServiceNow collects and uses the inputs, outputs, and edits to outputs of this application to develop and improve ServiceNow technologies including ServiceNow models and AI products. In addition, this application will collect information about scripts (and associated script records) in which Now Assist for code generation is called. Customers can opt out of future data collection at any time, as described in the Now Assist Opt-Out page.

For more information, see the Now Assist documentation.

Exploring UI generation

Learn about how AI-generated experiences can empower developers building on the ServiceNow AI Platform.

Experience generation overview

With experience generation, you provide text describing the experience you want and Now Assist will create a fully functional starter experience for you.

Experience generation workflow

To generate an experience, you describe the table, chart type, and navigation your users will be working with. For best results, include as much detail as possible.

1. The user describes the desired experience and triggers experience generation.
2. The user reviews the AI-Generated experience and either accepts or rejects it.
   - If the user accepts it, the experience is saved and the user can begin working with the experience.
   - If the user rejects it, the experience is not saved, and the user can make changes to the experience description to achieve the desired outcome.

General guidelines experience generation

Use these general guidelines for experience generation to get better experiences.

Writing prompts

Write clear and specific but concise prompts

Describe the table and navigation your users will be working with. For best results, include as much detail as possible.

Experiment with different prompts

As you refine and experiment, the Now LLM Service uses this feedback to learn and improve.

- Try adding a specific title for your desired experience.
- Try including navigation details.
- Keep track of your prompts, including any modifications, and instructions for generating prompts to meet your specifications. This tracking enables easy regeneration of past results for comparative analysis.

Example prompts for experience generation

Strong prompt	Weak prompt	Notes
Create a Tabbed experience called, "Incident Management" to help track Incidents by Priority.	Track incidents.	Includes experience title details.
Create a Breadcrumb style experience called, "Change Request Tracking" that uses a Pie chart to group Change Requests by State.	Change request pie chart.	Includes navigation details.
Create a Tabbed experience to allow user to see Requested Items trending by Created Date in a Line Chart.	Requested items.	Includes chart details.
Create a Breadcrumb navigation experience called, "Task Tracker" with a Bar chart	Task tracker bar chart.	Includes Grouped by details.

Example prompts for experience generation (continued)

Strong prompt	Weak prompt	Notes
showing Tasks in a Bar Chart Grouped By Assigned to.		

Reviewing generated experience

Review experience

Implement strict and detailed reviews of the AI-generated experience to determine its accuracy, efficiency, and how well it adheres to your standards.

Configuring UI generation

Configure and install UI generation.

Install UI generation

Install the ServiceNow® Now Assist for Creator application from the ServiceNow Store to get Now Assist for UI generation.

Before you begin

- Review the Now Assist for Creator application listing in the ServiceNow Store for information on dependencies, licensing or subscription requirements, and release compatibility. Now Assist for Creator installs the Now Assist for UI generation application.

Role required: admin

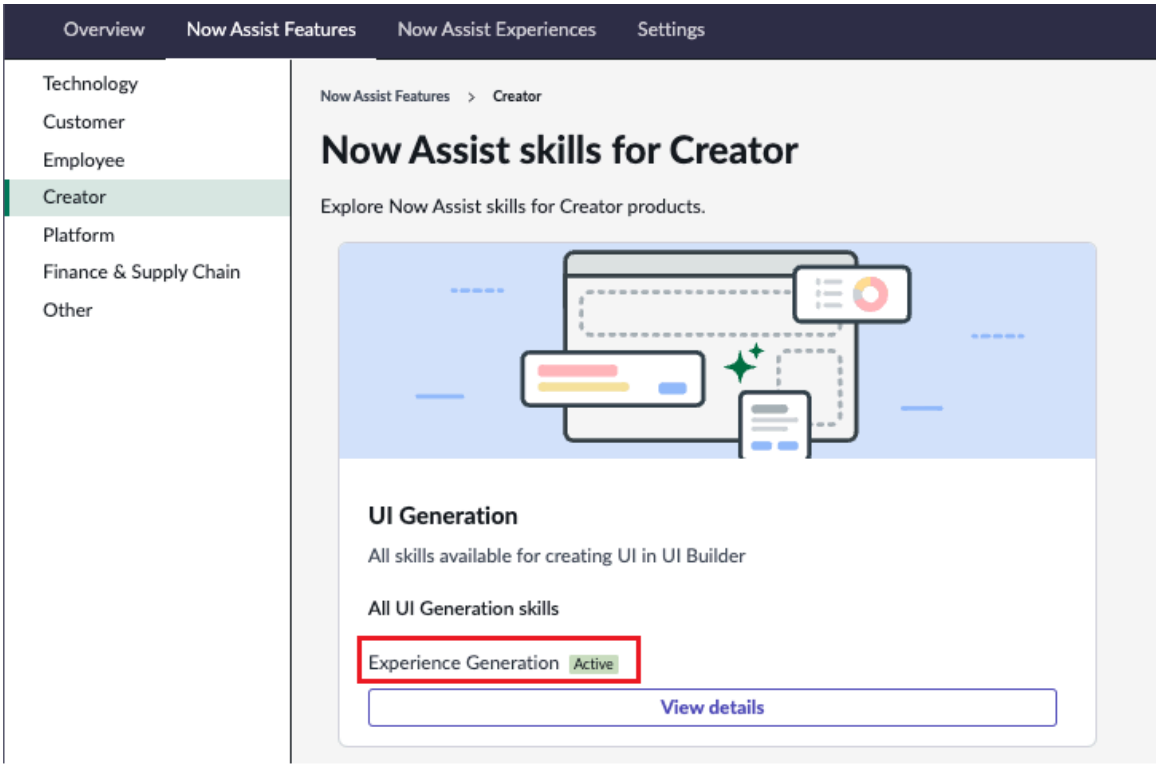
Procedure

1. From the Now Assist for Creator application page on the ServiceNow Store, select **Request App**.
2. After approval has been granted, on your instance, navigate to **All** > **System Applications** > **All Available Applications** > **All**.
3. Using the search bar, search for the Now Assist for Creator application (sn_now_creator).
4. Select **Install**.
5. Verify that Now Assist for Creator is installed:

   a. Navigate to **All** > *Now Assist Admin* > **Features**.

   b. In the workflow list, select **Creator**.

c. On the UI Generation card, verify that the Experience Generation skill is active.



For more information about Now Assist Admin, see Now Assist.

What to do next

Grant the now.assist.creator role to each user you want to use UI generation.

Related topics

Install Now Assist for Creator

Using UI generation

Quickly generate multi-page experiences using natural language with Now Assist.

When UI generation is enabled, the option to create an AI-Generated experience (✨) appears in the UI Builder dialog.

Developers must be assigned now.assist.creator role to use UI generation.

Experience generation

Use experience generation to create a multi-page experience using natural language with Now Assist.

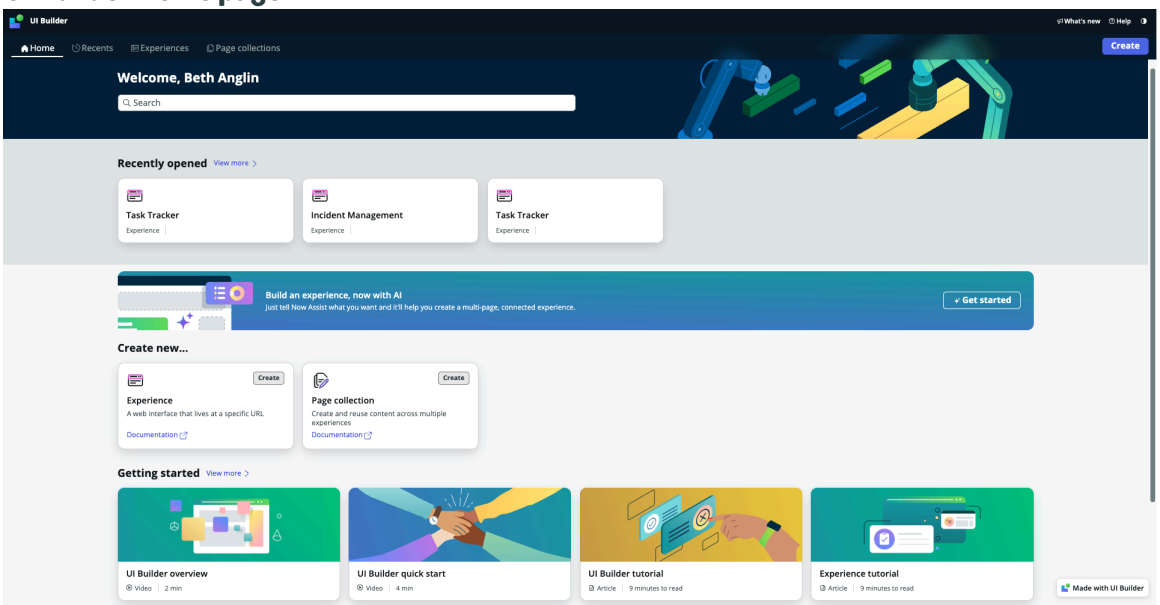
Before you begin

Role required: now.assist.creator

Procedure

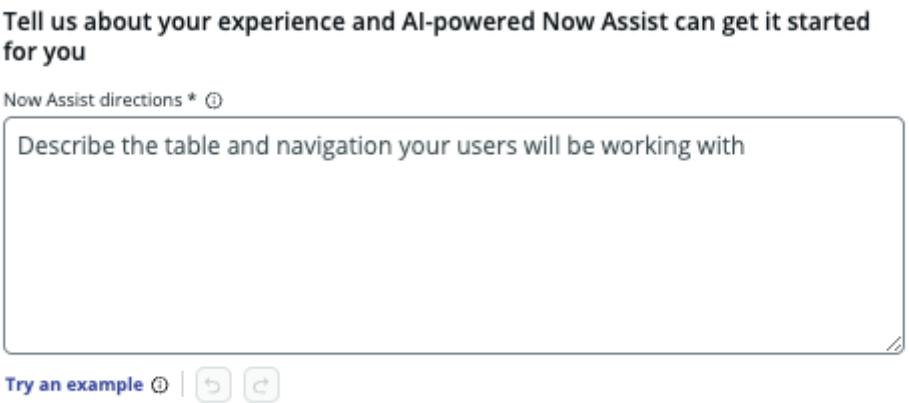
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.

UI Builder Home page



2. From the UI Builder Home page, select **Create**.
3. Select **AI-Generated experience**.

Create an experience form



4. In the **Create an experience form**, enter a description of the experience you would like to generate.

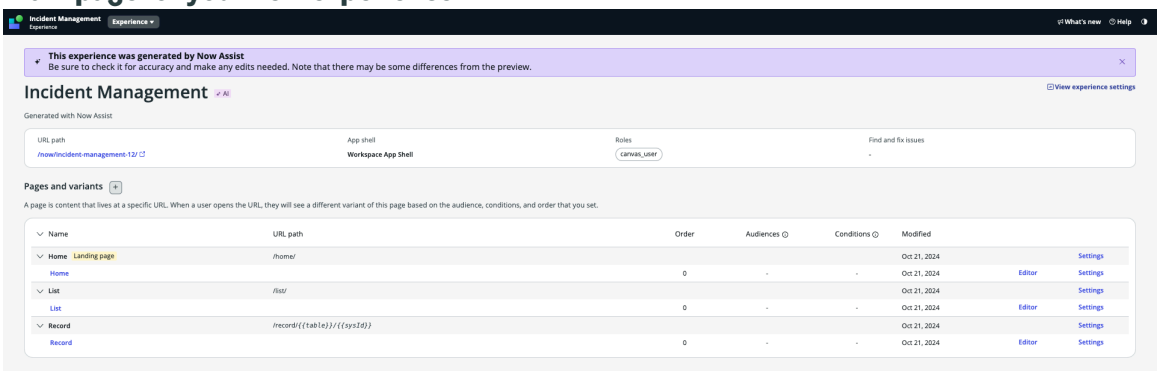
> **Tip:**
> Select **Try an example** to insert different experience examples to see what good Now Assist directions include.

5. Select **Generate preview**.

6. Review the experience and complete one of the following steps:
   - To proceed in creating your experience, select **Save and edit experience**.
   - To create a new experience, reword your description and select **Regenerate preview**.

   When you select **Save and edit experience**, the experience settings page indicates the UI was generated with Now Assist.
7. Review the main page for your new experience.

Main page for your new experience



Note:
In addition to the name of your new experience, the screen includes the URL, app shell, and the roles of users who can view the experience. Any pages or variants you create for the experience appear in the **Pages and variants** section.

UI generation reference

Reference topics provide additional information about configuration properties, roles, and more.

UI generation roles

The following roles are installed for use with UI generation.

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

Now Assist Creator [now.assist.creator]

Create experiences using AI-powered UI generation.

Groups

This role is assigned to no groups by default.

Contains Roles

This role contains no roles.

Elevated

This role is not an elevated role.

Special considerations

None

UI generation properties

You can adjust how UI generation functions on an instance using several advanced properties.

Note:
To open the System Properties [sys_properties] table, enter `sys_properties.list` in the navigation filter.

Decision Tables

Use decision tables to decouple decision logic from your code by creating and maintaining decision rules.

Important:
Try creating decision tables in Workflow Studio instead. Workflow Studio integrates workflow authoring, configuring, and monitoring into a single page experience. For more information, see Create decision tables in Workflow Studio.

Use decisions tables to help you reach outcomes that depend on multiple factors. In these tables, each factor is a decision input. For example, if you are trying to make a decision about car insurance coverage, your inputs might include the location where the insured person lives, the age and accident history of the insured person, the car make, the car model, and the car year. This logic can save time and present a more organized, readable format than using a script. Decision tables provide a single point where you can create, view, and modify decisions.

To interact with Decision Tables in script, use the DecisionTableAPI - Scoped, Global.

Components of a decision

Decision Table [sys_decision] records

Decision table records represent a single decision. In this record, you specify which table you want to use for your answers. This record also contains related lists where you can create your decision inputs and decisions.

Decision inputs [sys_decision_input] records

Decision input records represent your inputs that are used to obtain answers in a decision. These inputs can include a variety of types, including strings, references, true/false inputs, and dates. Each decision input has a specified input type and can be assigned a default value.

Decision [sys_decision_question] records

Each decision record represents a decision that is based on your inputs. Use the condition builder on the decision record form to create a condition that is based on the inputs for this decision. Then, you can select an answer record for this decision. The answer record can be any record from the table that you have defined in the Decision Table record. You can select the same answer record for more than one decision.

Answer [sys_decision_multi_result_element] records

Answer records represent answers that are reached using different decision input values. Answers records can be records on any table, but you need to choose the table when creating your Decision Table record. You could, for example, use the sys_choice table to use existing choice records. From the example about insurance coverage, you could create a table where each record contains details about the levels of insurance coverage.

Decision tables in the classic environment

In the classic environment, you can develop a table to use for your decision answer records. You then associate an answer record to each decision on your decision table. This answer record is returned when the decision is reached based on your inputs.

Note:
Workflow Studio decision tables provide a new intuitive interface to create decision tables. For more information, see Workflow Studio.

Create a decision table to resolve complex decisions

Create a decision table to help you resolve a complex decision that depends on multiple factors.

Before you begin

Role required: decision_table_admin

Procedure

1. Create or select a table to use for your decision answer records.
   You will associate an answer record to each decision on your decision table in later steps. This answer record is returned when that decision is reached based on your inputs. For information on creating a table, see Create a table.
2. Ensure that one field on the table that you use for your decision records has been set as the display value.

   This field is used as a label when you display decision answers in Workflow Studio.

   You can use an existing table or create a new table to use for your decision records.
3. Create a record on this table to each answer in your decision.
   If you have modified an existing table, you may need to customize the form to show the new fields that you have added.
4. Create a decision table record.

   a. Navigate to **System Definition** > **Decision Tables**, and select **New**.

   b. On the form, fill in the fields.

   Decision Table form

Field	Description
Name	Descriptive label for this decision.
Application	Application scope of this decision. This field defaults to the current scope.
Answer table	Table that this decision uses for answer records. Select the table that you created in the previous steps.
Accessible From	Option to limit the availability of this decision to the current scope. Select **All application scopes** or **This application scope only**.

   c. Right-click the form header and select **Save**.
   The form refreshes with the Decision Inputs and Decisions related lists.
5. Create decision input records.

a. In the Decision Inputs related list, select **New**.

b. On the form, fill in the fields.

Decision Input form

Field	Description
Name	Name of the decision that is associated to this decision input. This field is automatically populated.
Application	Application scope of this decision input. This field defaults to the current scope.
Type	Type of data that is used for this input. When you use this decision in a flow, you can only use data pills that match this type.
Active	Check box to activate this input. This field is selected by default.
Label	Descriptive label for this input.
Read only	Check box to make this decision input read-only.
Column name	Column name for this input. This field is automatically populated when you give the **Label** field a value.
Mandatory	Check box to make this decision input mandatory.
Display	Check box to indicates that this decision input is the display value for reference fields.
Choice	Select a method for users to see a list of suggested values: ▪ List menu without -- None -- ▪ List menu with -- None -- ▪ Suggestion field type If a choice is used, define your choices in the Choices related list at the bottom of the form.
Default Value	Default value for this input.

c. Select **Submit**.
Your changes are saved and the decision table record reopens.

6. Create decision records.

a. In the Decisions related list, select **New**.

b. On the form, fill in the fields.

Decision form

Field	Description
Label	Descriptive label for this decision table.
Application	Application scope of this decision. This field defaults to the current scope.
Order	Order in which the flow evaluates decisions. The order can be important in flows that use the **First decision that matches** option and ends after the first match is found.
Answer	Answer that is used when the conditions in this decision are met.
Default Answer	Check box to enable this decision as the default for your decision table.
Condition	Conditions that are needed to reach this decision. The fields available for your condition are the Decision inputs that are associated with this decision table.

c. Select **Submit**.
Your changes are saved and the decision table record reopens.

What to do next

With Workflow Studio, you can add your decision to the **Make a decision** flow logic. For more information about the **Make a decision** flow logic, see Make a decision flow logic.

Table Builder

Table Builder enables you to create, design, and administer tables and forms visually from a single user interface.

Announcements

The Table Builder for App Engine application has been combined with the standalone Table Builder application. Some features such as schema view, spreadsheet view, flows, and PDF extractor that previously were only available in the Table Builder for App Engine application are now available in Table Builder.

Get started

Explore	Use	Reference
Learn about Table Builder concepts and features.	Manage tables, forms, and related policies and rules using Table Builder.	Get details about Table Builder components like field properties and properties for policies and rules.

Get help with Table Builder

Contact your company's Customer Admin to unlock or add user accounts, perform restores or zBoots, and more.

Following is the intro video about the table builder.

https://player.vimeo.com/video/1138041032?h=eb605503b7&badge=0&autopause=0&player_id=0&app_id=58479

- Ask or answer questions in the Table Builder community forum
- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

Exploring Table Builder

Learn about Table Builder features that enable you to design tables, forms, and flows visually using a single user interface.

Learn more about Table Builder	Additional ServiceNow resources
You can use Table Builder to configure form elements such as layouts, fields, field configuration, and UI policies.	  Table Builder: Creating a custom order form

Key features

Table Builder includes these key features:

- Edit tables by adding additional fields or changing existing field properties for a particular application or globally.
- Visually customize form views by using a simple drag-and-drop editor interface. You can add and arrange sections, fields, and annotations as needed from a table that you select.
- Configure UI Policies for your forms from one interface or view these other related policies and rules:
  - Access control rules
  - Client scripts
  - Business rules
  - Workspace view rules
- Preview your forms throughout the editing process so that you can be sure that any changes function well in your environment.
- Integrate with App Engine Studio to promote a unified experience for designing your forms and managing data.

Accessing Table Builder

You can access Table Builder in several ways.

Launching Table Builder from App Engine Studio

Table Builder can be currently accessed from within App Engine Studio.

See Editing data in App Engine Studio for instructions on editing application data.

1. Open an application in App Engine Studio.

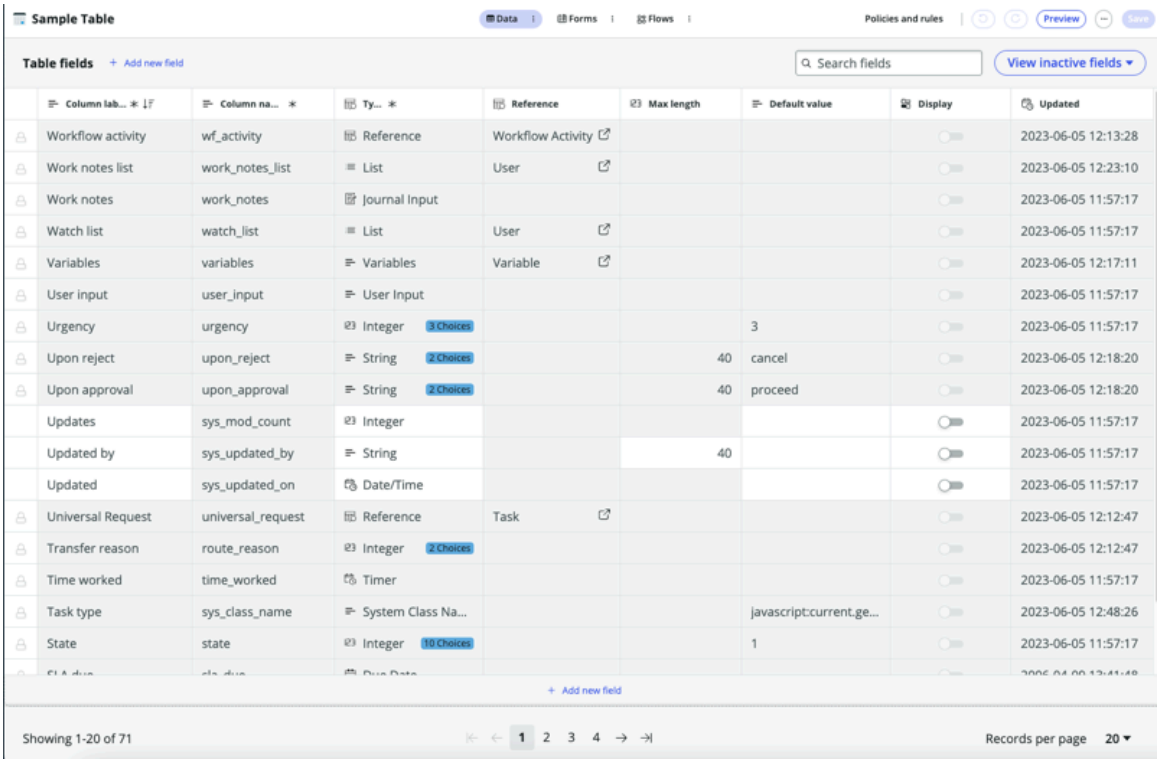
   Note:
   A data table must first be present in the application so that you can access Table Builder.

2. From the application home, select the menu icon (•••) next to a table, and then select **Edit**.

   Table Builder opens with the **Data** tab selected as shown in the following example.

Note:
To edit form views for the selected data table, navigate to the **Forms** tab.

Data tab



Launching Table Builder from UI Builder

Table Builder can be accessed from within UI Builder by clicking **Edit form view** at the bottom of the Config panel, for a selected form object.

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**, and then select a form component on a page.

   See Work with components in UI Builder for more information on adding components.

2. Click **Edit form view** at the bottom of the Configuration pane.

Table Builder opens with the **Forms** tab selected, where you can customize your form views for the selected table. To edit table data, navigate to the **Data** tab as shown in the following example.

Forms tab

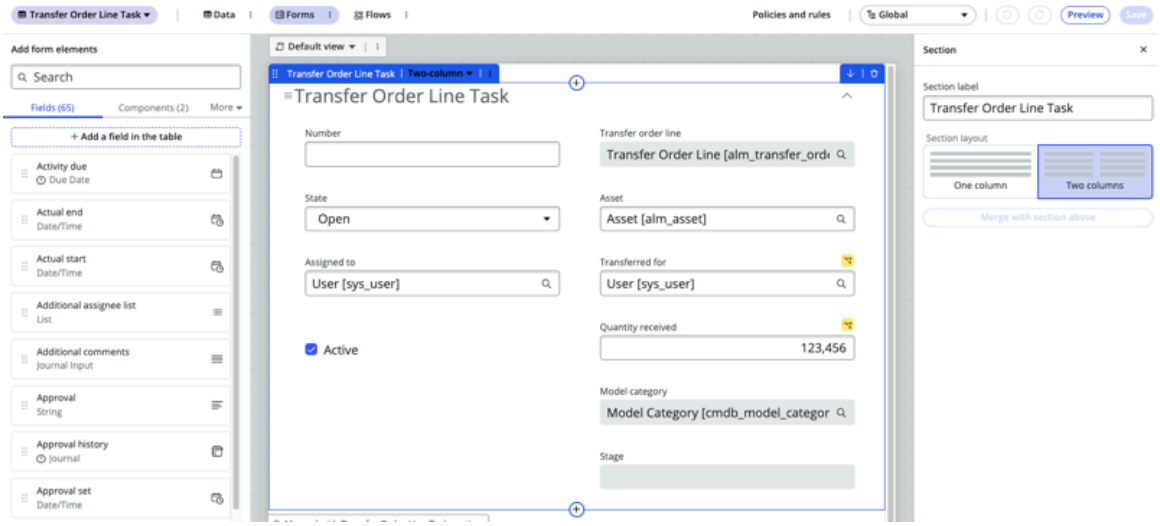


Table Builder permissions

To use Table Builder, you must either be an administrator, or have the following role permissions in ServiceNow AI Platform, or have relevant AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Table Builder permissions

Permission	Access
Personalize Form (personalize_form)	• Required to access **Forms** tab. • Required to customize fields on the form. • Does not grant dictionary access. • Does not grant Policies and Rules access. • Does not grant Tables access.
Personalize Dictionary (personalize_dictionary)	• Required to access **Data** tab. • Required to access field configuration. • Required to change dictionary fields. • Does not grant Policies and Rules access.
Personalize Rules (personalize_rules)	Enables access to the **Policies and Rules** tab.
Personalize Choice	Allows a user to configure choices for a field.

Table Builder permissions (continued)

Permission	Access
(personalize_choices)	
Flow Designer (flow_designer)	Required to access the **Flows** tab. Allows a user to configure flows. To work with Table Builder flows, you must also have personalize_form and personalize_dictionary permissions (or related delegated developer permissions).

Accessing Form Builder

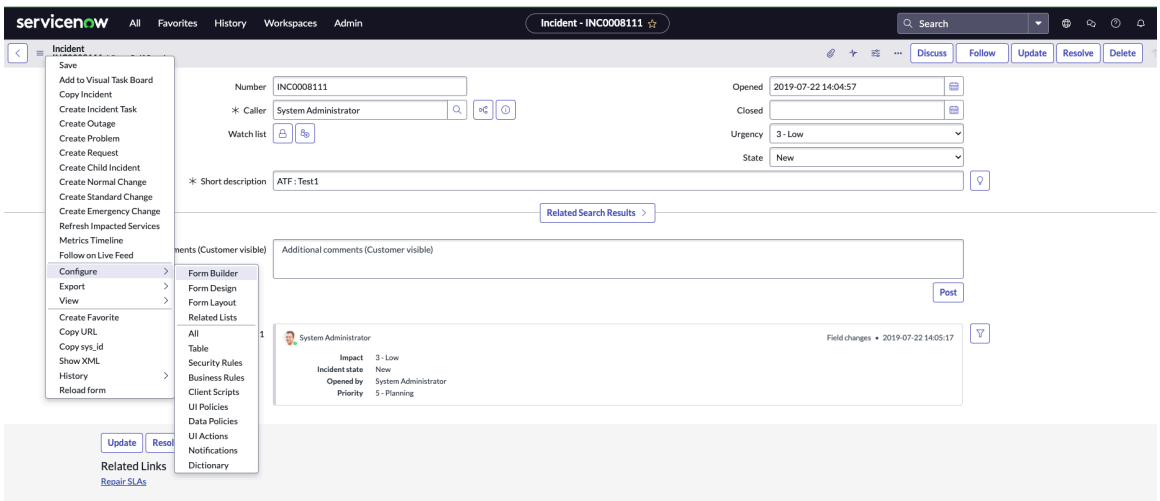
Form Builder is an application that is accessed most often through Table Builder on the **Forms** tab. You can also access Form Builder through the ServiceNow AI Platform® and the **Related Links** section of any table.

Accessing Form Builder from Table Builder

In Table Builder, access Form Builder on the **Forms** tab. For more information, see Forms in Table Builder.

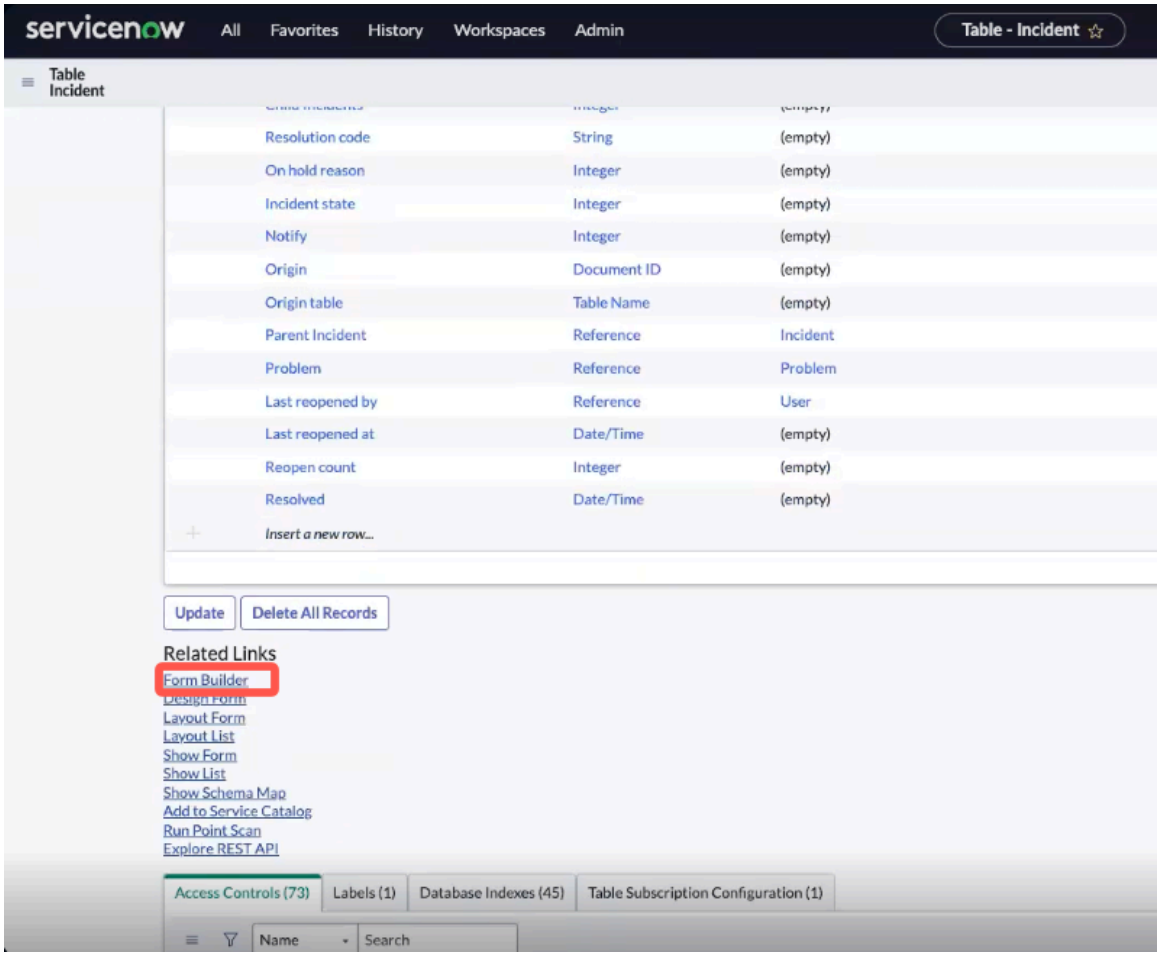
Accessing Form Builder from the ServiceNow AI Platform

Customize any ServiceNow AI Platform record or form using Form Builder. On the form context menu, select **Configure**, and then select **Form Builder**.



Accessing Form Builder from Related Links

You can also access Form Builder from the Related Links section of any table. Open Form Builder to make edits to the form that you use to populate records for that table.



Accessing Form Builder from UI Builder

You can access Form Builder directly from the UI Builder stage. Move your cursor to a form component that has a set **Table** value and **View** value, then select the **Edit Form** button. Your changes appear on the form after closing Form Builder.

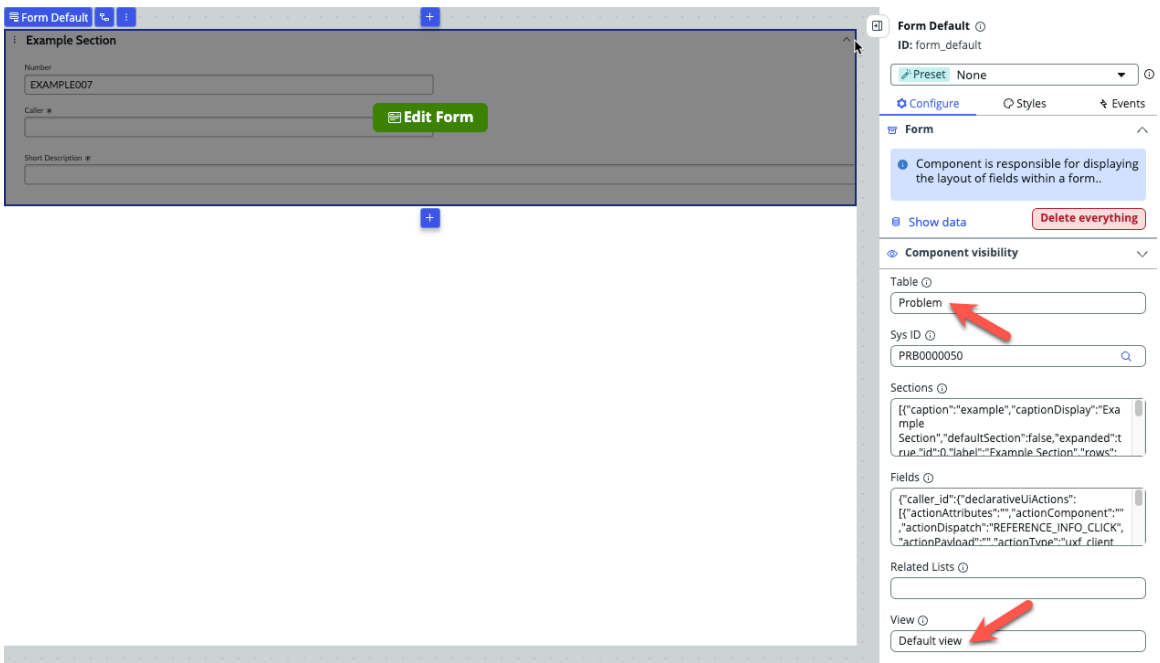


Table Builder workflow and navigation

You can start using Table Builder to manage your data tables, forms, or flows by reviewing this high-level workflow.

Learn about the high-level actions that you perform when you are using Table Builder.

Basic workflow

The following table outlines the basic actions that you take when working in Table Builder.

Basic workflow

Action	Description
Choosing a table or form to work with	To choose a data table or form to work with, launch Table Builder as instructed in Accessing Table Builder and choose the table or form to work with. **Note:** While using Table Builder in App Engine Studio, the table selected here is chosen when you click on a table in your app. However, if Table Builder is launched from UI Builder, you can choose a new table to work with from the drop-down list that displays the name of your currently selected table.
Selecting a domain and application scope	Select a domain and application scope to work within from Table Builder navigation. **Note:** While using Table Builder in App Engine Studio, the application scope selected is the app you are working in. However, if Table Builder is launched from UI Builder, you can choose an application scope to work with from the drop-down list in the top right side of your screen (e.g., Global). See Domain separation and Table Builder and Using an application scope with Table Builder.
Modifying table properties	Select **Data** tab to edit table properties directly. For more information, see Data in Table Builder.
Customizing form views	Select **Forms** tab to create and customize new views or configure the form elements for a selected view. For more information, see Forms in Table Builder.
Configuring policies and rules	Select **Policies and rules** tab to configure and control how data is displayed for your form views.

Basic workflow (continued)

Action	Description
	For more information, see Policies and rules in Table Builder.
Previewing a form and managing your changes	At any point in the process, you can preview the changes you've made by selecting **Preview**. For more information, see Preview your form. Save your form view changes at any point in the process by clicking **Save**. Your form view is ready and can be accessed by other applications in the ServiceNow AI Platform. **Note:** To undo ( ) or reapply ( ) changes you make, select the Undo or Redo icons at the top of your screen.

Navigational elements

The following table shows and describes the basic navigational elements shown in Table Builder.

Basic navigation elements

Navigational element	Description
Table selection list (Problem)	Choose an option to customize the policies and rules, table properties, fields, or form views for the selected table.
Select domain list (global)	Choose an option to customize the policies and rules, table properties, fields, or form views within the selected domain (for example, the Global domain).
Select scope list (Global)	Choose an option to customize the policies and rules, table properties, fields, or form views within the selected application scope (for example, the Global application scope).
Undo ( ) Redo ( )	Select the undo ( ) or redo ( ) icons in the navigation bar to revert to a previous editing state.
Advanced table properties menu ( )	Choose an option: • **Advanced view** to manage the advanced properties in the selected table. For more information, see Table administration. • **Form designer** to launch the selected table in Form Designer. For more information, see Using the Form Designer. • **Delete table** to delete the selected table.

Basic navigation elements (continued)

Navigational element	Description
	Note: This option only appears on the Data tab.
Preview (Preview)	Select **Preview** to review the selected form as how it is displayed in the ServiceNow AI Platform. For more information on previewing, see Preview your form.
Save (Save)	Select **Save** to commit your changes to your selection.
Search fields and filter options ( )	Use the search box to filter the list of field columns by keyword or use the **Filter options** list to show inactive fields or to hide extended fields from the list (if a table has been extended from an existing table).
PDF extractor ( )	If a PDF is associated with the selected data table, select this element to launch the PDF extractor tool and view the PDF. See Use a PDF to create data tables.

Domain separation and Table Builder

Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Support level: Standard*

The support level is Standard but has some exceptions or special conditions.

- Includes all aspects of **Basic** level support.
- Business logic: The service provider (SP) creates or modifies processes per customer. The use cases reflect proper use of the application by multiple SP customers in a single instance.
- The instance owner must be able to configure minimum viable product (MVP) business logic and data parameters. This configuration is done per tenant, as expected for the specific application.

Sample use case: An admin must be able to make comments required when a record closes for one tenant, but not for another.

For more information on support levels, see Application support for domain separation.

Overview

Table Builder enables developers to configure the layout and logic for each form view for tables in their application. When launched from UI Builder, Table Builder supports domain separation, which is the ServiceNow instance-wide multi-tenant architecture.

Note:
Domain separation is supported for Table Builder when it is launched from UI Builder. Domain separation is not supported when launching Table Builder from App Engine Studio.

Table Builder enables developers in domain-separated environments to create forms while they are in the same browser window. Domain separation in Table Builder works similarly to an application scope that helps administrators to create or edit in a multi-tenant environment.

Standardization is the key principle to maintaining a stable, healthy, and scalable ServiceNow instance, where domain separation is installed. By having standardization, you have a common configuration that most of the instance operates by. When an instance has hundreds or thousands of domains, managing them successfully requires rigorous governance. Domain-specific configurations should be done only if they are deemed necessary by the instance owners. Most instances should follow the common instance configuration to provide a more uniform experience across the instance. It also lets instance owners minimize the code sprawl that slows the adoption of new ServiceNow features that are included as part of the release upgrades.

How domain separation works in Table Builder

Table Builder enables developers to switch the session domain to create domain overrides to the form layout, section layout, and form logic. System dictionary record changes (such as field metadata for a field label) do not require domain overrides. Form layout changes require domain overrides to both the form record and the section when the change is performed at a lower domain. When creating a domain override, section labels can't be edited prior to saving the override.

Note:
UI policies can have overrides at lower domains, as well. An override indicator appears on the UI policy.

In the global domain, a developer can expand the domain scope to see all form views from different domains. Developers can see all form views and views with overrides that are indicated with an icon.

Note:
Single section form records do not have a *sys_ui_forms* record attached to it. Instead, these records only have an attached *sys_ui_section* record until a second section is created. If a single section form has existing domain overrides, and the developer adds another section to the form, the existing domain overrides will break.

Related topics

Domain separation for service providers

Using an application scope with Table Builder

Selecting an application from the application scope list in Table Builder enables you to customize form views for the users of the selected application in the ServiceNow AI Platform.

Application scoping protects applications by identifying and restricting access to application files and data. As an administrator, you can specify what parts of an application are accessible to other applications, which helps to protect data and application files. In Table Builder, you and your developers can set an application scope when you are working with a table or form.

Changing the scope in Table Builder also changes the scope in the ServiceNow AI Platform®, and changing the scope in the ServiceNow AI Platform® will change the scope in Table Builder.

When you are creating a form, you must be aware of the scope that you are working in. Choose the correct application scope for a table, form, or policy. The **Select scope** list is near the **Select Domain** list. The scope defaults to the scope that the user is in within the ServiceNow AI Platform®. If your users change to a different scope while in a page, they are notified that they are in a different application scope from the one that the form was created in.

Using Table Builder

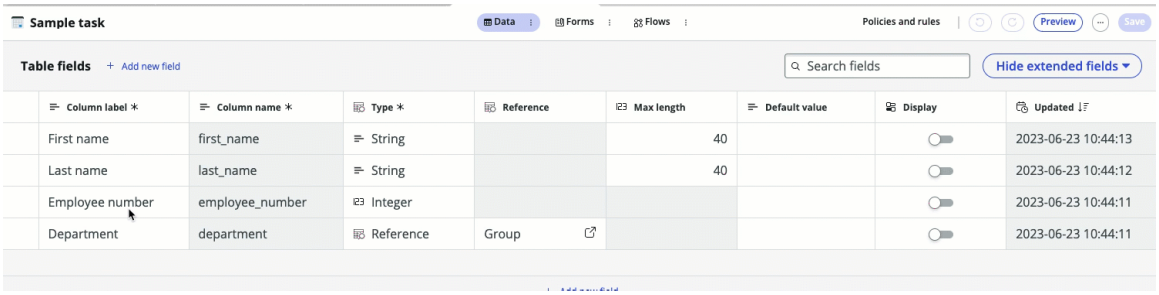
Use Table Builder, to design and customize tables, forms, and form elements for users from a single interface without needing to navigate between existing ServiceNow AI Platform tools.

Data in Table Builder

With Table Builder, you can modify the table properties and manage the fields that are available for a selected data table.

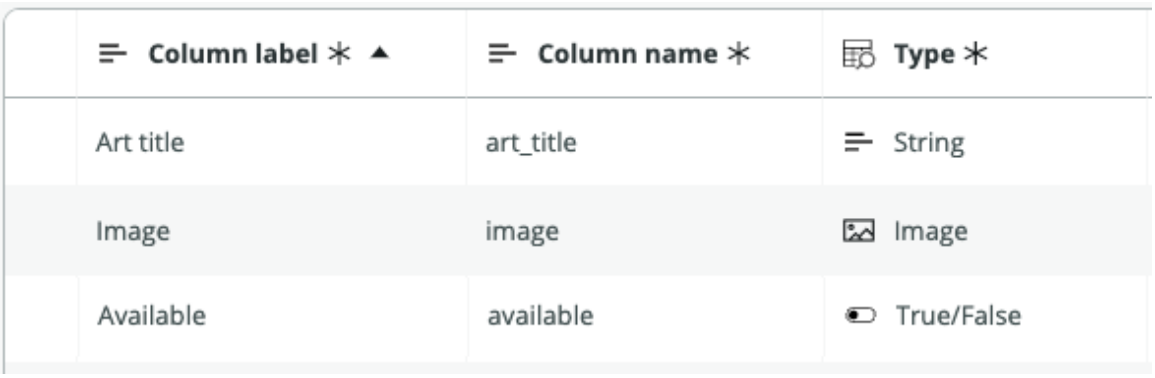
Table Builder is a tool for editing data tables. You edit a table on the **Data** tab by editing the columns of the table.

Table Builder

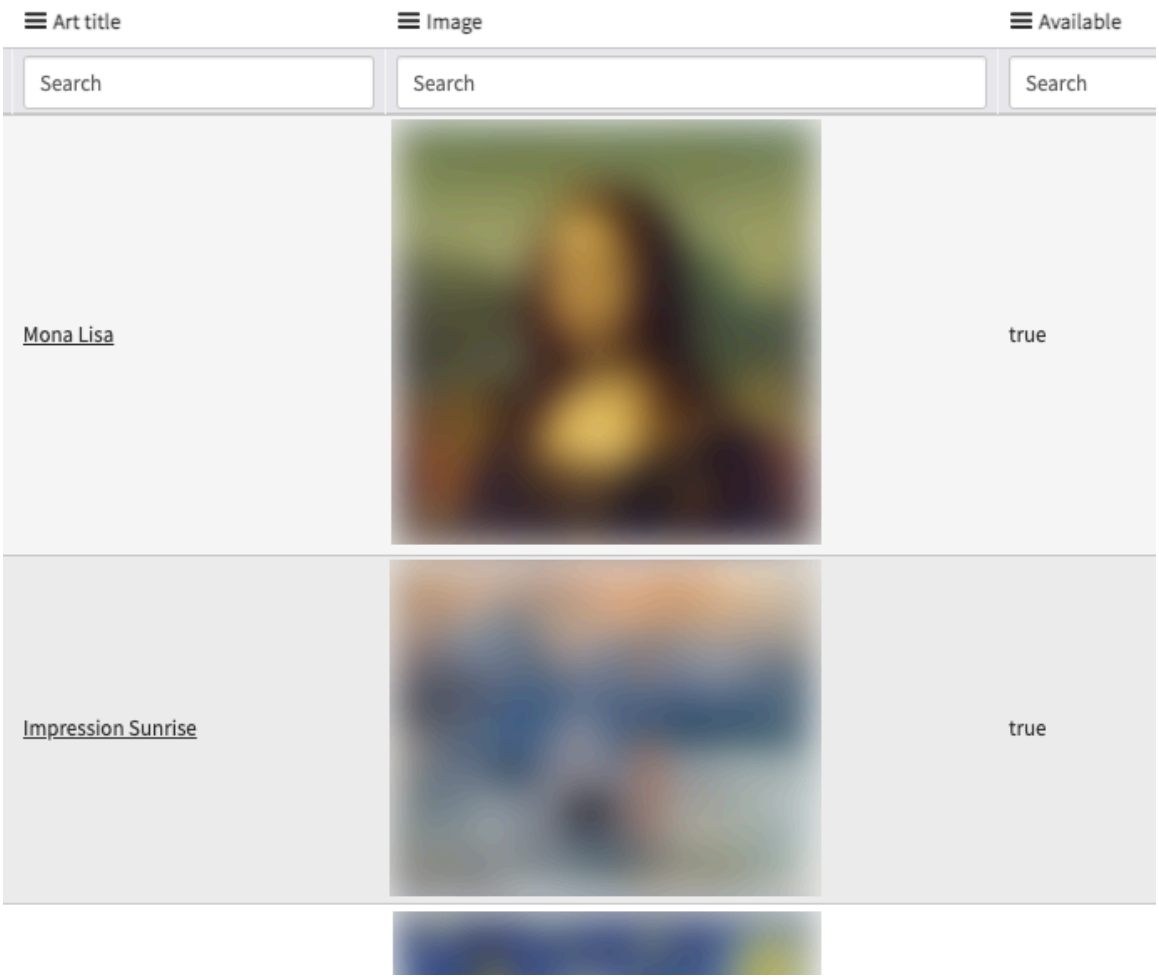


In Table Builder, each table column is represented as a row. For example, "Art title," "Image," and "Available" appear as column headers on the actual table. However, when you edit the columns in Table Builder, they appear as rows.

Example: Columns in Table Builder



Example: Actual data table shown with two records



To easily find a specific table column to edit in the default Fields view, type the column name into the search box at the top right. As you type, the list of columns below is filtered to match the entered keyword.

Note:
Use the **Filter options** list to show inactive fields or to hide extended fields from the list (if a table has been extended from an existing table).

Search for a table column

As you edit the table columns in Table Builder, think of each row as a field on a form. For example, if you change a column label, use a label that describes the data that you expect the user to enter in the form field. Change the column type to support this kind of form entry.

Modify table properties and table structure in **Fields** view, where you can perform the following actions:

- Add a new column to a table. See Add a table column in Table Builder.
- Delete a table column. See Delete a table column in Table Builder.
- Edit table properties. See Edit table properties in Table Builder.

- Change a column label. See Change a column label in Table Builder.
- Change a column name. See Changing a table column name in Table Builder.
- Change a column type. See Change a column type in Table Builder.
- Add a default value to a table. See Add a default value to a table column in Table Builder.
- Select a column as table display name. See Select a column as the table display value in Table Builder.
- Delete a table. See Delete a table in Table Builder.

Work with table structure and your data records in spreadsheet format within **Spreadsheet** view, where you can perform the following actions:

Note:
This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.

- Add a new data record. See Add a new data record.
- Edit individual data records. See Edit individual data records.
- Sort your data records. See Sort your data records.
- Filter your data records. See Filter your data records.
- Edit column field properties. See Edit column field properties.
- Add new columns. See Add new columns.
- Duplicate an existing column. See Duplicate an existing column.
- Reorder columns. See Reorder columns.
- Set column visibility. See Set column visibility.
- Pin columns. See Pin columns.
- Delete a table column. See Delete columns.

You can also view the table relationships for your table in **Schema** view. See Schema view in Table Builder.

Note:
This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.

Spreadsheet view in Table Builder

Use **Spreadsheet** view in Table Builder to work with your application data records and data structure in a spreadsheet format.

Overview

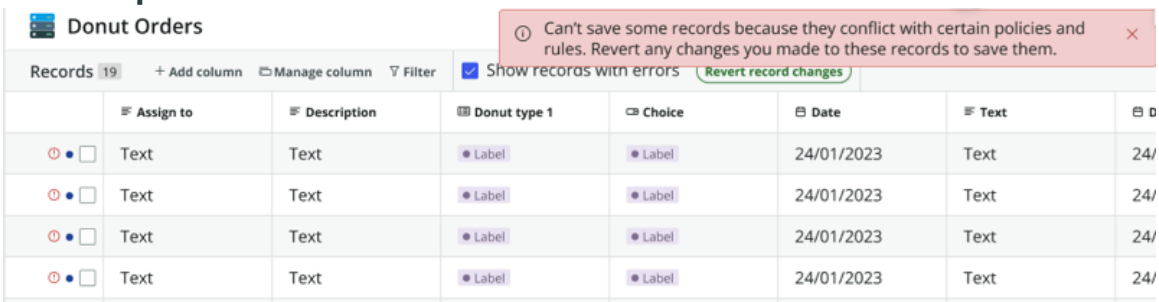
Spreadsheet view in Table Builder, enables you to review your application tables and their accompanying data in a familiar spreadsheet format. You can add columns, delete columns, and change column properties as well as review and update individual data records within those columns from a single screen.

Saving data records in Spreadsheet view

Sometimes errors occur when trying to save data records (i.e., data conflicting with applicable policies or business rules). When this happens, a warning popup displays along with a filtered list of errored data records.

Correct the data in each record individually, or select **Revert record changes** to revert any changes you made to the errored records.

Errors in Spreadsheet view



Use the following procedures to edit table field structure and any imported data records for a selected table using **Spreadsheet** view in Table Builder

Add a new data record

Add a new data record within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

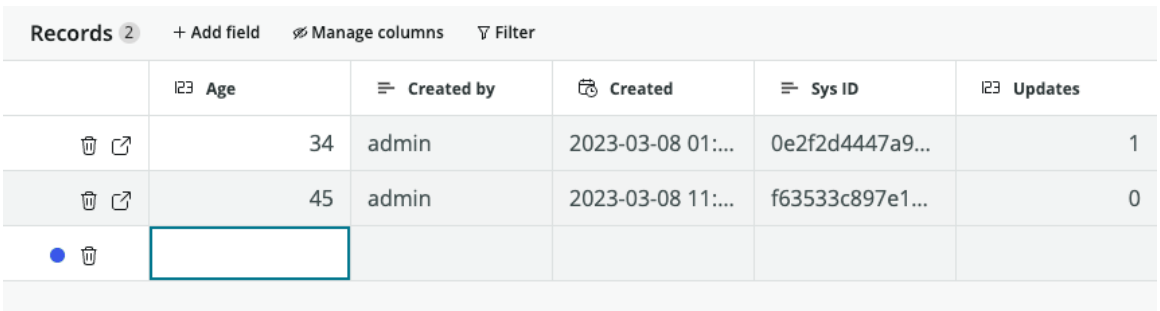
Procedure

1. Navigate to the **Data** tab.

> **Note:**
> Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Select the **Add new record** icon ( ).

Add new record



3. Make your changes to the data record.
4. Select **Save**.
   See Saving data records in Spreadsheet viewfor details on troubleshooting errors.

Edit individual data records

Edit individual data records within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

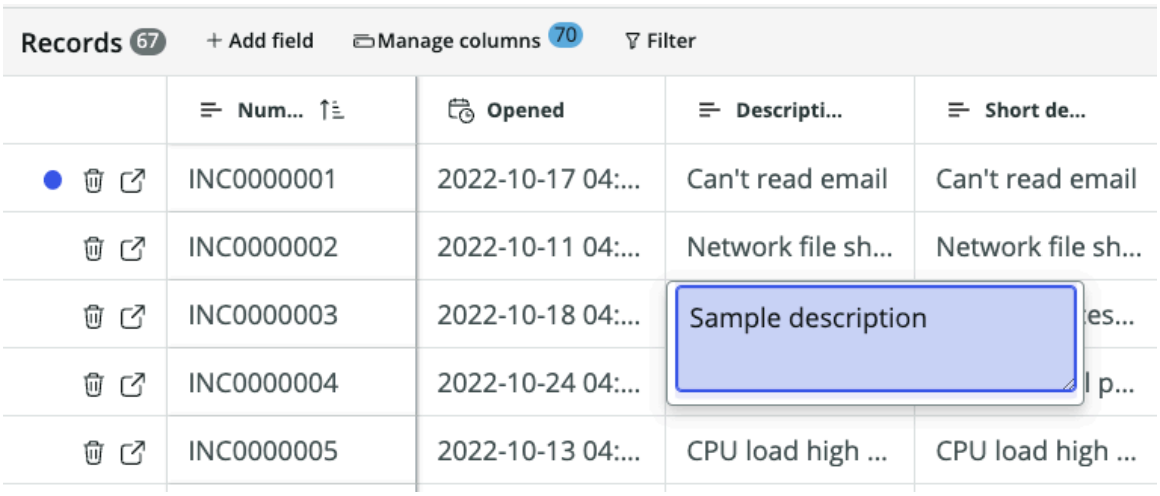
Procedure

1. Navigate to the **Data** tab.

   Note:
   Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Click into any editable data record field to launch the data for editing.

   Edit a field



3. Make your changes to the data.
4. Select **Save**.
   See Saving data records in Spreadsheet viewfor details on troubleshooting errors.

Sort your data records

Sort your data records by a specific field column within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

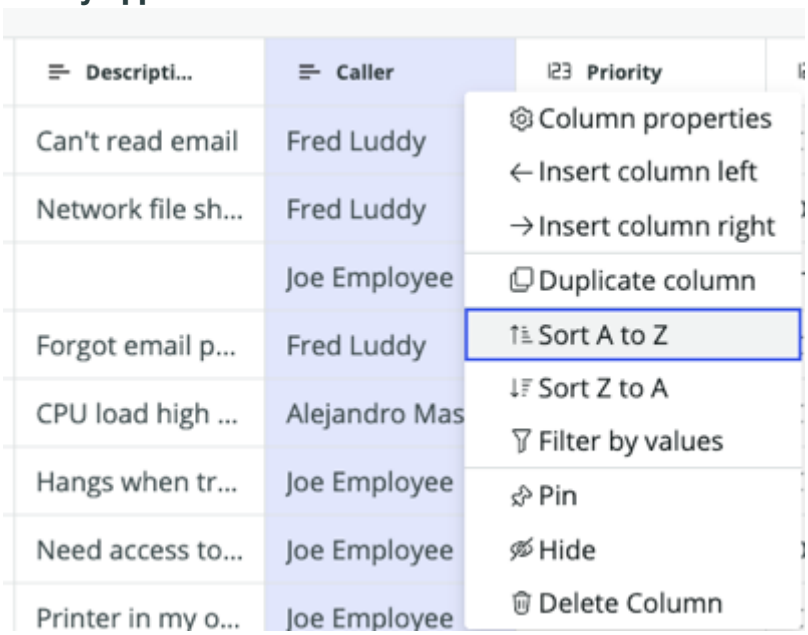
Procedure

1. Navigate to the **Data** tab.

   Note:
   Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Hover over a column header that contains the data by which you would like to sort the list of records, and then select the **Additional actions** icon ( ) to launch a column-specific list of actions (e.g., sort records by Approval Date in ascending order).
3. Select from the following options to choose whether to sort the data for the column in ascending or descending order.
   - Sort A to Z or Sort Oldest to Newest (ascending order)
   - Sort Z to A or Sort Newest to Oldest (descending order)

Sort by Approval Date



The list of records will then display in the order you chose.

Show matching data records

Show a filtered list of data records that contain matching data for a selected field within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

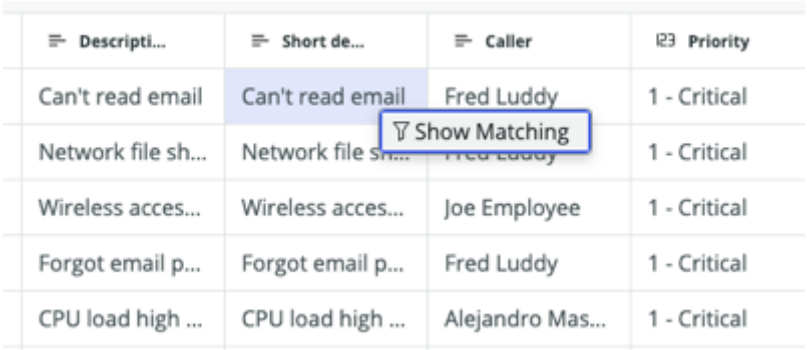
Procedure

1. Navigate to the **Data** tab.

> **Note:**
> Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Select and hold (or right-click) the data record in a cell to create a filtered list of records that contain the matching data, (e.g., show all records where **Short description** is "Can't read email".).

Show matching data records



A filtered list of records with matching data displays and a new filter query is added and accessible when you click the **Filter** button at the top of the grid.

Note:
To remove the filtering you just added, select the **Filter** button, and then delete the new filter condition from the list. See Filter your data records.

Filter your data records

Display only the data records you want to see in your list by setting up a conditional query within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.

   Note:
   Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Select **Filter** above the spreadsheet to launch a filter query.
   You can also select **Filter by value** from a specific column's **Additional actions** menu ( ) to pre-select that column's data for your query.

   Note:
   When a number appears to the right of the **Filter** button, it means that there are existing filters set for the view.

   The Filter window displays (along with any filter criteria that has already been set). In this case, an existing filter is set to display records where **Age** is 35.

Filter data

3. Enter your filtering query to specify which data records you want to display.
   - Select the **New condition set** button to enter a new filter condition which is connected by an OR statement.
   - Select the **or** or **and** buttons to extend the existing filter set with the desired boolean.
   - Select the **Delete** icon to remove a filter condition row.
4. Select **Apply** to apply the filter condition to your view.
   The filtered list of records will then display based on the criteria you chose.

Edit column field properties

Edit field properties such as the field label, type, attributes, and default values within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

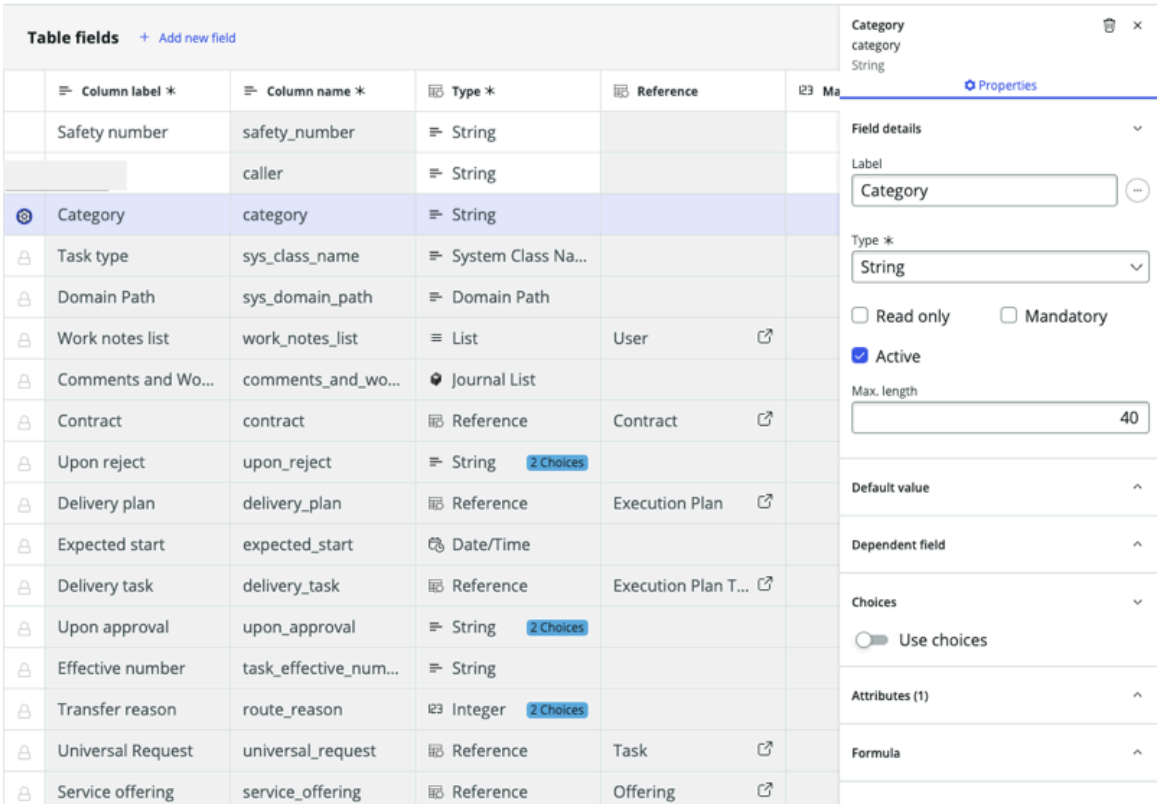
Procedure

1. Navigate to the **Data** tab.

   Note:
   Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Hover over the desired column header, and then select the **Additional actions** icon ( ) to launch a column-specific list of actions.
3. Select **Column properties**.
   The Properties pane displays for the selected field column.

Properties pane



4. Edit the desired field property for the column.
   See Field configuration in Table Builder for more information on field properties.
5. Select **Save**.

Add new columns

Add new field columns to a selected table within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.

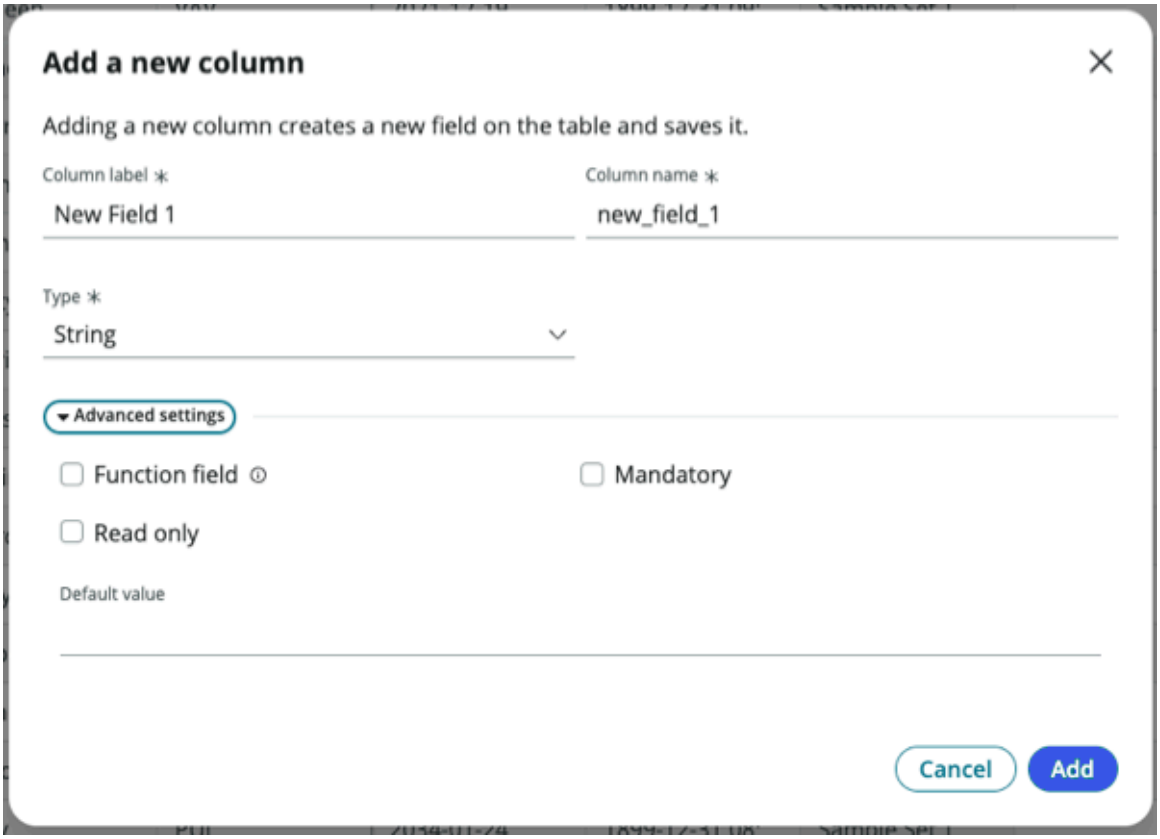
   Note:
   Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Select the **+ Add field** button.

   Note:
   You can also select the **Additional actions** icon ( ) to launch a column-specific list of actions, and then select **Insert column left** or **Insert column right** to insert the new column in a specific place within the table.

   The Add a new column window displays.

Add new column



3. Enter the following information for your new field column.

4. **Optional:** Select **Advanced settings** to expand it and enter any additional properties for your new field column.

5. Select **Add** to add the new field column to your table.

Duplicate an existing column

Duplicate an existing field column within a table shown in Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

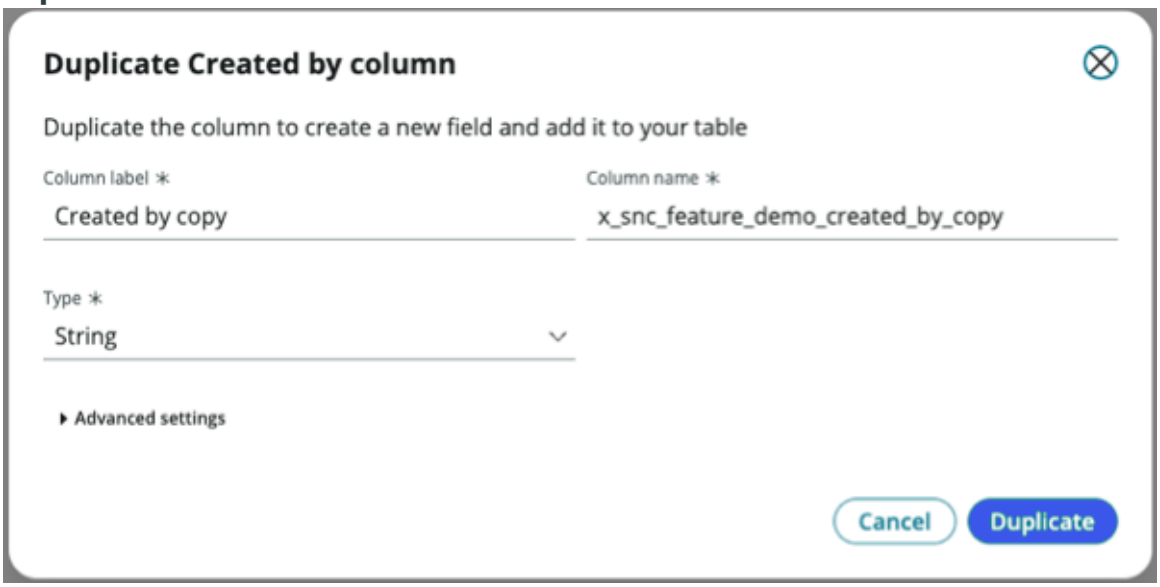
1. Navigate to the **Data** tab.

   Note:
   Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Hover over the header for the column to duplicate, and then select the **Additional actions** icon ( ) to launch a column-specific list of actions.
3. Select **Duplicate column**from the menu.

The text "copy" is automatically appended to the existing column label.

Duplicate column



4. Edit the **Column label** to the desired display name for the duplicated field column.
5. Edit the **Type** for the field column as desired.

   See Field types.
6. Select **Duplicate**.
   A new column displays in the view.

Reorder columns

Change the order in which your columns are displayed within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

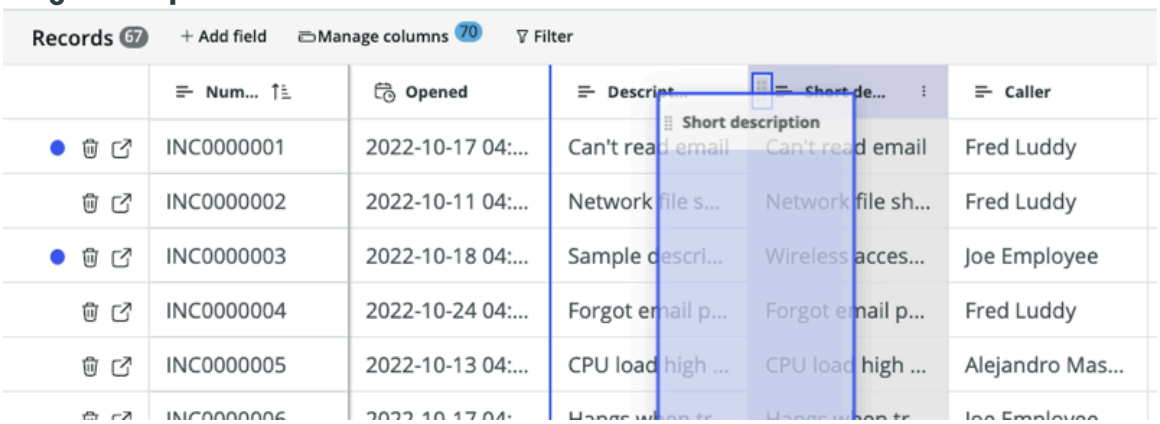
Procedure

1. Navigate to the **Data** tab.

   Note:
   Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Click to select the **Column handle** icon ( ) for a column and then drag the column to the desired locations in the table.

Drag and drop columns



Note:
If your table contains many table columns, select the **Manage columns** menu to view a compact list of column names, and then visually drag and drop within this list to order your table columns.

Set column visibility

Choose whether to hide or show specific field columns within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.

Note:
Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Choose one of the following options.

Note:
A number displays to the right of the **Manage columns** button to show how many columns are active in the view.

Pin columns

Pin specific columns to always display on the left when you scroll to the right in a table with a large number of field columns within Table Builder **Spreadsheet** view. Pinned columns are saved for each individual user, not globally. The next time a user logs in, any columns they pinned will be shown.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.

   Note:
   Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Choose one of the following options to pin or unpin a field column.

   Note:
   You can also view which columns are pinned in the view by clicking the **Manage columns** button. Pinned columns will have a pin icon displayed next to the right of the listed column.

Delete columns

Delete a field column for a selected table within Table Builder **Spreadsheet** view.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.

   Note:
   Spreadsheet view displays by default. You can also access Spreadsheet view from the Additional actions menu ( ).

2. Hover over a column header you want to delete, and then select the **Additional actions** icon ( ) to launch a column-specific list of actions (e.g., sort records by Approval Date in ascending order).
3. Select **Delete column**.
   A confirmation window displays.
4. Select **Delete** to delete the field column from your table.

   Note:
   If there are data records stored for the column you want to delete, you will not be able to delete the column until you have removed the records stored for this field.

Fields view in Table Builder

Modify the table properties and manage the table fields by using the default Fields view on the **Data** tab in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

Use the following procedures to edit tables using **Fields** view in Table Builder.

Add a table column in Table Builder

Store more information in a table record by adding a table column.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

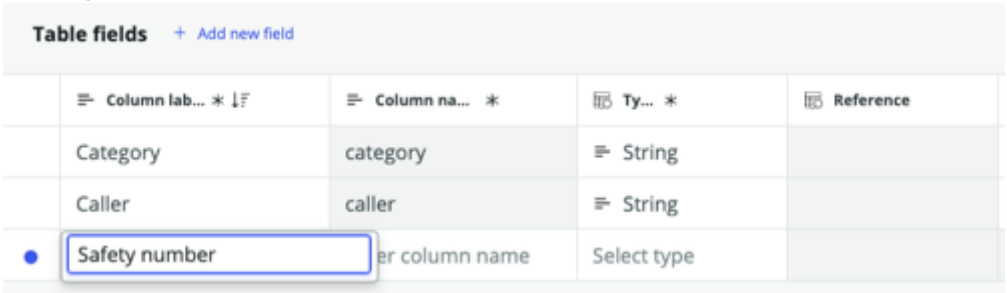
Procedure

1. Navigate to the **Data** tab.

   Note:
   Fields view displays by default. You can also access the Fields view from the Additional actions menu ( ).

2. Select **+ Add new field**.
3. On the resulting blank row, fill in the desired properties for the new data table column.

Adding a new field to a table



4. Select **Save**.

Result

If you preview the table, you can see the new column that was added to your table. For more information on previewing a table, see Preview your data in Table Builder.

Delete a table column in Table Builder

Delete a column from your table using Table Builder.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.

   Note:
   Fields view displays by default. You can also access the Fields view from the Additional actions menu ( ).

2. Next to the table field row you want to delete, hover over the leftmost column, and select the Open side panel icon ( ).

3. At the top right corner of the side panel, select the trash icon ( ) to delete the field.

   Note:
   Your table includes several default columns (including Created and Updated), that you can't delete. Also, you can't delete columns that are extended from another table or columns where data records are already present. To delete the column, you must first delete the existing data.

4. Select **Delete** in the confirmation dialog.

Result

If you preview the table, you can see that the table column is no longer used in the table. For more information on previewing a table, see Preview your data in Table Builder.

Change a column label in Table Builder

Modify the label that uniquely identifies the data that is stored in it to users.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

About this task

A column label is the text that's visible to your users when they're reading or updating application data. For example, if you create a form for requesters to fill in, each column label appears as a form field label.

Use a column label that describes the specific information that you expect users to enter. For example, to prompt a user to add comments, you would use the column label "Comments" instead of "Text".

Procedure

1. Navigate to the **Data** tab.

   Note:
   Fields view displays by default. You can also access the Fields view from the Additional actions menu ( ).

2. Change a column label in the Column label column, by updating the text in the corresponding row.

   For example, to change the label of the "Office location" column, you would select the cell that says "Office location" and enter a different label.

The column label updates automatically after you select another cell.

3. Select **Save**.

Changing a table column name in Table Builder

Change the database name for the column. You may opt to change the column name manually if a reviewer finds that the column name in the database was causing issues in testing.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

About this task

Note:
You can't change the column name for existing fields that have information in them saved already.

A column name is the text that an administrator uses to track column data in the database. A column name is created automatically after you enter a column label. In Table Builder, if you change the column label, the column name also changes automatically.

Procedure

1. Navigate to the **Data** tab.

   Note:
   Fields view displays by default. You can also access the Fields view from the Additional actions menu ( ).

2. Change the column name in the Column name column by updating the text in the corresponding row.

   For example, to change the name of the "Office location" column, you would select the cell that says u_office_location and enter a different name.

   The column name updates automatically after you select another cell.

3. Select **Save**.

Change a column type in Table Builder

Change the type of field that will store information for rows in a table column. For example, if your table is tracking dates, you would want to select the Date column type.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.

   Note:
   Fields view displays by default. You can also access the Fields view from the Additional actions menu ( ).

2. Select a cell and clear the text from the **Type** column.

3. From the list, select a column type.

   You can search the list by entering new text. For example, to find all the string types, you would enter `string`.

4. If required, define the column type with more information.

   For example, if you select **Choice**, define the list of choices that users can select from.

   The column type updates automatically after you select another cell.

5. Select **Save**.

Edit field properties in Table Builder

Edit field properties such as the field label, type, and default values in Table Builder.

Before you begin

Launch Table Builder. For more information, see Accessing Table Builder.

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.

   Note:
   Fields view displays by default. You can also access the Fields view from the Additional actions menu ( ).

2. Select a field you want to modify in the table.

   For information about basic field properties, see Field Configuration.

3. Next to the field you want to edit, hover over the leftmost column, and select the Open side panel icon ( ).

4. Make your changes to the field properties.

   For information about modifying a field's UI policies, see Modify a UI policy in Table Builder.

   Note:
   The **Config** tab on the right stays open even if you switch between the **Data** and **Forms** tabs.

5. Select **Save**.

Add a default value to a table column in Table Builder

Define the default value to populate a table column automatically after a user creates a record.

Before you begin

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.

   Note:
   Fields view displays by default. You can also access the Fields view from the Additional actions menu ( ).

2. Add the default value of a column in the **Default value** column by updating the content in the corresponding row.

   You define the default value by selecting a cell and then entering the value. The value that you enter depends on the column type. For example, if the column requires a choice, you would select one of the choices that you defined in the Type column.

   The column name updates automatically after you select another cell.

3. Select **Save**.

Select a column as the table display value in Table Builder

Select a column value that will be displayed in the reference fields of other tables. For example, the Opened by column of the task table could refer to a specific column in the user table so it can display a user name.

Before you begin
Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

About this task
A reference field shows the display value of the table to which it is referring. For example, the Opened by column of the task table refers to the user table. Because the display value of the user table is the user name, the **Opened by** field shows something like "Beth Anglin" or "Joe Employee". When you select a display value, choose the table column that would act as an appropriate title for individual records.

Only one column can act as the display value for a table.

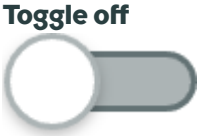
Procedure

1. Navigate to the **Data** tab.

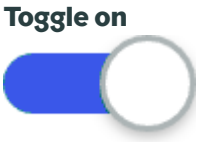
   Note:
   Fields view displays by default. You can also access the Fields view from the Additional actions menu ( ).

2. Check that all selections are cleared from the **Display** column.
   To clear a selection, toggle off the switch as shown in the following example.

   Toggle off

3. From the Display column, select one row to act as the table display value.
   To select a row, toggle on the switch as shown in the following example.



4. Select **Save**.

Schema view in Table Builder

Use **Schema** view in Table Builder to explore data relationships for your application data.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

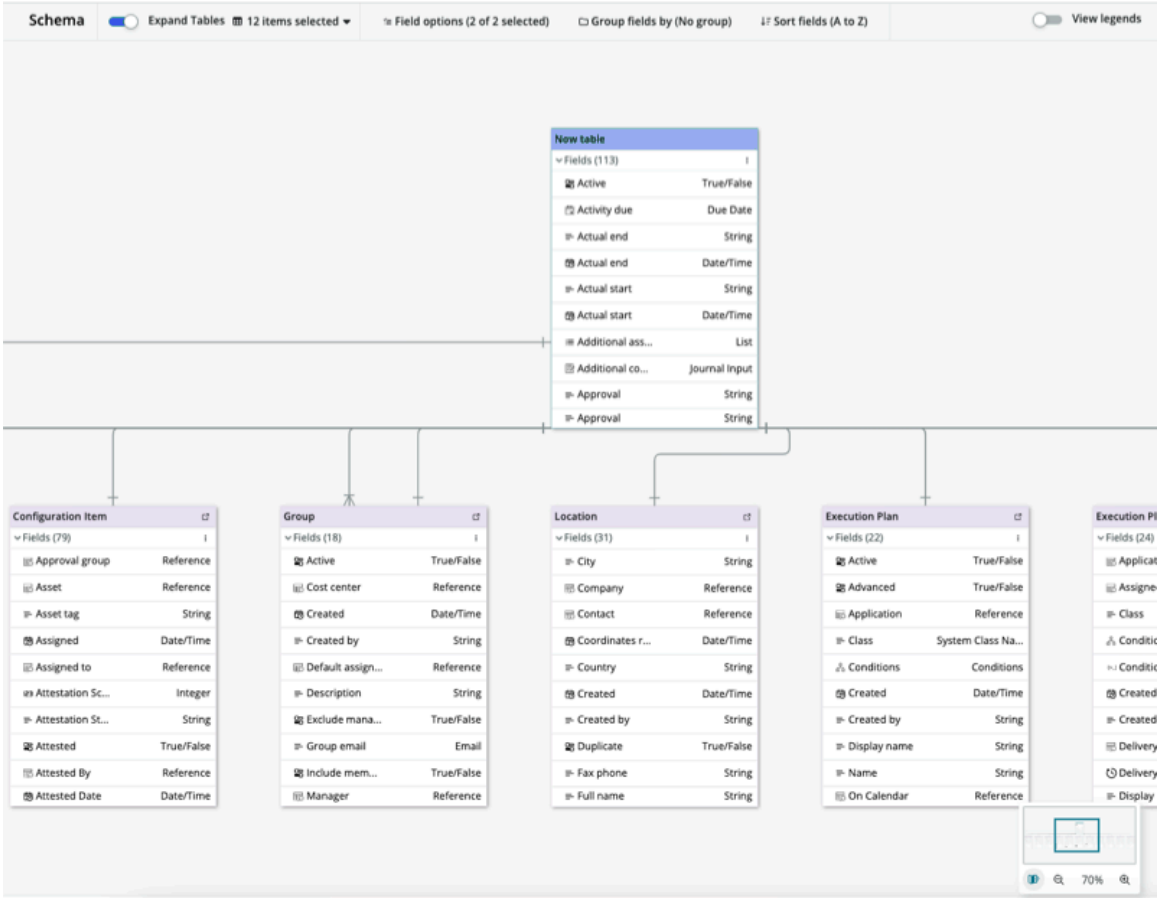
Procedure

1. Navigate to the **Data** tab.

   Note:
   Spreadsheet view displays by default.

2. **Optional:** Select **Schema** from the additional actions menu ( ) to access Schema view.
   A visual depiction of the table and its relationships to other tables displays on the screen.

Schema view



3. Navigate through the schema diagram by dragging the canvas with your mouse.
4. You can perform the following additional actions on this read-only screen.

Option	Description
Navigation overview controls	Select the **Overview toggle** icon ( ) in the lower right corner to toggle the navigational overview controls on or off. This navigational control enables you to perform the following actions: ○ When the schema diagram is zoomed in, you can drag the rectangle showing your view around by clicking and dragging it to focus it on the part of the diagram you want to review. ○ Select the **Zoom in** icon ( ) to zoom in. ○ Select the **Zoom out** icon ( ) to zoom out.
Show / hide table fields	The **Expand tables** toggle is toggled on by de fault to show all the fields within the displayed schema tables. To hide the fields within the displayed schema tables, select **Expand ta bles** to toggle it off.
Show / hide tables	Use the **Items selected** menu to view a list of application and non-application tables that

Option	Description
	are displayed in the schema diagram. You can select the table name to remove the selection checkmark, which hides the table from dis playing in the diagram. To show the table, se lect the table name again to toggle the check mark back on.
Set field options	Use the **Field options** menu to control how fields display in the schema diagram. ○ Select **Extended fields** to display extended fields within schema tables. ○ Select **Field type** to show the type of field for each field displayed in the schema ta bles.
Group fields	Use the **Group fields by** menu to group fields within the displayed tables (e.g., grouping fields by type). Choose from the following grouping options: ○ No group ○ Extended ○ Reference ○ Type
Sort fields	Use the **Sort fields** menu to sort fields within the displayed tables. Choose from the follow ing sorting options: ○ Alphabetically (A to Z) ○ Alphabetically (Z to A)
View legends	Select **View legends** to view information for in terpreting the displayed schema diagram.
Table-specific options	Select the **Additional options** icon (⋮) for a ta ble to control how fields in the selected table are displayed, grouped, and/or sorted similar to the top-level options described above.
View in Table Builder	Select the **View Table** icon (↗) to view the se lected table in Table Builder.

Preview your data in Table Builder

See what your table looks like after you've edited the table columns.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Role required: none

About this task

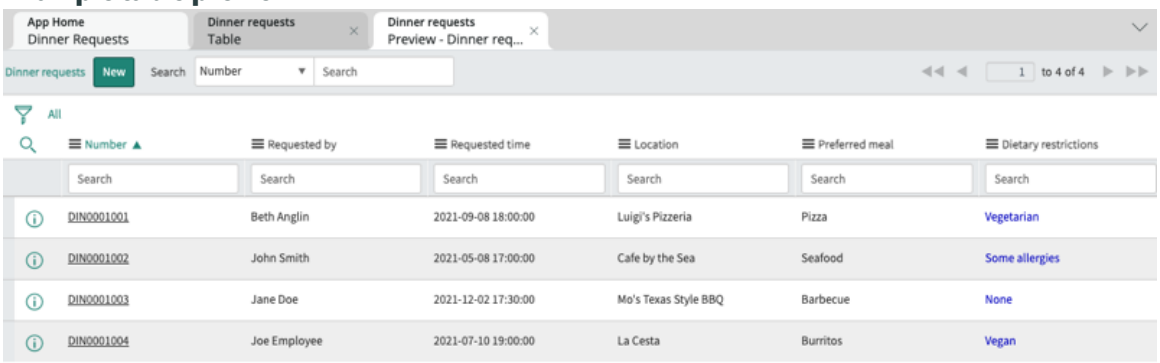
Each row in Table Builder represents a column from your table. Previewing the table enables you to see what the columns look like in their proper place. You can also test the experience of creating records on your table.

Note:
The table which you are editing must have at least one role that is associated to it or preview is disabled.

Procedure

1. On the **Data** tab, select **Preview**.
2. From the new tab, review the table preview as shown in the following example.

Example table preview



What to do next

If your table doesn't work as expected, continue editing in Table Builder. Then, preview the table again.

Delete a table in Table Builder

Delete a table from your application so that it no longer appears in the app dashboard.

Before you begin

Note:
Parent tables can't be deleted until all existing child tables are deleted.

Role required: Users must have canDelete access to **sys_db_object** or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Data** tab.
2. Select the menu icon (•••), and then select **Delete Table**.
3. In the dialog box, enter `delete` and then select **Delete**.
4. Select **Save**.

Result

The table is no longer available in the app dashboard.

Edit table properties in Table Builder

Change table properties such as the table label or other settings so that you can make a table extensible or add record numbers by using Table Builder.

Before you begin
Role required: none

> **Note:**
> User must have canWrite access to *sys_db_object* or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

About this task
By editing table properties, you can relabel your table, make your table extensible, or add record numbers. You can also set application access settings.

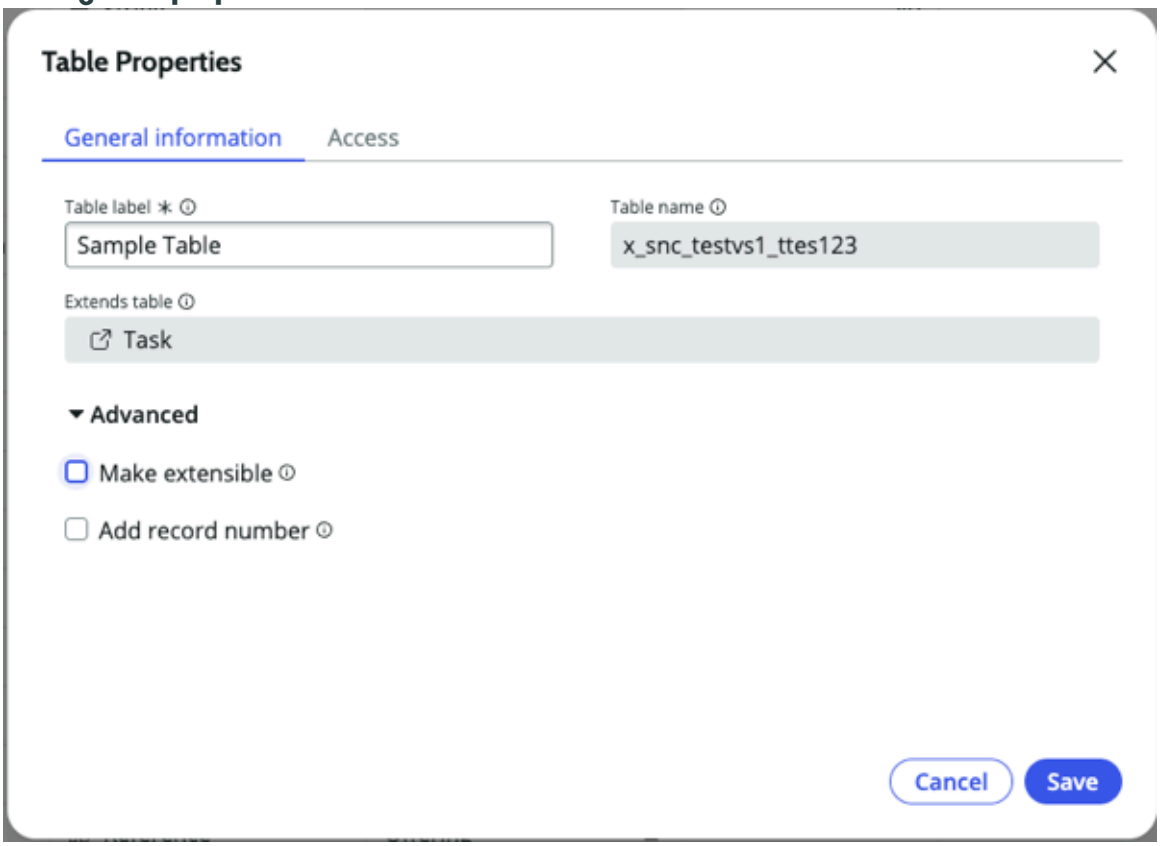
Making a table extensible means that you are enabling new tables to share columns from your table. For example, if you created an "Office location" column in your table, you can allow new tables to use the "Office location" column also. For more information on table extensions, see Table extension and classes.

Adding record numbers means that a tracking number is created automatically for each new table entry. For example, a new entry to the Travel Requests table would get a record number like TRV1234567. You can use this number to find table records more easily.

Application access settings determine whether script objects from other applications can access the table in your application. You can give these script objects access to read, create, update, or delete records on your table. Alternatively, you can disable access to your table from other applications. For more information on the application access controls, see Table design and runtime settings.

Procedure
1. Navigate to the **Data** tab.
2. Select **Properties**.
3. In the dialog box that appears, update the table properties as shown in the following example. For more information on these properties, see Table properties.

Editing table properties



4. To close the dialog box, select **Save**.
5. Select **Save**.

Forms in Table Builder

On the **Forms** tab in Table Builder, you can visually create, configure, and customize the different form views for your users using the form editor. The views that you define contain the elements that appear when a user opens that form or list.

Overview

A form view defines the elements that appear to a user when the user opens a form or list. Each form has a default view, but you may want to create several different form views for different groups of users.

In Table Builder, you can customize multiple views for a form for the different users who input data or view the form. You can also customize the default form view for everyone.

Explore the following major areas when you configure a form or list:

- Form view selection
- All form views list
- Form layout
- Form fields
- Form annotations

- Formatters
- Embedded lists

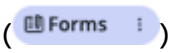
Basic form navigation

The **Forms** tab consists of the following four major areas:

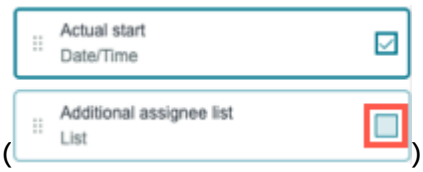
- Top navigation menu
- Add form elements panel
- Form editor
- Configuration panel

The following table contains a list of some of the other basic navigational elements within the **Forms** tab in Table Builder.

Navigational elements in the Forms tab

Navigational element	Description
Forms additional actions list ()	The Additional actions list (vertical ellipsis) on the **Forms** tab provides a quick way to perform any of the following actions: • Add a new form view • Duplicate the selected form view • Configure a related list • Configure a UI action • Navigate to a filterable, sortable list of all form views by selecting **All form views**. See All form views list. Changes to the form made in the editor are committed to the selected view. **Note:** The additional actions menu (vertical ellipsis) is visible only when forms are accessed throughServiceNow Studio or App Engine Studio, not throughServiceNow AI Platform.
Form view list ()	Select a form view from this list or use the Additional actions menu (vertical ellipsis) to add a view. Changes to the form made in the editor are committed to the selected view.
PDF extractor ()	If a PDF is associated with the selected data table, select this element to launch the PDF extractor tool and view the PDF. See Use a PDF to create data tables.

Navigational elements in the Forms tab (continued)

Navigational element	Description
	Note: This feature is only available if your licensing entitles you to "exclusive low code capability" and you have Table Builder for App Engine installed. Contact your Solutions consultant for more information.
Form element search bar ()	Enter keywords here to filter the list of form elements.
Multiple selection check box 	When you point to a form element in the Add form elements panel, you can select this option if you want to select and drag multiple fields to the form editor at once.
Add form element icon ( )	When you point to a form element or section, select this icon to add a new section or element above or below the existing element.
Move element icon ( )	When you point to a form element, this icon displays on the left edge. Select this icon, and then drag the element to move it to the desired location.
Delete form element icon ( )	When you point to a form element in the editor, you can select this icon to remove it from the form view.
Move section arrow icon ( )	When you select a form section, you can select the up or down arrow to move it up or down on the form view.
Delete section icon ( )	When you select a form section, you can select this icon to remove the section from the form view.
Open panel icon ( ) Close panel icon ( )	Selecting the Open panel icon expands the panel to the right. Selecting the Close panel icon collapse it.

Form view selection

Select a form view to work with from the form view list in the form editor.

Note:
You can also select a form view from the Additional actions menu on the **Forms** tab.

See Choose a form view in Table Builder.

All form views list

Select **All form views** from the Additional actions menu to the right of the **Forms** tab to browse a filterable list of form view cards (vertical ellipsis). This page allows you to create complex filtering and sorting if you have a great deal of form views to navigate.

You have the following options for finding a desired form view if you have a large list of them.

All form views list

Option	Steps
Search	Type a keyword in the **Search** box. The list of cards will be filtered as you type.
Sort form views	Select an option from the sorting list to sort the form view cards. • Form name (a to z) • Form name (z to a) • Updated by (a to z) • Updated by (z to a) • Updated (recent to last) - Default option • Updated (last to recent)
Filter forms by setting up filter criteria	**1.** Select the filter icon ( ). **2.** Add your filter condition criteria including any AND/OR logic. **3.** To add additional conditions, select **+ New condition set**. **4.** Select **Apply**.

Form layout (sections)

A view is composed of sections that group the data elements that you want displayed.

You can add multiple sections and change the layout of each section into one or two columns. A section is where you can group the data elements that you want to display for a form view. These elements include form fields, annotations, formatters, and embedded lists.

See Customize your form layout in Table Builder.

Form fields

When you select a table in Table Builder, the available fields are displayed in the form elements panel. You can create fields for the table or configure the fields and their properties.

By dragging these fields into the form editor, you can then visually arrange these fields on the form view you have selected.

See Add fields to a form layout in Table Builder.

See Modify field properties in Table Builder form editor.

Form annotations

A form annotation is an additional piece of information on a form, such as a line or paragraph of text.

For example, you may want to add instructional text for a particular section on the form. By using Table Builder, you can add a form annotation to the view that displays on-screen instructions on how a user should enter information.

See Create form annotations in Table Builder.

Formatters

A **Formatter** is a read-only element in a form that displays additional information about the selected table record. Unlike fields, formatters are not editable.

See Formatters.

By using Table Builder, you can visually drag any of the displayed formatters onto the form editor and arrange them.

See Add formatters in Table Builder.

Embedded lists

You can use an embedded list to display the data for a selected related list on a form. A related list shows the records in the tables that have relationships to the current record. For example, for a form view that references a Problem table, you might want to display a list of locations that are impacted by a selected problem.

With Table Builder, you can visually drag a list from the **Embedded Lists** tab of the form elements panel onto the form editor, and then arrange them.

By adding an embedded list to a form, you enable your users to view or edit the related list without needing to navigate away from the form. Changes to the data in the list are saved when the user saves the form.

See Add embedded lists in Table Builder.

Choose a form view in Table Builder

Choose existing views, duplicate views, or create entirely new views for form users in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

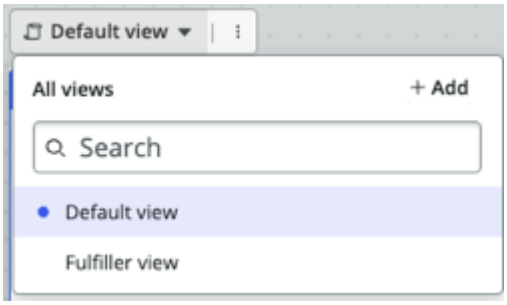
Note:
To understand how to approach customizing your forms, review Table Builder workflow and navigation.

Role required: personalize_form or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

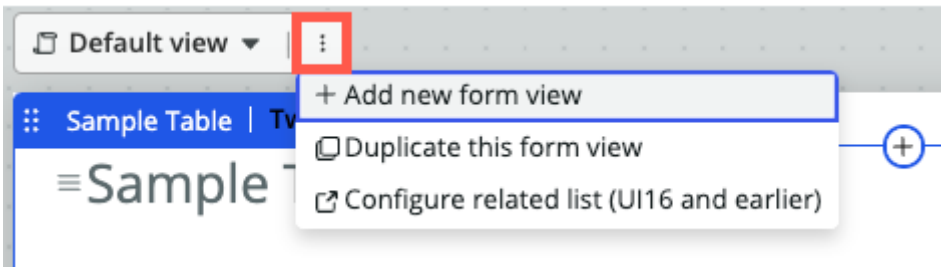
1. Navigate to the **Forms** tab in Table Builder.
   The "Default view" is selected.

   View list

2. You can modify the default view selected by using the View list or select another view.
   You can also add a view or choose one the other form view actions from the Additional actions menu.

   Additional actions menu

   - Add a new view to customize by clicking **Add new form view** from the view list or by selecting an option from the Additional actions (⋮) menu.
   - Duplicate an existing view to customize by selecting a view and then choosing **Duplicate this form view** from the Additional actions (⋮ menu.
   - Configure a related list for the selected form view by selecting this option from the Additional actions (⋮) menu.

   Note:
   You can also choose a view from the Additional actions list to the right of the **Forms** tab or navigate to the **All form views** list to filter and sort a large list of form views. See All form views list.

3. With the view selected, you can modify the form view in the following ways:
   - Add fields to your form. See Add fields to a form layout in Table Builder.
   - Customize the layout of your form (sections). For more information, see Customize your form layout in Table Builder.
   - Add non-field elements that display data on your form. For more information, see Add formatters in Table Builder.

- Add instructional text or other text to your form. For more information, see Create form annotations in Table Builder.
- Add related lists to your form. For more information, see Add embedded lists in Table Builder.

Note:
To customize the elements that display on a form, see Forms in Table Builder.

Customize your form layout in Table Builder

Visually design the look of the forms your users see by customizing form views in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Note:
To understand how to approach customizing your forms, review Table Builder workflow and navigation.

Role required: personalize_form or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Forms** tab in Table Builder.
2. Choose a view to work with.
   For detailed information on how to choose a view for a form, see Choose a form view in Table Builder.
3. Customize the form layout that displays in the form editor by performing the following actions.
4. To rearrange the sections within the form, in the section header, select the section and then drag it to the desired location on the form or use the up and down arrow icons.
5. Select **Save**.

Add fields to a form layout in Table Builder

Visually add and arrange fields within a form layout to create a form that matches your requirements using Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Note:
To understand how to approach customizing your forms, review Table Builder workflow and navigation.

Role required: personalize_form or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

About this task

Procedure

1. Navigate to the **Forms** tab in Table Builder.
2. Choose a view to work with.
   For detailed information on how to choose a view for a form, see Choose a form view in Table Builder.
3. To display a list of the existing fields for the selected table, in the add form elements panel, select **Fields**.

   Note:
   You can also launch the Add form elements panel by selecting the Add (+) icon above any existing element in the form editor.

4. Customize the fields that display on the selected form view by performing one of the following actions.

   Note:
   Certain field types are only supported in Core UI. Fields are flagged with an exclamation icon (!) to indicate an unsupported type.

5. Select **Save**.

Modify field properties in Table Builder form editor

Modify basic field properties for the fields that display for users in a form view from within the **Forms** tab in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Note:
To understand how to approach customizing your forms, review Table Builder workflow and navigation.

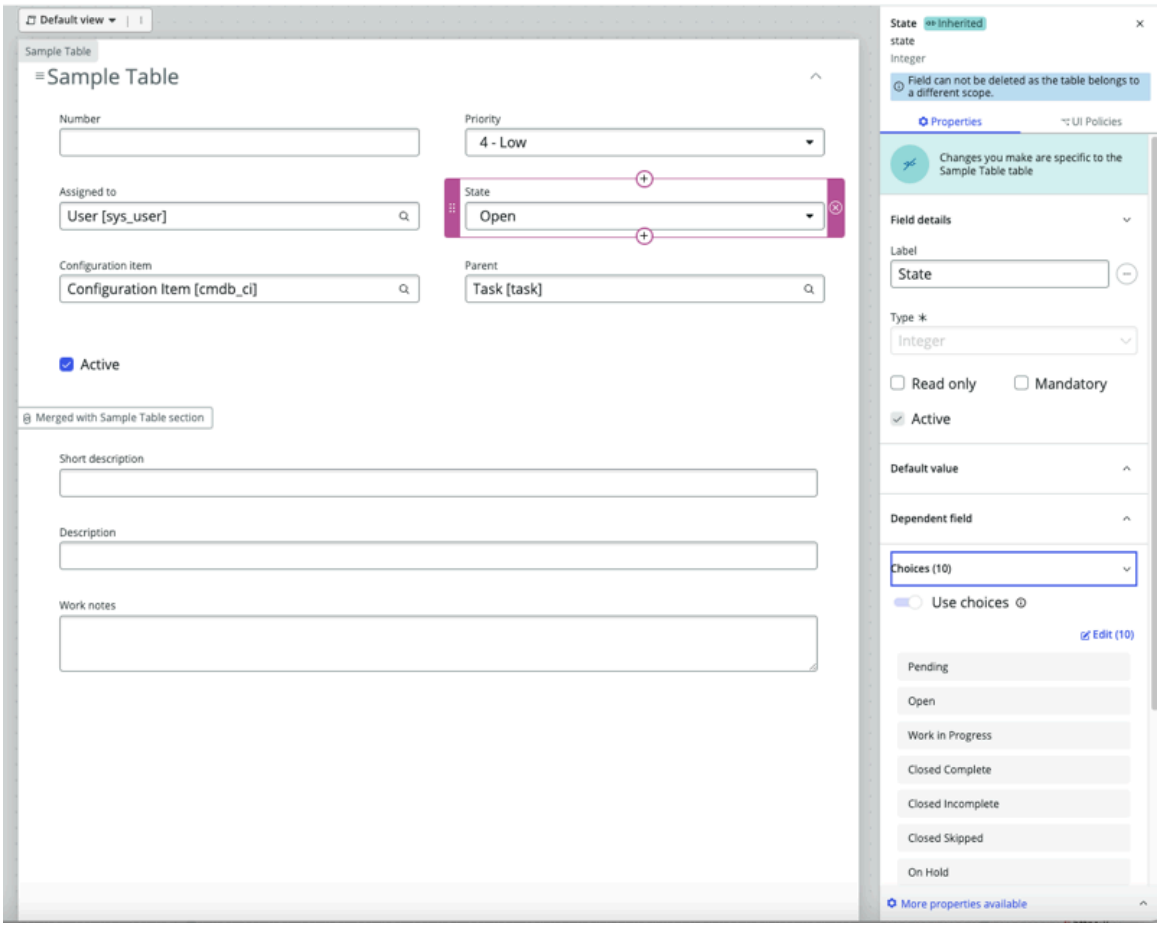
Role required: personalize_form or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. On the **Forms** tab, choose a view to work with.
   For detailed information on how to choose a view for a form, see Choose a form view in Table Builder.
2. In the form editor, select a field to highlight it.

The following example shows that the panel on the right displays the **Properties** tab. This tab shows the basic properties of the selected field.

Properties tab



3. Edit the basic field properties as needed for the selected field.
   The sections displayed vary depending on the field type. For a description about these fields, see Field configuration in Table Builder.

   Note:
   Below the **Properties** tab, select **More properties available** to launch the full dictionary entry for the selected field.

4. To view any UI policies that are set up for the selected field and make edits to that field, select the **UI Policies** tab.
   For more information about the policies and properties, see Policies and rules properties in Table Builder.
5. Select **Save**.

Create form annotations in Table Builder

Add instructional text and other design elements to your forms by using form annotations in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Note:
To understand how to approach customizing your forms, review Table Builder workflow and navigation.

Role required: personalize_form or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

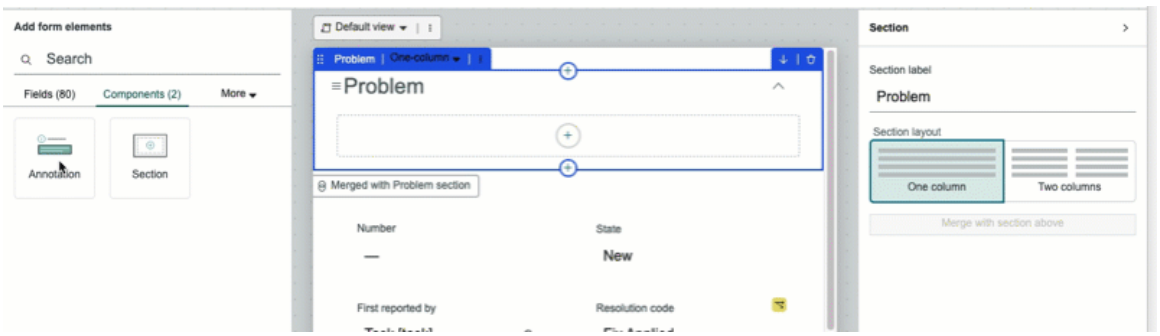
Procedure

1. Navigate to the **Forms** tab in Table Builder.
2. Choose a view to work with.
   For detailed information on how to choose a view for a form, see Choose a form view in Table Builder.
3. To display a list of the available components that you can add to your form view, in the Add form elements panel, select **Components**.

   Note:
   You can also launch the Add form elements panel by clicking the Add (+) icon above an existing element in the form editor.

4. Select the **Annotation** elements, and then drag it to the location in the form editor where you'd like to position the annotation text as shown in the following example.

   Add annotation

5. Select the **Annotation type**.
6. Select an option for **Annotation text**.
7. Enter the desired annotation text in the field that displays in the Annotation panel.
8. Select **Save**.

Add formatters in Table Builder

You can add form elements that display non-field information to users by adding a formatter in Table Builder. For example, you may want to display a **Comments** field where users can add comments to a displayed form.

Before you begin

Note:
Formatters are only supported in Core UI and earlier.

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Note:
To understand how to approach customizing your forms, review Table Builder workflow and navigation.

Role required: personalize_form or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Forms** tab in Table Builder.
2. Choose a view to work with.
   For detailed information on how to choose a view for a form, see Choose a form view in Table Builder.
3. To display a list of the available components that you can add to your form view, in the Add form elements panel, select **Formatters**.

   Note:
   You can also launch the Add form elements panel by clicking the Add (+) icon above an existing element in the form editor.

4. Select a formatter, and then drag it to the location in the form editor where you'd like to place it.

   Note:
   Formatters in forms are only supported in Core UI and earlier interfaces.

   For information on how you can add a formatter to a form, see Formatters.
5. Select **Save**.

Add embedded lists in Table Builder

Add an embedded list using Table Builder to display data for a selected related list on your form. A related list shows the records in the tables that have relationships to the current record. For example, for a form view that references a Problem table, you might want to display a list of locations that are impacted by a selected problem.

Before you begin

Note:
Embedded lists are only supported in Core UI and earlier.

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Note:
To understand how to approach customizing your forms, review Table Builder workflow and navigation.

Role required: personalize_form or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. Navigate to the **Forms** tab in Table Builder.
2. Choose a view to work with.
   For detailed information on how to choose a view for a form, see Choose a form view in Table Builder.
3. To display a list of the embedded lists that you can add to your form view, in the Add form elements panel, select **Embedded lists**.

   Note:
   You can also launch the Add form elements panel by clicking the Add (+) icon above an existing element in the form editor.

4. Select an embedded list, and then drag it to the location in the form editor where you'd like to place it.

   Note:
   Embedded lists in forms are only supported in Core UI and earlier interfaces.

   For information on how you can add an embedded list to a form, see Embed a list within a form.
5. Select **Save**.

Preview your form

Previewing and saving your form in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Note:
To understand how to approach customizing your forms, review Table Builder workflow and navigation.

Role required: personalize_form or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

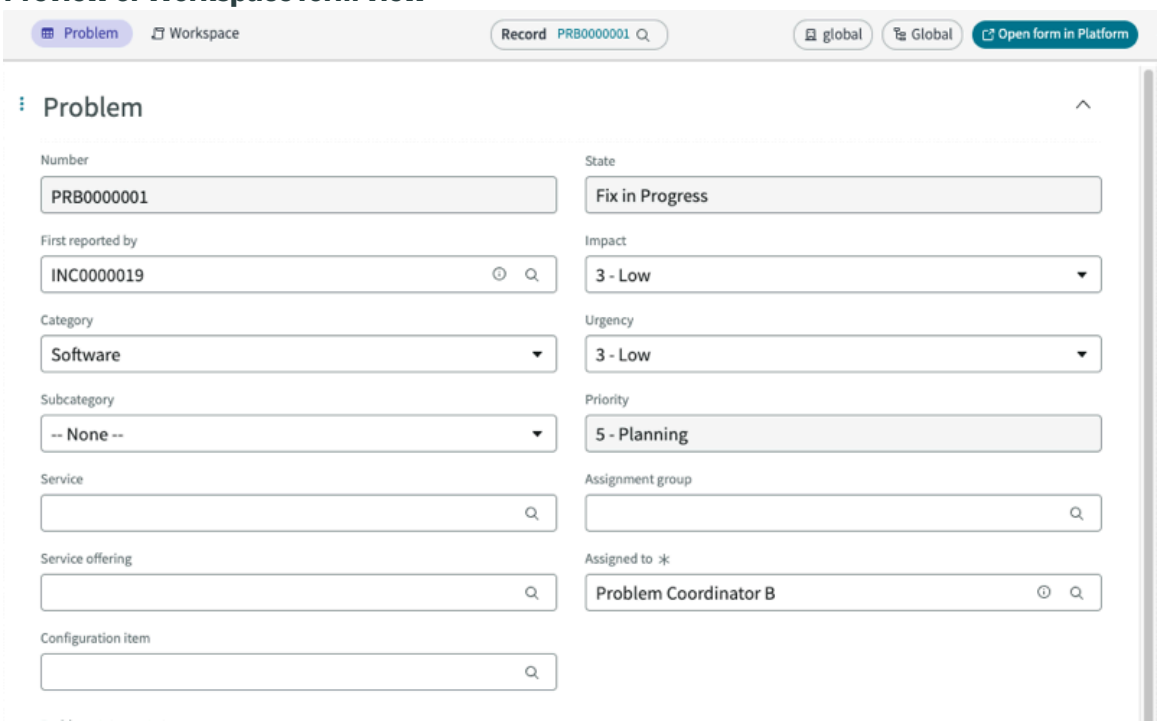
Procedure

1. Access Table Builder.
   - For information on viewing forms, see Launching Table Builder from UI Builder.
2. Select the **Forms** tab.
3. Select a form view.
4. Customize the form view as desired.
5. Save your changes by selecting **Save**.

> **Note:**
> Once you save your changes, you cannot undo them using the Undo function. The Undo icon ( ) is grayed out.

6. Select **Preview** to review your saved changes.
   The Preview window displays a record for your form. The Workspace form view is shown in the example.

Preview of Workspace form view



7. You can perform any of the following actions as you review your changes to better understand how your form will display for users.
   - When previewing a form, you can search for a new record to populate your form with by clicking the New record icon ( ). This will also let you add a new record if none exist so you can really see what the form will look like.
   - When previewing a form, you can open the form in the ServiceNow AI Platform by clicking the Open form in Platform icon (Open form in Platform).
8. When you are finished reviewing, close the browser window.

Policies and rules in Table Builder

Policies and rules define how information appears to users based on role and entries, as well as who can access tables and forms built in Table Builder.

Sections for working with policies and rules

The **Policies and rules** tab in Table Builder has several sections, but you can edit only UI policies within Table Builder. The other sections are read-only and contain links to manage their content in other parts of the ServiceNow AI Platform.

Administrative UI policy and rules sections

Section	Description	Editable within Table Builder
UI Policies	Dynamically change the behavior of information on a form and control custom process flows for tasks.	Yes.
Access control rules	Control access to the specified resource based on role, condition, and script criteria being met.	No.
Client Scripts	Create scripts that control how form fields appear based on defined criteria.	No.
Business Rules	Create business rules to accomplish tasks like automatically changing values in form fields when certain conditions are met.	No.
Workspace view rules	Define rules to control how users view workspaces based on criteria.	No.

To manage a rule or policy script outside of Table Builder, select for the entry. Or, you can view the full list of rules and scripts outside of Table Builder by selecting the **Full list** link for each section of the **Policies and rules** tab:

- **Full ACL list**
- **Full client scripts list**
- **Full business rule list**
- **Full workspace view rules list**

UI policies for tables and forms

Configure UI policies to adjust how forms appear based on roles and user input.

For example, you can use UI policies to make a field on a form read-only or required or hide certain fields depending on a user's role. Basic UI policies do not require any scripting. For more advanced actions, use the **Run scripts** option.

You can also use client scripts to perform all these actions, but for faster load times use UI policies when possible.

Create a UI policy in Table Builder

Configure UI policies in Table Builder to adjust how forms appear based on roles and user input.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

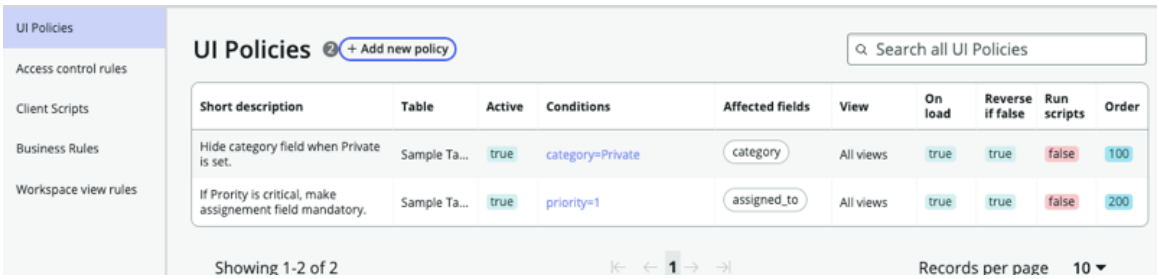
Role required: personalize_rules or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

About this task

You can create and edit UI policies only for applications that you own. For more information on application scope, see Domain separation and Table Builder.

Procedure

1. On the Table Builder screen, choose a table to work with.
2. Select the **Policies and rules** tab.
   Alternatively, you can add a UI policy for a field directly in the **Forms** tab of Table Builder in the right pane that appears when you select a field.
3. Select the UI Policies section to display a list of UI policies.
4. Select the **Add new policy** link to display the policy details.
   - If the link is inactive, you must change the scope that you're working with on the **Forms** tab.
   - If a field has policy and appears in the view, the active icon appears on the top corner of the field as shown in the following example.



5. On the form, fill in the fields for the Policy details section.
   For more information on field definitions, see Policies and rules properties in Table Builder.
6. In the When these conditions are met section, specify the conditions that, if fulfilled, cause the UI policy to be applied using the condition builder.
   To add multiple sets of conditions, select **New condition set**.
7. In the Do the following section, create field actions that should be performed when the conditions are met.
8. To finish adding the policy, select **Add UI policy**.

Result

After you add the UI policy, you can configure scripts and the related list actions for it. For more information on client scripts and UI policies, see Client scripts for UI policies.

Modify a UI policy in Table Builder

Edit UI policies to change how forms appear based on roles and user input in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Role required: personalize_rules or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

About this task

You can create and edit UI policies only for applications that you own. For more information on application scope, see Domain separation and Table Builder.

Procedure

1. In the UI policies section, on the **Policies and rules** tab, select the policy that you want to edit. You can also edit a field's UI policy directly in the form by selecting it in the **UI policies** tab of the right pane that appears when you select a field.
2. Change the UI policy fields and attributes as needed.
   For more information on field definitions, see Policies and rules properties in Table Builder.
3. Click **Update UI policy**.

Flows in Table Builder

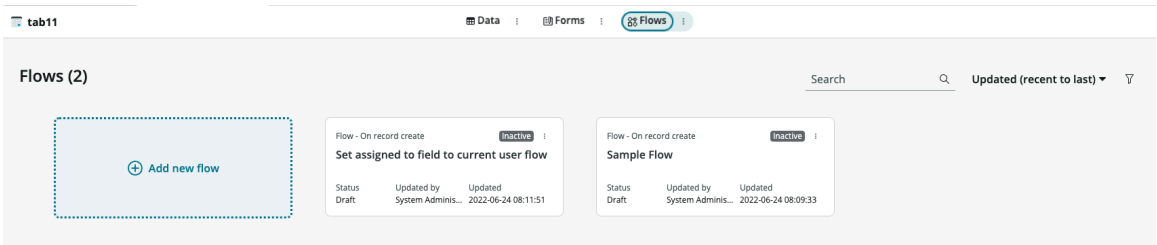
You can use the Workflow Studio functionality that is integrated within Table Builder to manage your table's record-based flows. You can manage these flows by using the **Flows** tab.

Workflow Studio is a ServiceNow AI Platform® feature that enables process owners to automate work. Build multi-step flows from reusable components without having to code.

The **Flows** tab within Table Builder is where you can view and work with record-based flows for a selected data table (that is, actions that are triggered when a table record is created or updated or both).

Flows tab

The following diagram shows flow cards in Table Builder.



Find a record-based flow

Use the **Flows** tab in Table Builder to search and filter a list of flows that are triggered when selected table records are updated, created, or both.

Before you begin

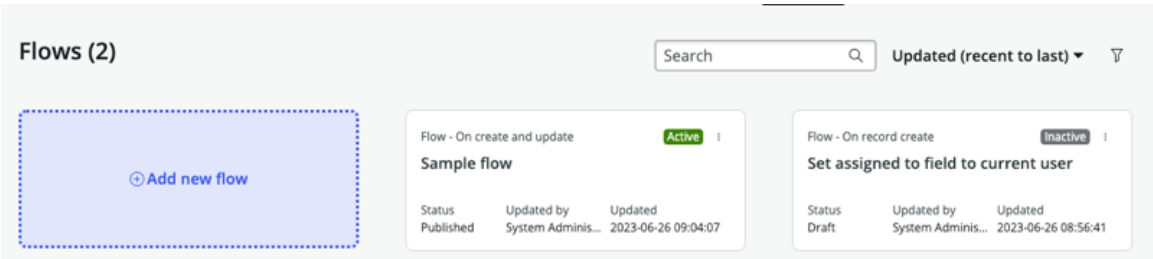
- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Role required: personalize_forms, personalize_dictionary, and flow_designer permissions (or relevant AES user role and delegated developer permissions). For more information, see Delegate developers using AES.

Procedure

1. Launch Table Builder as described in Accessing Table Builder.
2. Select the **Flows** tab.
   The flows triggered by the selected table are listed as flow cards.

Flow cards list



3. Filter or search the list of flow cards as needed to find the desired flow.

Find the desired flow card

Option	Steps
Search text	Type a keyword in the **Search** box. The list of cards will be filtered as you type.
Sort flows	Select an option from the sorting list to sort the flow cards. **a.** Flow name (a to z) **b.** Flow name (z to a) **c.** Updated by (a to z) **d.** Updated by (z to a) **e.** Updated (recent to last) - Default option **f.** Updated (last to recent)
Filter flows by setting up filter criteria	**a.** Select the filter icon ( ). **b.** Add your filter condition criteria including any AND/OR logic. **c.** To add additional conditions, select **+ New condition set**. **d.** Select **Apply**.

What to do next

- Select the flow card to edit your new flow using Workflow Studio. See Modify a record-based flow.
- Create a new record-based flow from an existing one. See Copy a record-based flow.
- Delete the flow. See Delete a record-based flow.

Add a record-based flow

Add flows that are triggered when a table record is updated, created, or both updated and created by using the **Flows** tab in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Role required: personalize_forms, personalize_dictionary, and flow_designer permissions (or relevant AES user role and delegated developer permissions). For more information, see Delegate developers using AES.

Procedure

1. Launch Table Builder as described in Accessing Table Builder.
2. Select the **Flows** tab.
3. Select the **Add new flow** button.
4. On the New Flow form, fill in the following fields for your new flow.

New Flow form

Field	Description
Name	Name to uniquely identify your flow. The system computes the internal name of the flow from the name.
Description	Description of your flow.

5. **Optional:** To enter advanced options for your flow, select **Show advanced options** on the New Flow form.

Advanced options

Field	Description
Protection	Option that you can select if the flow is read-only. You can only select a value when you create the flow in an application scope that you own.
Run as	(Optional) Option to specify the user that runs the flow. You can select the system user or the user who initiates the session. Select the user who initiates the session option when updates should come from the user who triggered the flow. For example, use this option when you want the incident record comments to come from the user who started

Field	Description
	the flow. Settings for the Run as option in a flow don't apply to child subflows. To create a flow that can run with a personal OAuth token, select the user who initiates the session option. If the user who is running the flow has a personal OAuth token, the flow runs with that token. For more information about creating a personal OAuth token, see OAuth 2.0 credentials. When flows run as the user who initiates the session, the system limits the flow actions by user access control list (ACL) restrictions. Ensure that security restrictions don't prevent users who trigger the flow from performing flow actions. Flows that are run by the initiating user also respect user-specific settings, such as the date/time formats. **Note:** Inbound email flows ignore this setting and always run as the user who initiates the session. To test access controls for an inbound email flow, impersonate a typical inbound email user and manually trigger the flow.
Run with roles	Roles that the flow runs with. This option is only available when **Run as** is set to **user who initiates the session**.

6. Select **Continue**.
7. Choose the event that your flow will be triggered by.

Option	Description
When the record is created	Create a flow that is triggered when a record in the selected table is created.
When the record is updated	Create a flow that is triggered when a record in the selected table is updated.
When the record is created or updated	Create a flow that is triggered when a record in the selected table is created or updated.

The flow that you just created is displayed in the flow diagramming view. See Flow diagramming view.

8. Select **Add node**, and then select one of the following options to modify your flow:

Option	Description
Action	Select the desired action. Workflow Studio includes Workflow Studio actions that are available to flows and subflows. Alternatively, a user with the action_designer role can create additional actions to add to flows. The Integration Hub and Spokes plugins install additional actions. To add draft actions from the More Actions menu, set **Show draft actions** to **true**. To view spokes that are available in the ServiceNow Store, set **Show store spokes** to **true** from the More Actions menu.

Option	Description
	Note: Under **Not Installed Spokes**, the system displays spokes that are available in the ServiceNow Store based on compatibility with the ServiceNow version and application dependency on Workflow Studio.
Flow Logic	Select an option to specify conditional or repeated operations.
Subflow	Select a published subflow and define the input values. In addition to adding a subflow as a flow action, you can enable the **Show triggered flows** option from the More Actions menu to select an activated flow and define the required inputs. Running a triggered flow ignores its trigger conditions and runs all actions.

Note:
Alternately, you may toggle the Workflow Studio default view from the view selector, and then select **Add an Action, Flow Logic, or Subflow** in that view of Workflow Studio. For more information, see Flow diagramming view.

9. Continue modifying your flow in Workflow Studio.

Note:
For detailed information on how to work with flows for your application, see Create a flow.

10. Select **Save**.
Workflow Studio saves a draft of the flow, trigger, and actions.

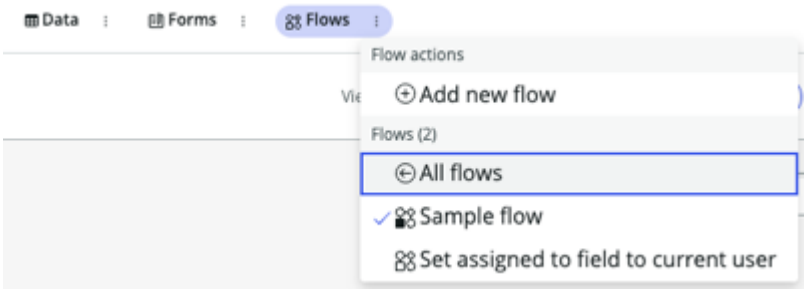
What to do next

Test your flow until you're ready to activate it. For more information on testing and editing flows, see Activate a flow.

Note:
Your application can trigger only activated flows. For detailed information on how to design, test, and activate flows for your application, see Flows in Workflow Studio.

To return to the comprehensive list of flows related to the selected table after editing a flow, select the Additional actions (⋮) menu to the right of the **Flows** tab, and then select **All flows** as shown in the following example.

All flows



Modify a record-based flow

Edit flows that are triggered when table records that you select are updated, created, or both updated and created by using the **Flows** tab in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Role required: personalize_forms, personalize_dictionary, and flow_designer permissions (or relevant AES user role and delegated developer permissions). For more information, see Delegate developers using AES.

About this task

The **Flows** tab in Table Builder provides a list of flows that are triggered by record changes to a table and enables you to manage them from the record table that triggers the workflow.

Procedure

1. Launch Table Builder as described in Accessing Table Builder.
2. Select the **Flows** tab.
   The available flows that are triggered by the table are listed as flow cards.
3. Filter or search the list of flow cards as needed to find the desired flow.
   See Find a record-based flow.
4. Select a flow card to open the flow for editing.
5. Edit the record-based flow by using Workflow Studio.
   For detailed information on how to work with flows for your application, see Create a flow.

   Note:
   To return to the list of flow cards that are related to the table, select the Additional actions (⋮) menu to the right of the **Flows** tab, and then select **All flows**.

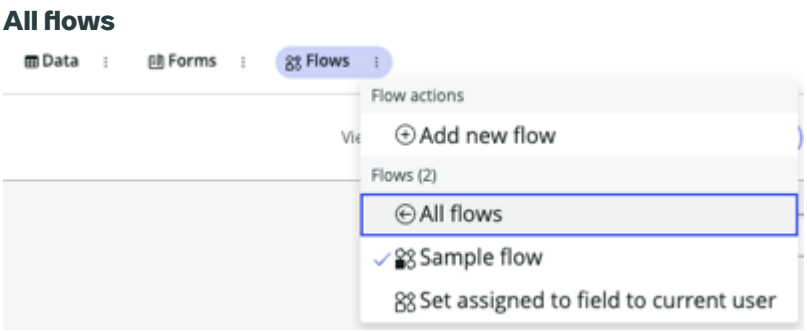
6. Select **Save**.

What to do next

Test your flow until you're ready to activate it. For more information on testing and editing flows, see Activate a flow.

Note:
Your application can trigger only activated flows. For detailed information on how to design, test, and activate flows for your application, see Flows in Workflow Studio.

To return to the list of flows related to the table that you selected after you edited a flow, select the Additional actions (⋮) menu to the right of the **Flows** tab, and then select **All flows** as shown in the following example.



Delete a record-based flow

Delete a record-based flow by using the **Flows** tab in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Role required: personalize_forms, personalize_dictionary, and flow_designer permissions (or relevant AES user role and delegated developer permissions). For more information, see Delegate developers using AES.

Procedure

1. Launch Table Builder as described in Accessing Table Builder.
2. Select the **Flows** tab.
   The available flows that are triggered by the table are listed as flow cards.
3. Filter or search the list of flow cards as needed to find the desired flow.
   See Find a record-based flow.
4. Select the Additional actions (⋮) menu in the top-right corner of the flow that you want to delete, and then select **Delete** from the list of options.
5. In the delete confirmation window, enter DELETE, and then select **Delete**.
6. Select **Save**.

Copy a record-based flow

Create a new record-based flow from an existing flow by using the **Flows** tab in Table Builder.

Before you begin

- Launch Table Builder. For more information, see Accessing Table Builder.
- (Optional) Choose a domain to work within (if not global). For more information, see Domain separation and Table Builder.
- (Optional) Choose an application scope to work within (if not global). For more information, see Using an application scope with Table Builder.

Role required: personalize_forms, personalize_dictionary, and flow_designer permissions (or relevant AES user role and delegated developer permissions). For more information, see Delegate developers using AES.

Procedure

1. Launch Table Builder as described in Accessing Table Builder.
2. Select the **Flows** tab.
   The available flows that are triggered by the table are listed as flow cards.
3. Filter or search the list of flow cards as needed to find the desired flow.
   See Find a record-based flow.
4. Select the Additional actions (⋮) menu in the top-right corner of the flow that you want to delete, and then select **Copy** from the list of options.
5. In the Create a copy of the flow window that appears, enter a name for the new flow that you want to create from the existing flow, and then select **Copy**.
6. Edit your new flow by using Workflow Studio.
   For detailed information on how to work with flows for your application, see Create a flow.
7. Select **Save**.

Note:

To return to the list of flows after editing a flow, select the Additional actions (⋮) menu to the right of the **Flows** tab, and then select **All flows**.

Table Builder reference

Reference topics provide additional information about the form and table elements you can configure using Table Builder.

Table properties in Table Builder

Various table and table column properties can be modified by using Table Builder.

Table column properties

The following table shows descriptions of the properties that you can modify for the columns in a data table.

Table column properties

Field	Description
Column label	Unique label for the column.
Column name	Database name for the column.
Type	Type of information that the column contains. For example, to contain plain text in the column, select **String**. Depending on the type that you select, fill in the additional fields to further define the table column. For example, if you select **String**, define the character limit of the string input. Or, if you select **Choice**, define the choices that users can choose from. See Field types.
Reference	Table that is associated with the column. This field applies only if the column type is **Reference**.
Max length	Maximum number of characters that users can enter in the field.

Table column properties (continued)

Field	Description
Default value	Value that populates the field automatically after a new record is created.
Display	Option to set the column as the display value for the table. A reference field shows the display value of the table to which it is referring. For example, the **Opened by** column of the task table refers to the user table. Because the display value of the user table is the user name, the **Opened by** field shows something like **Beth Anglin** or **Joe Employee**. When you select a display value, choose the table column that would act as an appropriate title for individual records. Only one column can act as the display value for a table.

Table properties

The following table shows the descriptions of the properties that you can modify for a selected data table.

Table properties

Option	Procedure
Change the table label	In the General information tab, update the **Table label** field.
Make the table extensible	1. In the General information tab, select **Advanced**. 2. Select the **Make extensible** check box.
Add record numbers	1. In the General information tab, select **Advanced**. 2. Select the **Add record number** check box. 3. Define the record numbers by updating the **Prefix**, **Starting number**, and **Number of digits** fields.
Accessible from	1. Open the **Access** tab. 2. In the **Accessible From** field, select **All Application Scopes** or **This Application Scope Only**.

Table properties (continued)

Option	Procedure
Application access controls	1. Open the **Access** tab. 2. In the **Application Access Controls** field, select Read, Create, Update, and/or Delete selection boxes to specify the desired level of access for the table.

Field configuration in Table Builder

You can configure the basic field properties for any field within Table Builder.

Field details

The Field details section lets you configure the basic field elements for the selected field in the form editor.

To see field type descriptions and learn more about what field types you can use, see Field types.

To understand the basic field properties and learn more about adding and customizing a field in a table, see Add and customize a field in a table.

The following table shows field descriptions for the Field details section.

Field details

Field	Description
Label	Name of the field that is displayed for form users.
Active	Option that you can select to make a field active so that form users can view it. Deselecting **Active** inactivates a field. When **Active** is not selected, the field will not display in the form editor. To reactivate a field, hover to the left of the field column in the **Data** tab and select the Open side panel icon [⚙]. Select **Active** and save changes.
Read only	Option that you can select so that the field is not editable by the form users.
Mandatory	Option that you can select so that the data that your users enter in the field is required before they can save their changes to the form.
Configure Label details menu (⋯)	Properties for a field label:

Field details (continued)

Field	Description
	• **Tooltip** property that is displayed when a user points to the field. • **Plural** property that indicates if multiple instances of a field exist, your user should enter the plural form of the label (for example, the plural of cat is cats) • **URL** property that indicates if the field label links to some other location, your user should enter the URL here.

Default value

The default value automatically fills in the field on the empty form for a new record, or fills in the field later (if the field is empty) when your user submits the new record. You can specify the default values as either a constant or use a script to generate them.

To learn how to define a default value for the fields, see Specify a default field value.

The following table shows field descriptions for the Default value section.

Default value

Field	Description
Use dynamic default	Property that displays the **Dynamic default value** field where you can set the default value that is displayed for the field.
Default value	String or number that represents the default value of the field.
Dynamic default value	Keywords that can be entered in the search field. Your user can then select a field that contains the desired default value for the field.

Function definition

A function field generates a value that is based on the simple computations of other fields and constants.

This section is displayed in the editor when you are creating a field and you select **Function field** in the Advanced properties section.

For information on defining function fields, see Function field.

The following table shows field descriptions for the Function definition section.

Function definition

Field	Description
Function Definition	Function expression that generates the desired value for the field.

Reference

A reference qualifier restricts the records that are available for reference fields.

For information on defining reference fields, see Reference field type.

The following table shows field descriptions for the Reference section.

Reference fields

Field	Description
Reference	Data table that the selected field references.
Use reference qualifier	Qualifier options: • **Simple** option that enables your users to build a simple conditional expression. Your user can select the **Edit Conditions** button below the **Reference qualifier** field. • **Advanced** option that enables your user to build an advanced conditional expression by using the **Reference qualifier** field. The advanced reference qualifiers enable your user to define an inline encrypted query string or JavaScript (actual code or the name of an existing script include or business rule) filter in the **Reference qualifier** field. • **Dynamic** option that enables your users to build a conditional expression for the selected reference field by using the **Reference qualifier** field. Dynamic reference qualifiers enable your users to use a dynamic filter to run a query against a reference field to filter the returned data set.
Reference qualifier	Referenced qualifier that is based on your user's selection for the **Use reference qualifier** field.
Edit conditions	Hyperlink to launch a pop-up window where your user can enter a simple conditional expression for the **Reference qualifier** field.

Dependent field

A choice or reference field can be declared dependent on another field on the same table. Dependent fields limit their available values based on the value in the dependent field.

Using dependent fields turns on choices. The available choices for your field rely on the value that a user chooses for the dependent field.

For information on defining field dependencies, see Make a field dependent.

The following table shows field descriptions for the Dependent fields section.

Dependent fields

Field	Description
Use dependent field	Field dependency that you can activate for the selected field.
Select dependent field	List where you select a different field on the form that can then be used to control which choices that are displayed to users in the form field you are configuring.

Choices

A choice list is a type of field that lets the user select from a pre-defined set of choices. You can define the available choices and customize the behavior and appearance of your choice lists. For information about the choice list configuration, see Choice list field type.

In the Choices section of the Config panel, select **Use choices**, and then select **Edit** or the **+Add** icon to customize the available choices for the selected field.

The following table shows field descriptions in the Choices section.

Choice properties

Field	Description
Label	Text that is displayed to your users for the choice in the list.
Value	Value for the choice field.
Dependent value	Value that you map in the selected Dependent field to the entered choice.
Domain	Domain that the choice resides in.
Active	Option to display the choice as a selection. Inactive choices do not display for users.
Options	Option that enables you to select **Use the choices from another field** to duplicate the displayed choices for this field from the choices in another selected field.
Show choices as	Options that control how a list of choices displays for users of your form.

Choice properties (continued)

Field	Description
	• **Dropdown** option. You must set a default value for the field. **Note:** If no default is set, then **None** automatically becomes the default value and is shown in the choice list to users. • **Dropdown with --None-- choice** option. **None** becomes the default value. • **Suggestions (Core UI only)** option. Choices are displayed as suggestions.
Filter	Option to filter the choice list. Select **Show inactive choices** to view all choices (including inactive ones).

Attributes

A dictionary attribute alters the behavior of the table or element that the dictionary record describes. You can add or modify the dictionary attributes. For information about dictionary attributes, see Dictionary attributes.

In the Attributes section of the Config panel, select **Edit** or **+Add** to customize the available attributes for the selected field.

The following table shows field descriptions for the Attributes section.

Attribute properties

Field	Description
Attribute	Option list of available attributes. Select an option from the list of available attributes. To review the list of available attributes, see Dictionary attributes.
Value	Attribute value that you can set to alter the field behavior that is described by the dictionary record.

Formula

A formula allows you to calculate the value of a column without writing a script. You can use one of the predefined formulas or combine two are more formulas to calculate the column value. For information on the predefined formulas and examples, see Formulas.

In the Formula section, select **Edit** or **+ Add** to edit or add a formula.

Formula properties

Field	Description
Formula editor	A text-editor like interface that allows you to enter and edit formulas.

Formulas for column values in Table Builder

You can use a predefined function and create a formula to calculate the value of a column without writing a script. Use a predefined function or create a nested formula by using the existing predefined functions to calculate the column value type.

Supported operators

The following comparison operators are supported only for number type values.

- = (Equal to)
- <> (Not equal to)
- > (Greater than)
- < (Less than)
- >= (Greater than or equal to)
- <= (Less than or equal to)

Simple math functions

Use simple math functions to perform basic mathematical calculations on numeric value columns.

AVERAGE

Returns the average value of the arguments.

Syntax	Input	Output
AVERAGE(argument 1, argument 2, ... argument n)	Numeric value, function call, or variable	Numeric value

Examples:

- Function: AVERAGE(1,2,3)

  The result is 2.

- Formula: AVERAGE(LENGTH(first_name), LENGTH(last_name))

  The result is the average value of the number of characters in the first_name column and last_name column.

DIVIDE

Returns the final quotient value after consecutively dividing the first argument with the next argument until the function reaches the last argument.

Syntax	Input	Output
DIVIDE(argument 1, argument 2 ... argument n)	Numeric value, function call, or variable	Numeric value

Examples:

- Function: DIVIDE(10,20, 0.25, 10)

  The result is 0.2.

- Formula: DIVIDE(LENGTH(full_name),2)

  The result is the number of characters in the full_name column divided by 2.

INDEXMATCH

Retrieves the first not null value from the specified set of arguments.

Syntax	Input	Output
INDEXMATCH(argument 1, argument 2 , ... argument n)	String, numeric value, function call, or variable.	Numeric value

Example:

Function: INDEXMATCH("","" ",2,"string")

The result is 2.

MAX

Returns the highest value in the specified arguments.

Syntax	Input	Output
MAX(argument 1, argument 2, ... argument n)	Numeric value, function call, or variable	Numeric value

Examples:

- Function: MAX(1, -5, 20, 6)

  The result is 20.

- Formula: MAX(LENGTH(first_name), LENGTH(last_name))

  The result is the number of characters in the first_name column or last_name column whichever is the highest.

MIN

Returns the lowest value in the specified arguments.

Syntax	Input	Output
MIN(argument 1, argument 2, ... argument n)	Numeric value, function call, or variable	Numeric value

Examples:

- Function: MIN(1, -5, 20, 6)

  The result is -5.

- Formula: MIN(LENGTH(first_name), LENGTH(last_name))

  The result is the number of characters in the first_name column or last_name column whichever is the lowest.

MULTIPLY

Returns the total multiplied value of the arguments.

Syntax	Input	Output
MULTIPLY (argument 1, argument 2, ... argument n)	Numeric value, function call, or variable	Numeric value

Examples:

- Function: MULTIPLY(12, 4)

  The result is 48.

- Formula: MULTIPLY(order, 2)

  The result is the order column value multiplied by 2.

POWER

Returns the result of the base value raised to the power of the exponent value.

Syntax	Input	Output
POWER(argument 1, argument 2)	argument 1 is base and argument 2 is exponent. • base: number or variable • exponent: number or variable	Number

Examples:

- Function: POWER(3,2)

  The result is 9.

- Formula: POWER(LENGTH(full_name),2)

  The result is the number of characters in the full_name column to the power of 2.

SUBTRACT

Returns the result value after consecutively subtracting the next available argument from the earlier argument until the function reaches last argument.

Syntax	Input	Output
SUBTRACT(argument 1, argument 2 ... argument n)	Numeric value, function call, or variable	Numeric value

Examples:

- Function: SUBTRACT(1.15, 0.02, 0.45, -0.85)

  The result is 1.53.

- Formula: SUBTRACT(LENGTH(full_name), LENGTH(first_name))

  The result is the number of characters from the full_name column minus the number of characters from the first_name column.

SUM

Returns the sum of all the arguments.

Syntax	Input	Output
SUM(argument 1,argument 2, ... argument n)	Numeric value, function call, or variable	Numeric value

Examples:

- Function: SUM(0.03, -0.02, 1)

  The result is 1.01.

- Formula: SUM(LENGTH(first_name), LENGTH(last_name))

  The result is the total number of characters in the first_name column plus the total number of characters in the last_name column.

COUNTIF

Returns the number of arguments that match the specified criteria within the specified set of arguments.

Syntax	Input	Output
COUNTIF(argument 1, argument 2, argument n-1, criteria)	• argument 1 ... argument n: String, numeric value, function call, or variable. • criteria: Criteria that evaluates the specified	Numeric value

Syntax	Input	Output
	set of arguments. String, numeric value, function call, or variable.	

Example:

Function: COUNTIF(2,3,2,"string",2)

The result is 2.

MODE

Returns the most frequently repeating value in the specified set of arguments.

Syntax	Input	Output
MODE(argument 1,argument 2, ... argument n)	Numeric value, function call, or variable	Numeric value

Example:

Function: MODE(1, 2, 2, 3, 3, 3)

The result is 3.

String functions

Use string functions to reformat or perform calculations on string column values.

CONCATENATE

Joins one or more input strings into a single string.

Syntax	Input	Output
CONCATENATE(string 1, string 2, ... string n)	String, function call, or variable	String

Examples:

- Function: CONCATENATE(first_name, ".", last_name, "@", LOWERCASE(example), ".com")

  The result is the concatenated value <first_name_value>.<last_name_value>@example.com. In this example, <first_name_value> and <last_name_value> are placeholders.

- Function: CONCATENATE(first_name, " ", last_name)

  The result is the concatenated string of first_name column value and last_name column value separated by a white space.

ISBLANK

Finds white spaces or blank values in the string and returns true if there are any.

Syntax	Input	Output
ISBLANK(argument)	String or value	True or false

Examples:

- Function: ISBLANK(“example_string”)

  The result is false.

- Function: ISBLANK(full_name)

  The result is true only when there are empty spaces in the full_name column. Otherwise, the result is false.

LENGTH

Returns the total number of characters in the input string.

Syntax	Input	Output
LENGTH(argument)	String value, function call, or variable	Numeric value

Examples:

- Function: LENGTH("example_string")

  The result is 14.

- Function: LENGTH(full_name)

  The result is the total number of characters in the full_name column value.

LOWERCASE

Converts the input string to all lowercase characters.

Syntax	Input	Output
LOWERCASE(argument)	String, function call, or variable	String in lowercase

Examples:

- Function: LOWERCASE(“ExamPle inpuT stRing”)

  The result is example input string.

- Function: LOWERCASE(sys_created_by)

  The result is the lower case string of the sys_created_by column value.

REPLACE

Replaces the characters in the source string with the characters in the target string.

Syntax	Input	Output
REPLACE(source_string, target_string, replacement_string)	• source_string: String, function call, or variable • target_string: String, function call, or variable • replacement_string: String, function call, or variable	String

Examples:

- Function: REPLACE(“Pepperoni Pizza”, “Pepperoni”, “Cheese”)

  The result string is Cheese Pizza.

- Function: REPLACE("abe.tuter@example.com", "example", company_name)

  The result string is abe.tuter@<company_name>.com. In this example, <company_name>is a place holder.

TITLECASE

Converts the input string to all title case characters.

Syntax	Input	Output
TITLECASE(argument)	String, function call, or variable	String in title case

Examples:

- Function: TITLECASE("example string")

  The result is Example String.

- Function: TITLECASE(full_name)

  The result is the full name column value in the title case.

UPPERCASE

Converts the input string to all uppercase characters.

Syntax	Input	Output
UPPERCASE(argument)	String value, function call, or variable	String in uppercase

Examples:

- Function: UPPERCASE("eXamPle sTring")

  The result is EXAMPLE STRING.

- Function: UPPERCASE(state)

  The result is the State column value in upper case.

FIND

Searches for the first occurrence of a substring within a string and returns the position of the first occurrence.

Note:
This function is case sensitive.

Syntax	Input	Output
FIND(search_string, source_string, from_index)	• search_string: Substring, function call, or variable. • source_string: Main string, function call, or variable. • from_index: Index position in the main string from where the search should start. Numeric value, function call, or variable.	Numeric value (integer). When the substring does not exist in the main string, -1 is returned.

Example:

Function: FIND("morning", "Good morning")

The result is 5.

SEARCH

Searches for a substring within a string and returns the position of the first occurrence of the substring.

Note:
This function is case insensitive.

Syntax	Input	Output
SEARCH(search_string, source_string, from_index)	• search_string: Substring, function call, or variable. • source_string: Main string, function call, or variable. • from_index: Index position in the main string from where the search should start. Numeric value, function call, or variable.	Numeric value (integer). When the substring does not exist in the main string, -1 is returned.

Examples:

- SEARCH("Morning", "Good morning")

  The result is 5.

- SEARCH("World","Hello world!")

  The result is -1.

SUBSTRING

Retrieves a substring from a string at the specified index position and for the specified length.

Syntax	Input	Output
SUBSTRING(source_string, start_index, length)	• source_string: String, function call, or variable. • start_index: Position in the string from where the substring is extracted. Numeric value, function call, or variable. • length: Length of the substring that should be extracted.	String

Example:

SUBSTRING("Hello, Good Morning", 7, 4)

The result substring is 'Good'.

Date and time functions

Use date and time functions to calculate or reformat the date and time column values.

NOW

Returns the current date and time of the instance in ISO format (YYYY-MM-DD hh:mm:ss).

Syntax	Input	Output
NOW()	No arguments are required for this function	ISO format of current date and time

Example:

Function: NOW()

The result is the current date and time in ISO format.

TODAY

Returns the current date with time offset to start of the day in ISO format in UTC time zone.

Syntax	Input	Output
TODAY()	No arguments are required for this function	Current date with time offset to start of the day in ISO format.

Example:

Function: TODAY()

The result is the current date and start time of the day in ISO format.

TIMEDIFF

Evaluates the time duration difference between two dates.

Syntax	Input	Output
TIMEDIFF(argument1, argument2)	Date in ISO format (YYYY-MM-DD hh:mm:ss) as string or variable	Duration

Examples:

- Function: TIMEDIFF("2021-05-02 9:10:12", "2021-04-07 6:2:23")

  The result is 25 03:07:49.

- Formula: TIMEDIFF(sys_created_on, NOW())

  The result is the time duration difference between the sys_created_on date and the current date of the system.

DATEDIF

Evaluates the difference between the two dates in days, months, or years.

Syntax	Input	Output
DATEDIF(start_date, end_date, date_difference_unit)	• start_date: Date in ISO format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string or variable. • end_date: Date in ISO format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string or variable. • date_difference_unit: Character String and either "Y","M", or "D" in lowercase or uppercase. Default is "D".	Numeric duration value based on the specified date difference unit.

Example:

Function: DATEDIF("2021-05-02 9:10:12", "2021-05-05 6:2:23 ","d")

The result is 3.

DATE

Creates a date from the specified individual year, month, and day values. The created date is in Coordinated Universal Time (UTC) time zone.

Syntax	Input	Output
DATE(year,month,day)	• year: Numeric value, variable or function. • month: Numeric value, variable or function. • day: Numeric value, variable or function.	Date in ISO format (YYYY-MM-DD hh:mm:ss)

Example:

Function: DATE(2021,5,2)

The result is 2021-05-02 00:00:00.

DAY

Retrieves the numerical day component from the specified date.

Syntax	Input	Output
DAY(date)	Date in ISO format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string, variable, or function.	Numeric value (integer). The values range from 1 through 31.

Examples:

- Function: DAY("2021-05-029:10:12")

  The result is 2.

- Function: DAY(NOW())

  The result will be the day component of the current date and time.

MONTH

Retrieves the numerical month component from the specified date.

Syntax	Input	Output
MONTH(date)	date: Date in ISO format (YYYY-MM-DD or YYYY-MM-	Numeric value (integer). The values range from 1(January) through 12(December).

Syntax	Input	Output
	DD hh:mm:ss) as string or variable.	

Examples:

- Function: MONTH("2021-05-02 9:10:12")

  The result is 5.

- Function: DAY(NOW())

  The result will be the month component of the current date and time.

YEAR

Retrieves the year component from the specified date.

Syntax	Input	Output
YEAR(date)	Date in ISO format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string, variable, or function.	Numeric value (integer)

Examples:

- Function: YEAR("2021-05-02 9:10:12")

  The result is 2021.

- Function: YEAR(NOW())

  The result will be the year component of the current date and time.

WEEKDAY

Returns the numerical day of the week for the specified date. The day range is 1 (Sunday) through 7 (Saturday).

Syntax	Input	Output
WEEKDAY(date)	date: Date in ISO format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string or variable.	Numeric value (integer)

Example:

Function: WEEKDAY("2021-05-02 9:10:12")

The result is 1.

TEXT

Retrieves the specific date components in a date in string format.

Syntax	Input	Output
TEXT(date, format_text)	• date: Date in ISO format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string, variable. • format_text: Date components as string or variable that are to be extracted.	String

Example:

TEXT("2022-08-17 9:10:12","yyyy-MM")

The result is 2022-08.

DATEVALUE

Converts a date in text format into a date in ISO format.

Syntax	Input	Output
DATEVALUE(date_text)	date_text: Date stored as text must be in YYYY-MM-DD format.	Date in ISO format (YYYY-MM-DD hh:mm:ss) as string.

Example:

Function: DATEVALUE("2021-05-02")

The result is 2021-05-02 00:00:00.

WORKDAY

Returns the nearest working day for the specified input date by excluding the specified holidays and weekends before or after the specified n number of days.

Syntax	Input	Output
WORKDAY(start_date, days, holiday 1,holiday 2, ..., holiday n)	• start_date: Date in ISO format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string or variable. • days: Number of days as a numeric value, string, or function. • holiday 1...holiday n (Optional): Date in ISO	Date in ISO format (YYYY-MM-DD hh:mm:ss) as string.

Syntax	Input	Output
	format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string or variable.	

Example:

Function: WORKDAY("2022-08-17 9:10:12",2)

The result is 2022-08-19 00:00:00.

NETWORKDAYS

Calculates the number of working days between two dates by excluding weekends and specified holiday dates. Number of working days includes the start date and end date.

Syntax	Input	Output
NETWORKDAYS(start_date,end_date,holiday 1,holiday 2, ... holiday n)	• start_date: Date in ISO format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string or variable. • end_date: Date in ISO format (YYYY-MM-DD or YYYY-MM-DD hh:mm:ss) as string or variable. • holiday 1,holiday 2, ... holiday n (optional) : List of holidays that should be excluded while calculating working days.	Numeric value (integer)

Example:

Function: NETWORKDAYS("2022-08-17 20:10:12","2022-08-19 9: 10:12")

The result is 3.

Logical functions

Use logical functions to perform logical operations on column values.

AND

Performs a logical AND operation on the arguments.

Syntax	Input	Output
AND(argument 1, argument 2)	String, function call, or variable	True or false

Examples:

- Function: AND(2>3, 4<5)

  The result is false.

- Formula: AND(LENGTH(sys_created_by)>25, LENGTH(sys_updated_by)>25)

  The result is true only when the number of characters in both the sys_created_by and sys_updated_by columns are greater than 25. Otherwise, the result is false.

IF

Executes the specified statements based on the Boolean output of the conditional expression.

Syntax	Input	Output
IF(<conditional_expression>, <do_this_when_true>, <do this_when_false>)	• conditional_expression: Logical conditional expression, function call, or variable **Note:** Logical comparison of strings is not supported in the conditional expression. • do_this_when_true: String, numeric value, function call, or variable that is returned when the condition evaluates to true • do_this_when_false: String, numeric value, function call, or variable that is returned when the condition evaluates to false	String, numeric value, function call, or variable based on the Boolean output of the conditional expression.

Examples:

- Function: IF(number_of_incidents >= 5, "High", "Medium")

  If the number of incidents is greater than 5, the string ‘High’ is returned. In other cases, the string ‘Medium’ is returned.

- Function: IF(LENGTH(full_name) > 100, "Number of characters exceed the limit", "Number of characters within the limit")

  If the number of characters for the full_name column is above 100, the string 'Number of characters exceed the limit' is returned. Otherwise, the string 'Number of characters within the limit' is returned.

OR

Performs logical OR operation on the arguments.

Syntax	Input	Output
OR(argument 1, argument 2)	Conditional expression, function call, or variable	True or false

Examples:

- Function: OR(2>3,4<5)

  The result is true.

- Formula: OR(LENGTH(first_name)>25, LENGTH(last_name)<25)

  The result is true when the number of characters in the first_name column is greater than 25 or the number characters in the last_name column is less than 25. Otherwise, the result is false.

IFERROR

Evaluates expression 1 and returns the expression 1 value when there are no errors in the expression 1. When an error occurs while evaluating expression 1, expression 2 is evaluated and expression 2 value is returned.

Syntax	Input	Output
IFERROR(expression 1, expression 2)	• expression 1: Arithmetic, Logical expression, function call, String, numeric value or variable. • expression 2: Arithmetic, Logical expression, function call, String, numeric value or variable.	Result of expression 1 when there are no errors in expression 1. Otherwise, result of expression 2.

Example:

Function: IFERROR(MULTIPLY(snr_factor, signal), MULTIPLY(default_factor, signal))

If the snr_factor value is a valid number, the multiplied value of snr_factor with signal is returned. If the snr_factor value is not a valid number, the multiplied value of default_factor value with signal is returned.

Add a formula to a column in Table Builder

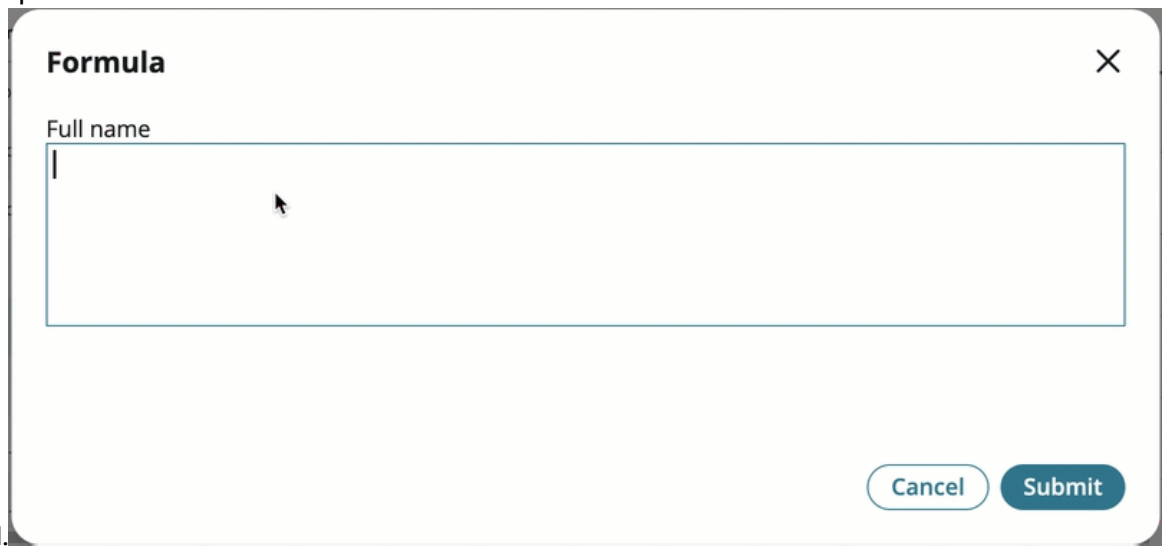
Use a predefined function or create a formula to calculate a value for a column without writing a script in Table Builder. You can use two or more functions to create a formula according to your requirements.

Before you begin

- The scope of the table should be within the scope of the application. Otherwise, you cannot add a formula to a column in the table that is outside the scope of the app.
- Role required: personalize_dictionary or AES user role and delegated developer permissions. For more information, see Delegate developers using AES.

Procedure

1. From the **Data** or **Forms** tabs, click the Open side panel icon ( ) for a column.
2. In the **Config** tab of **Formula** section, click **+ Add**.
3. Enter a predefined function or a formula in the **Formula editor** modal.



For example, to concatenate the First name column value with the Last name column value and populate the Full name column value, enter CONCATENATE(`first_name,last_name`). For more information about available predefined functions and example formulas, see Formulas for column values in Table Builder

For syntax guidance and examples, click the function in the auto-suggestion that appears as you begin typing.

Note:
You cannot add a formula for unsupported columns, columns that are function fields, and the columns that already have script as the calculated value type.

4. Click **Submit**.

Result
The formula is validated and added to the column if it doesn't have any errors.

Policies and rules properties in Table Builder

You can configure the basic field policies and rules for any field that you work with in Table Builder.

UI Policies

The UI Policies section enables you to configure how forms appear based on roles and user input.

To understand the basic field properties, see Create a UI policy in Table Builder.

The following table shows the field descriptions for the Policy details section when adding or editing a UI policy.

Policy details

Field	Description
Short description	Short summary of the UI policy.
Table	Data table that the policy applies to.
Domain	Domain that the choice resides in.
Order	Sequence for processing, from the lowest number to the highest number. If two policies conflict, the UI policy with the higher number runs. For inherited UI policies, the extended (child) table's UI policies run first. Then, the base table UI policies run, both from the lowest to the highest specified value.
Active	Status of the UI policy. Only active UI policies are applied.
Apply to all views/Apply to view [Advanced settings]	Option for specifying whether the UI policy applies to all form views or specific views. For more information on form views, see View management.
On load [Advanced settings]	Option for specifying that the UI policy behavior should be performed OnLoad as well as when the form changes. Check or clear the UI policy **On load** check box to control whether it runs every time a form is loaded and conditions are satisfied. In the following example, an administrator does not want an incident to enter the **Awaiting user info** state unless the user provides an explanation to the customer. The administrator creates a UI policy with the following settings: • In the When these conditions are met section, adds the condition **[State] [is] [Awaiting user info]** and clears the **On load** check box. The UI policy applies only when the state is changed to **Awaiting user info**. • In the UI Policy Actions list, creates a record that makes the **Additional comments** field required when the condition is met.
Reverse if false [Advanced settings]	Option for specifying that the UI policy action should be reversed when the conditions of its UI policy evaluate to false. When the conditions are true, actions are taken and when they change back to false, the actions are reversed (undone).

Policy details (continued)

Field	Description
Inherit [Advanced settings]	Option for specifying whether extended tables inherit this UI policy. When a child table has an inherited UI policy from its parent table, the UI policy on the child table always runs first, regardless of the **Order** of the UI policies. In the following example: • A child table has a UI policy with the**Order** value **500** that shows the **Urgency** field when its conditions are met. • Its parent table has a UI policy with the same conditions that hides the **Urgency** field. The parent table UI policy has the **Order** value **100**. • Although the parent table **Order** field has a lower value, the child UI policy runs first and then the parent UI policy runs. When the conditions are met, the **Urgency** field is hidden.

The following table shows field descriptions for the When these conditions are met section when adding or editing a UI policy.

When these conditions are met

Field	Description
Condition set	Conditions that, if fulfilled, cause the UI policy to be applied. Conditions are built with the condition builder by using logic statements. To set conditions using a script, use a client script instead. Conditions are only rechecked if a user manually changes a field on a form. If the change is made by a UI action, context menu action, or through the list editor, it is not evaluated.

The following table shows field descriptions for the Do the following section when adding or editing a UI policy.

Do the following

Field	Description
Field name	Field on the selected table to which the UI policy performs an action if conditions are met.

Do the following (continued)

Field	Description
	Note: If the specified field is not on the form, the UI policy performs the action on the variable with the same name.
Mandatory	List for specifying how the UI policy affects the required state of the field. The choices are the following: • **Leave alone** • **True** • **False**
Visible	List for specifying how the UI policy affects the visible state of the field, that is, whether it appears. The choices are the following: • **Leave alone** • **True** • **False**
Read only	List for specifying how the UI policy affects the read-only state of the field. The choices are the following: • **Leave alone** • **True** • **False**
Clear the field value	Option to clear the specified field if the conditions are met.

Access control rules

The Access control rules section enables you to configure access to the specified resource based on role, condition, and script criteria being met.

The following table shows field descriptions for the Access control rules section.

Field	Description
Name	Names each user who has permission to access the form or data. • Enter the names as firstname.lastname in lower case letters, separated by a period (for example, `john.smith`). Each name must have a corresponding user record in *`hi.servicenow.com`*.

Field	Description
	• Enter multiple names and separate them by commas if multiple users have permission to access the form or data. • Enter an asterisk (*) as the name to enable all users to access the form or data.
Description	Overview of what the access control rule restricts or enables.
Type	Type of object the access controls, including the following: • Client-callable script includes • Processor • Record • UI page • UX route For more information on object types, see ACL matching requirements for objects.
Operation	Type of action the system can take on the specified object, such as delete or execute. Some objects, such as records, support multiple operations, while other objects, such as a REST_Endpoint, only support one operation. For more information on action types, see ACL rule types.
Active	Active state of the rule: **true** The rule is active and available for use. **false** The rule is inactive and unavailable for use.
Condition	Any conditions that apply to the rule.
Roles	Any roles the rule applies to.
Admin overrides	Option to force the rule's evaluation for admin overrides at the access level. For more information on admin overrides, see Evaluate the admin override at the access level.
Advanced	Option to associate a script with the rule.

Client Scripts

The Client Scripts rules section enables you to create scripts that control how form fields appear based on defined criteria.

For a complete list of values for client script fields, see Client script form

The following table shows field descriptions for the Client Scripts rules section.

Field	Description
Name	Name for the script.
Description	Overview the script's functionality and purpose.
Table	Table or form that the script is available for.
Active	Active state of the script: **true** The script is active and available for use. **false** The rule is inactive and unavailable for use.
UI type	Type of application the script is available for, such as desktop or mobile.
Type	When the script runs on the table or form, including the following: • onCellEdit • onChange • onLoad • onSubmit
Field name	Field the script is run on.
View	Views on which the client script runs (not applicable to global scripts).
Inherited	Indicates whether the client script applies to extended tables.

Business Rules

The Business Rules section enables you to create business rules to accomplish tasks like automatically changing values in form fields when certain conditions are met.

For a complete list of values for business rules, see Create a business rule.

The following table shows field descriptions for the Business Rules rules section.

Field	Description
Name	Name for the business rule.
Table	Table or form that the business rule runs on.
Active	Active state of the business rule:

Field	Description
	true The business rule is active and available for use. **false** The business rule is inactive and unavailable for use.
When	When the business rule executes: **display**, **before**, **async**, or **after** the database operation is complete.
Filter conditions	Conditions created using the condition builder to determine when the business rule should run based on the field values in the selected table.
Role conditions	Roles that users who are modifying records in the table or form must have for the business rule to run.
Actions	Actions taken by the business rule on the specified fields, such as **insert**, **update**, and **delete**.
Set field values	Table or form fields that the action executes on.
Add message	Whether a message appears when this business rule is run: **true** A message appears when the business rule is run. **false** No message appears when the business rule is run.
Abort action	Whether the business rule aborts the current database transaction: **true** The business rule aborts the current database transaction when run. **false** The business rule doesn't abort the current database transaction when run. For example, on a before insert business rule, if the conditions are met, do not insert the record into the database.
Order	Indicates the sequence in which this business rule should run. If there are multiple rules on a particular activity, the rules run in the specified order specified, from lowest to highest.

Workspace view rules

The Workspace view section enables you to define rules to control how users view workspaces based on criteria.

The following table shows field descriptions for the Workspace view rules section.

Field	Description
Name	Name for the workspace view rule.
Table	Table or form that the workspace view rule is available for.
View	View that is used to render the form. The default view is used if this field is left empty or contains an invalid value.
Roles	Any roles the rule applies to.
Conditions	Any conditions that apply to the rule.
Hide section navigation	Option to enable or disable section navigation.
Disable section collapsing	Option to enable or disable section collapsing.
Default tab order	Order in which form tabs appear by default.

UI Builder

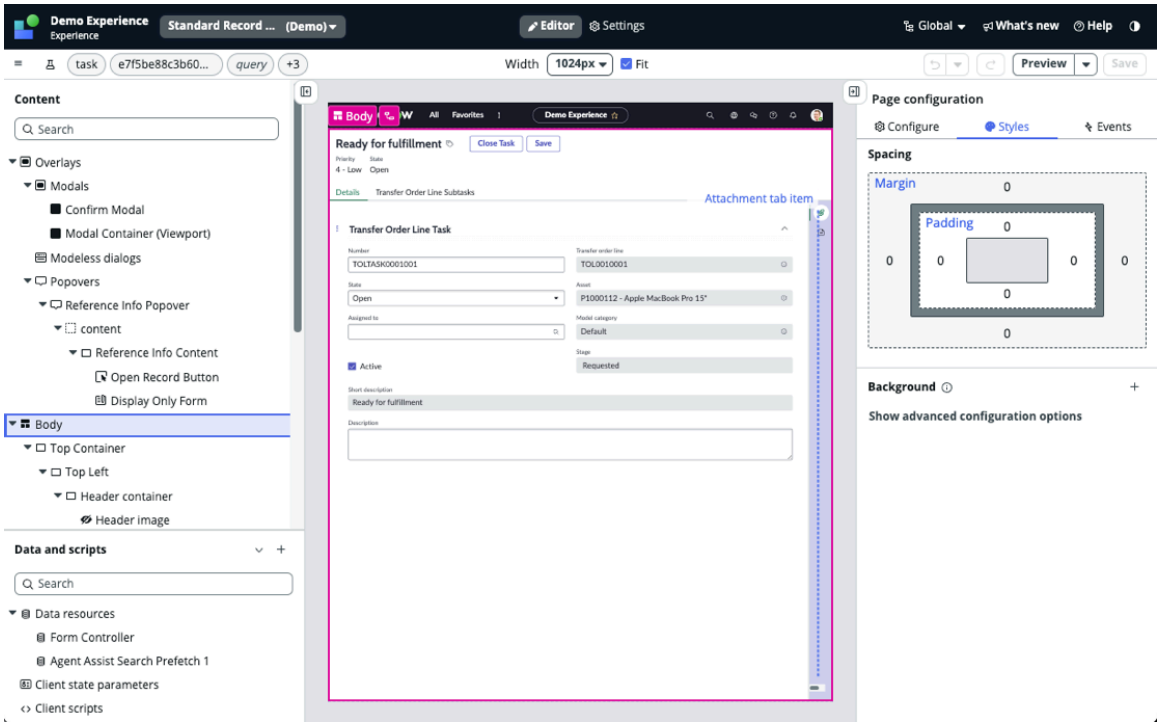
UI Builder is a web user interface builder. Use UI Builder to build pages for App Engine Studio generated workspaces or custom web experiences using Next Experience Components and custom web components.

UI Builder overview

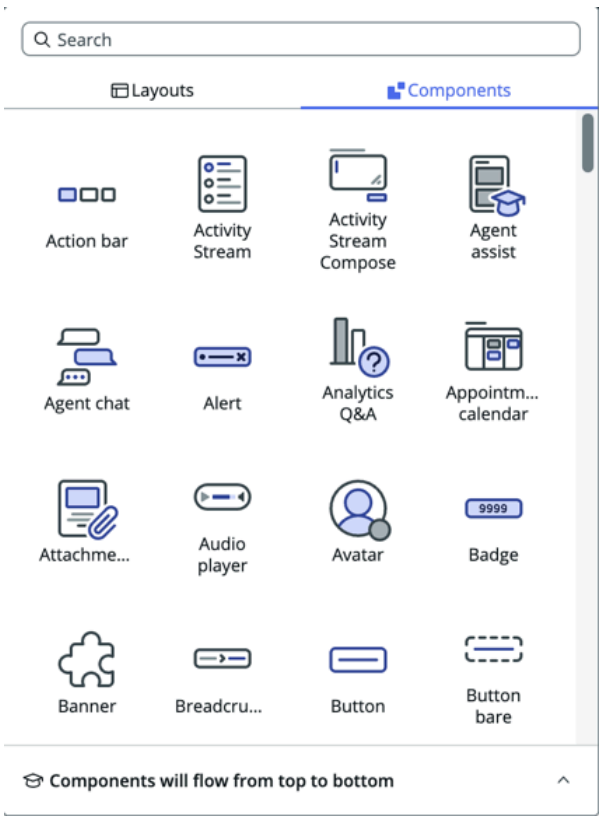
Create or customize pages for workspace experiences. A page is a collection of components that make up a workspace user interface.

Create variants of pages to target experiences for different audiences. For example, you can create a home page for agents, and a variant for managers at the same URL.

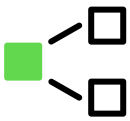
Note:
The ui_builder_admin role is required to complete tasks in UI Builder.



Use the UI Builder library of components to build your pages. Configure them any way that you need and then connect your organization's data to the components.



Get started with UI Builder

Explore	Learn
 Explore UI Builder concepts and features.	 Learn the basics of using UI Builder.
Work	**Advanced**
 Dive deeper into configuring UI Builder experiences and pages.	 Learn how to add data and make your pages dynamic.

Note:
UI Builder isn't yet capable of building or configuring ServiceNow base system service portals, such as the Employee Center. For service portals, continue to use the Service Portal Designer.

Troubleshoot and get help with UI Builder

- Contact your company's Customer Admin to unlock or add user accounts, perform restores or zBoots, and more.

  Contact your company's Customer Admin

- Ask or answer questions in the UI Builder community
- Search the Known Error Portal for known error articles
- Learn more about how to create your own apps on the developer site.
- Contact Customer Service and Support

Exploring UI Builder

Explore facets of UI Builder to set you up for creating pages quickly.

Learn more about UI Builder	Additional ServiceNow resources
UI Builder is a web user interface builder. Use UI Builder to build pages for CSM Configurable Workspace, App Engine Studio generated workspaces and portals, or custom web experiences using Next Experience Components and custom web components.	 ServiceNow Dev Program YouTube channel

Learn more about UI Builder	Additional ServiceNow resources
	 Community YouTube channel
	 ServiceNow Community site
	 Developer Advocate blog
	 ServiceNow Developer site
	 Components documentation

Check out the UI Builder quick start topic to understand the basics of UI Builder. If you want a deeper understanding of UI Builder, go through the tutorial to learn how to create your first page.

Find information relating to UI Builder audiences, security and roles settings like application scope and domain separation, as well as experience settings.

See how you can use UI Builder with CSM Workspace and portal experiences.

UI Builder quick start

This quick start guides you through the process of creating your first page in UI Builder. Creating your first page is the first step in understanding how to build user interface pages for your workspace or custom portal experiences.

Before you begin

Role required: ui_builder_admin

In this UI Builder quick start, you perform the following tasks to build your first page in UI Builder:

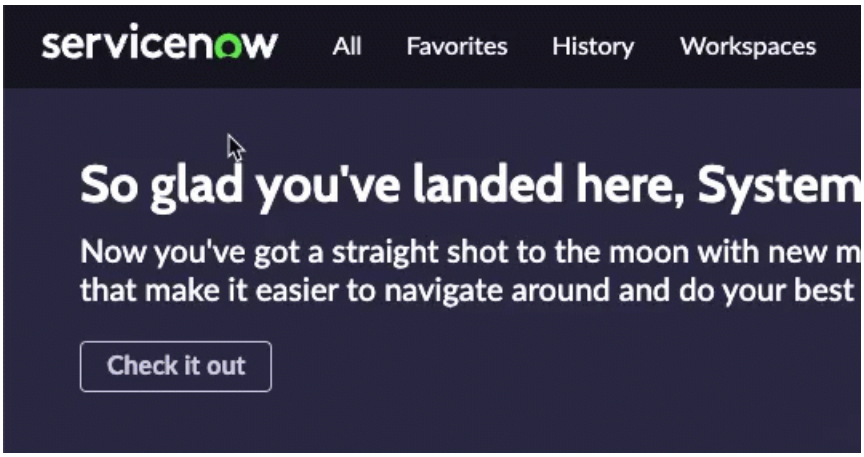
- Start UI Builder.
- Create a page for your workspace or custom portal experience. For more information about creating pages, see Create a page in UI Builder.
- Build your page by adding components. For more information about components, see Customize UI Builder pages using components.

- Save your page.
- Preview your page to see how it looks in a browser.

Procedure

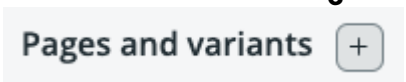
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.

Start UI Builder

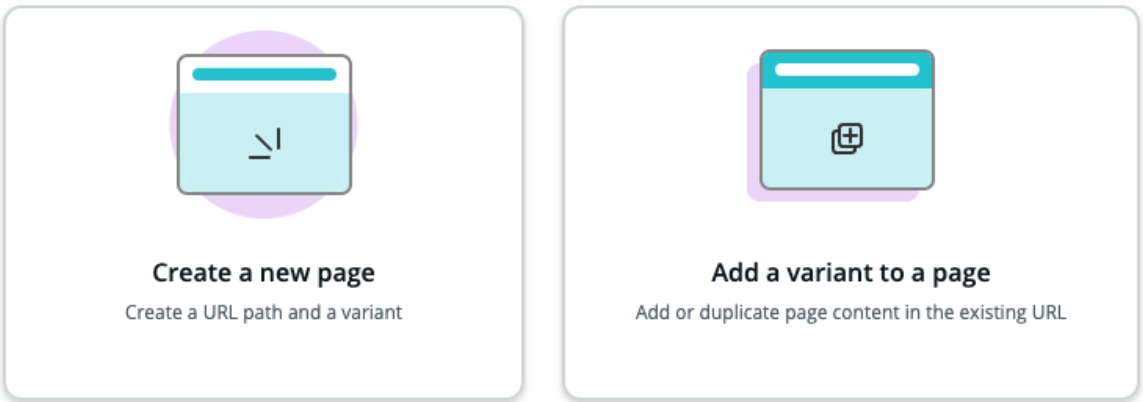


2. Select an experience that you want to work in from the UI Builder home page.
   If you don't see any experiences listed in which to work, contact your administrator to get access to an experience, or create an experience. For more information, see Configure how users interact with your applications in UI Builder.
3. Create a page.

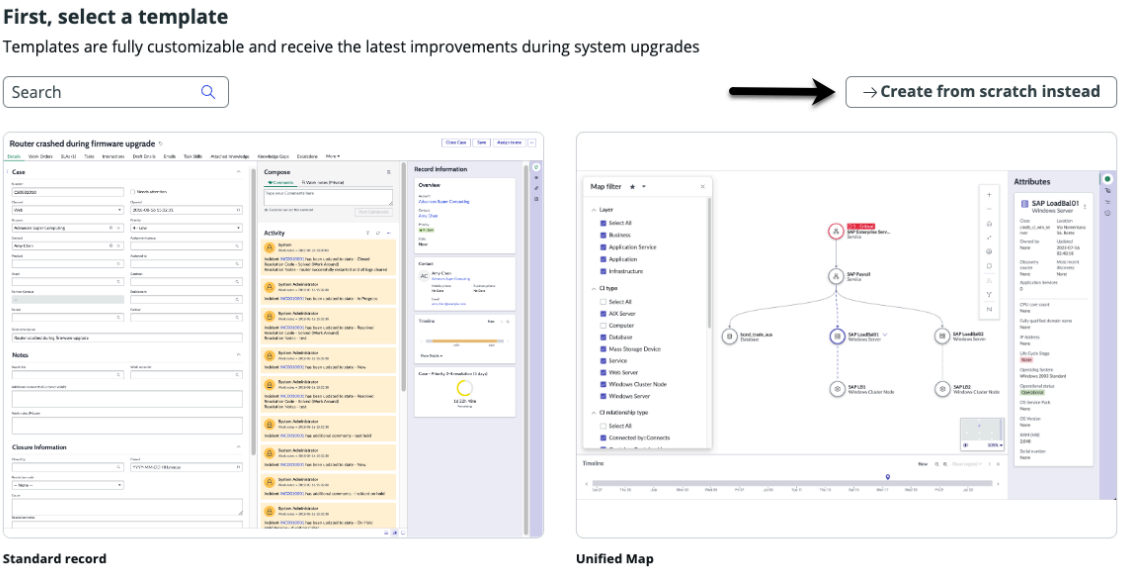
   a. Select the **+** icon in the **Pages and variants** section.



   b. Select **Create a new page**.



   c. Select → **Create from scratch instead**.

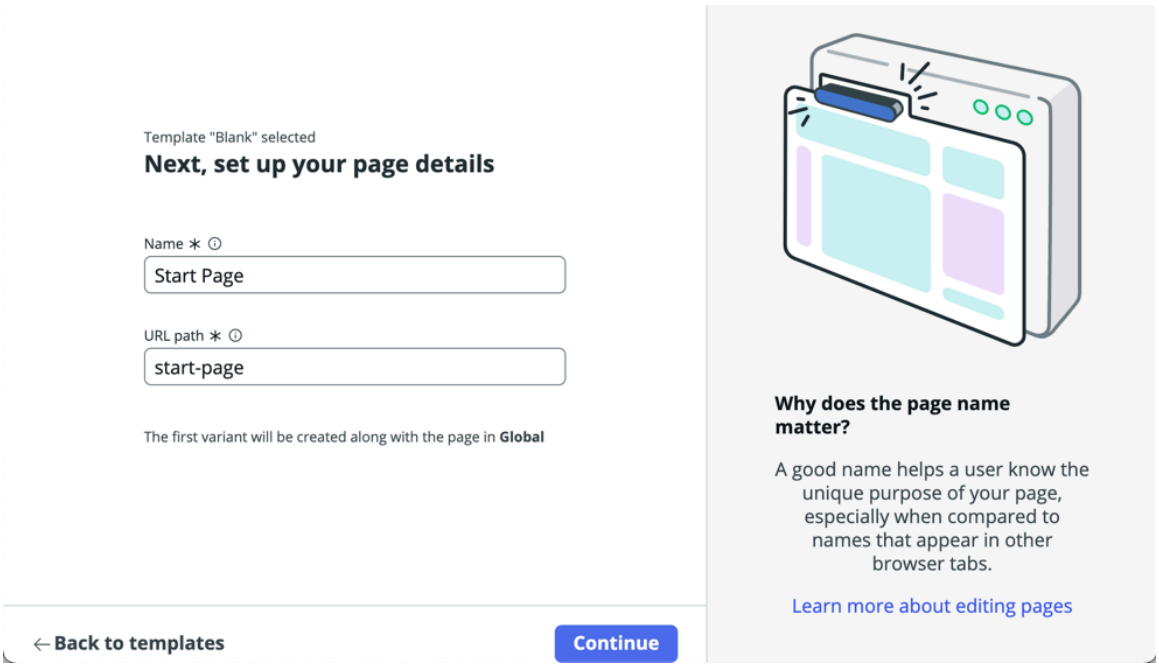


You can also create pages using page templates, see Create a page from a template for more information.

d. Enter `Start Page` as the unique name for the page in the **Name** field.

e. Specify a path for your page in the **URL Path** field. UI Builder generates a default path based on the name that you gave in the last step.

A default path is added based on your page name. You can also create your own path. The path is required and must be unique. The path can include digits (0-9), letters (A-Z, a-z), and a few special characters (`" - "`, `" . "`, `" _ "`, `" ~ "`), with the words separated by a forward slash or hyphen. The **URL preview** shows the path of your page.



Note:
The application scope defaults to the scope that the user is currently in within the ServiceNow AI Platform®. For more information about the application scope, see Learn about security and roles.

f. Select **Continue**.

g. **Optional:** Add parameters to your page URL by selecting **+ Add**.
For more information, see Manage UI Builder pages and page variants.

h. Select **Looks good**.

i. Enter `Manager Start Page` as the name for the page variant.

j. **Optional:** Add one or more audiences for this page.
If an audience you need isn't listed, you can choose the **Open audiences in the platform** link to create one.

k. **Optional:** Add conditions for when to display the page or tab.

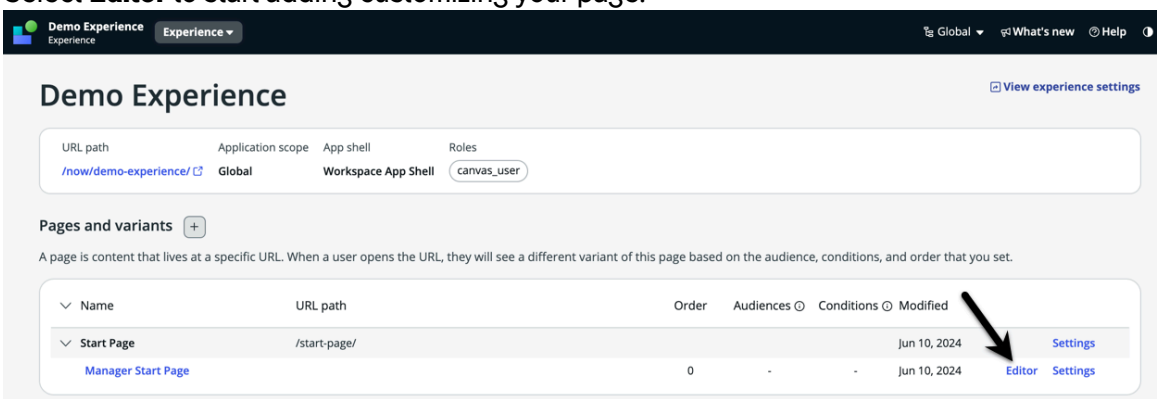
l. Select **Continue**.

m. On the next screen, select **Build responsive**.

n. Select **Create** to create your blank page.

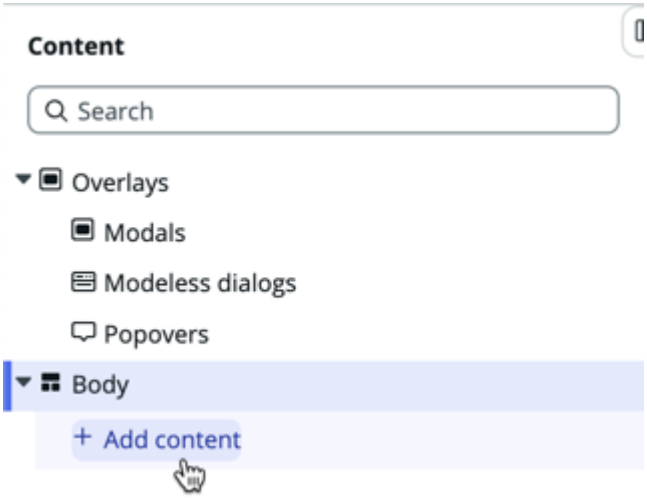
Congratulations! You've created your first page! The page is empty of content. You add components to the page to build functionality to it. Components are the building blocks of a page. UI Builder comes with many components ready for you to add to your page. Components can be as simple as a **Heading**, or as complex as a **List**.

4. Select **Editor** to start adding customizing your page.



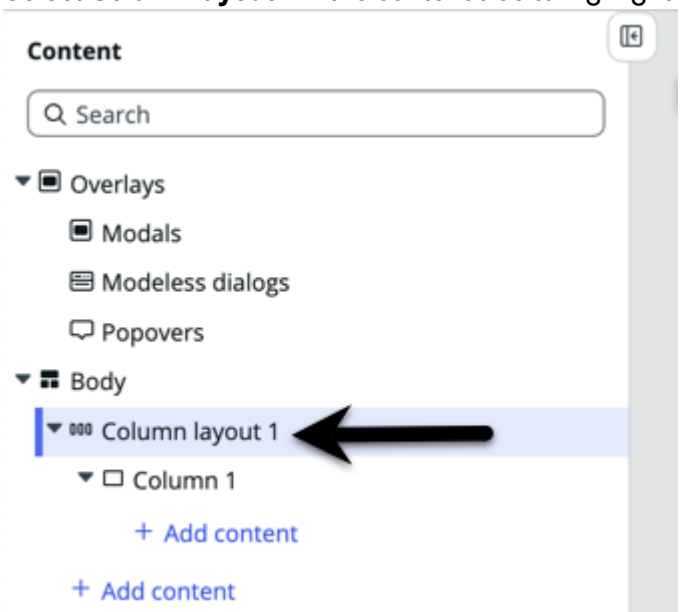
5. Add a container component to your page.
A container is what holds your components. Think of a container as an area of the page where you add information, images, or functionality (your components). You can have as many containers on a page as you want, with as many containers within containers, with as many components in the containers.

a. Select **+ Add content** in the content tree.

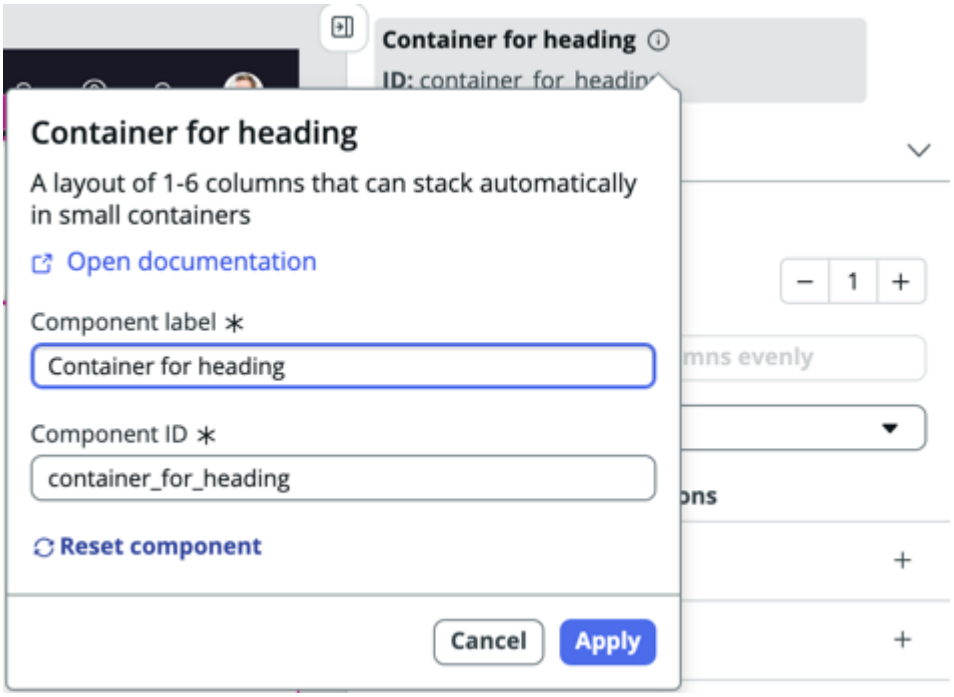


b. Select the **Single column** layout in the toolbox.

c. Select **Column layout 1** in the content tree to highlight the container.



d. Select the component name in the configuration panel.

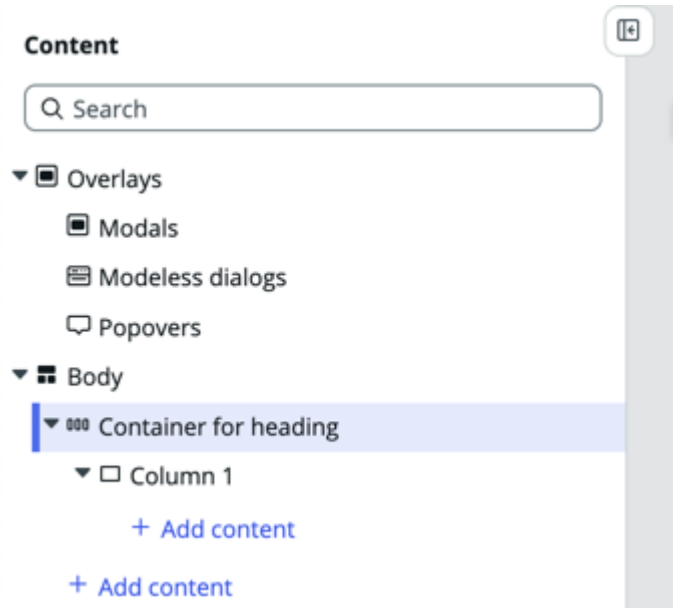


e. In the **Component label** field, type `Container for heading`.

f. In the Component ID field, type `container_for_heading`.

g. Select **Apply**.

See that the column layout name changes to **Container for heading** in the content tree. The content tree is an important concept. The content tree is an easy way to see the structured layout of your page. The content tree is especially important when you have multiple components on your page. You select the component in your content tree to highlight the component on the page, making it easier to build your page. You can do a text search for a component.

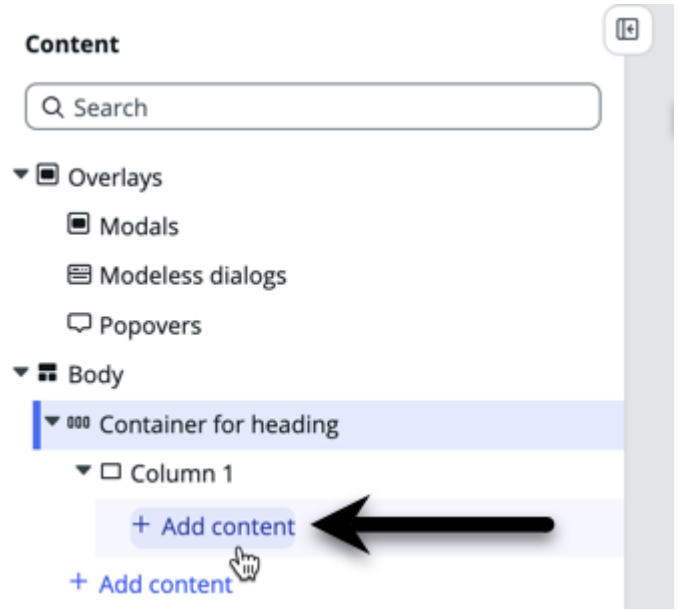


For more information on component IDs, see Component ID.

You've successfully added your first column layout to your page.

6. Add a **Heading** component to your column layout.
   You can add components to the page in different ways. For more information on the ways you can add components to your page, see Add and configure components.

   a. Select **+ Add content** in the content tree below the column layout created in step 5.



   b. Type `heading` into the search and select the **Heading** component.

   c. Select your new **Heading** component in the content tree to highlight it.

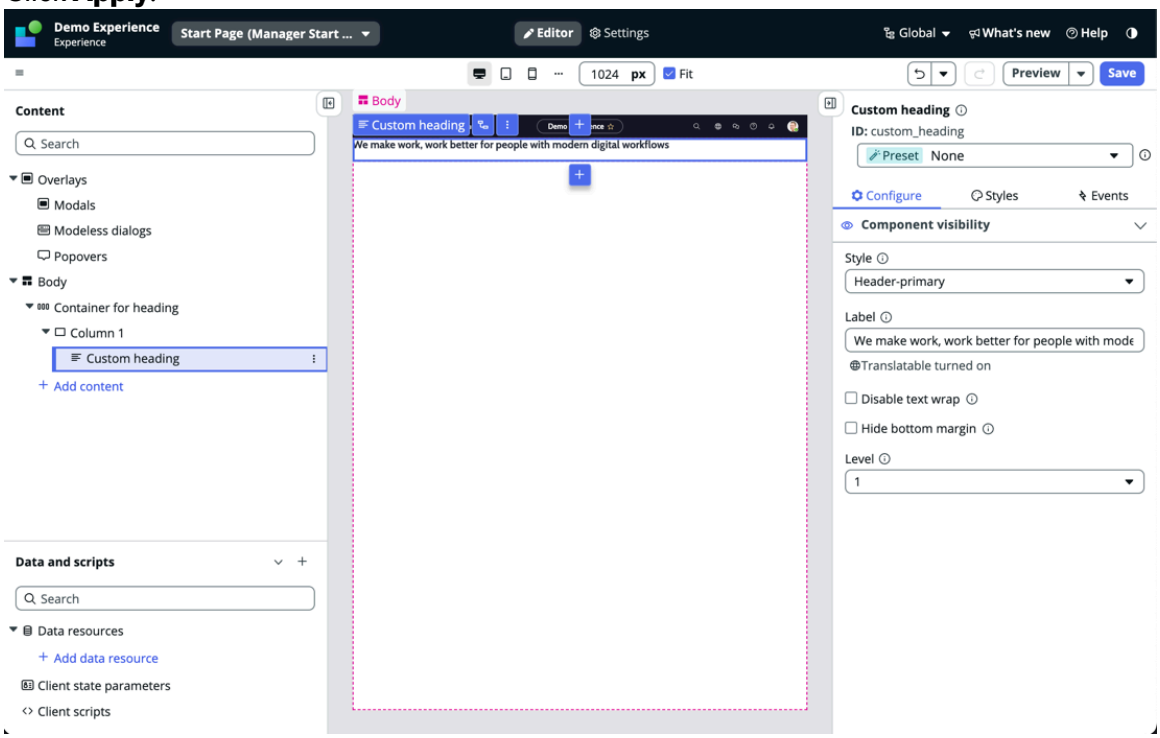
   d. Select **None** in the configuration panel.

   e. Select the component name in the configuration panel.

   f. In the **Component label** field, enter `Custom heading` as the unique label for the heading component.

   g. In the **Component ID** field, enter `custom_heading` as the unique ID for the heading component.

h. Click **Apply**.



7. Customize the heading component.

a. Select the heading component in the content tree.

b. Select the **Configure** tab in the configuration panel.

c. In the **Label** field, enter the text of your heading, such as `My new heading`.

d. The Style changes the size of the heading text.
For example, if you select **Header-secondary**, the text is smaller. Different headings sizes are useful if you have two headings and want the second heading smaller that the primary heading. For more information on configuring components, see Configure components in UI Builder.

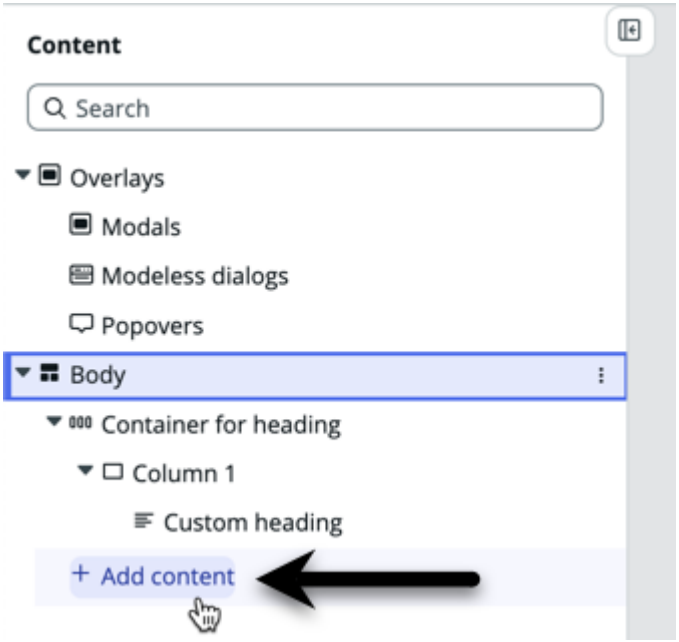
e. Leave the **Level** as **1**.

f. Select **Save**.

You've added and customized a heading component to your page.

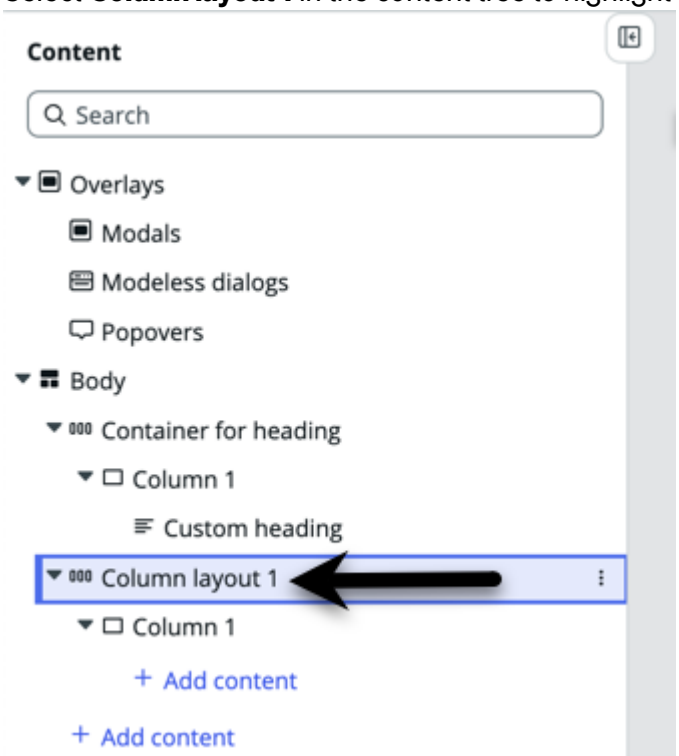
8. Add a second column layout to your page.

a. Similar to before, select **+ Add content** in the content tree.



b. Select the **Single column** layout in the toolbox.

c. Select **Column layout 1** in the content tree to highlight the container.

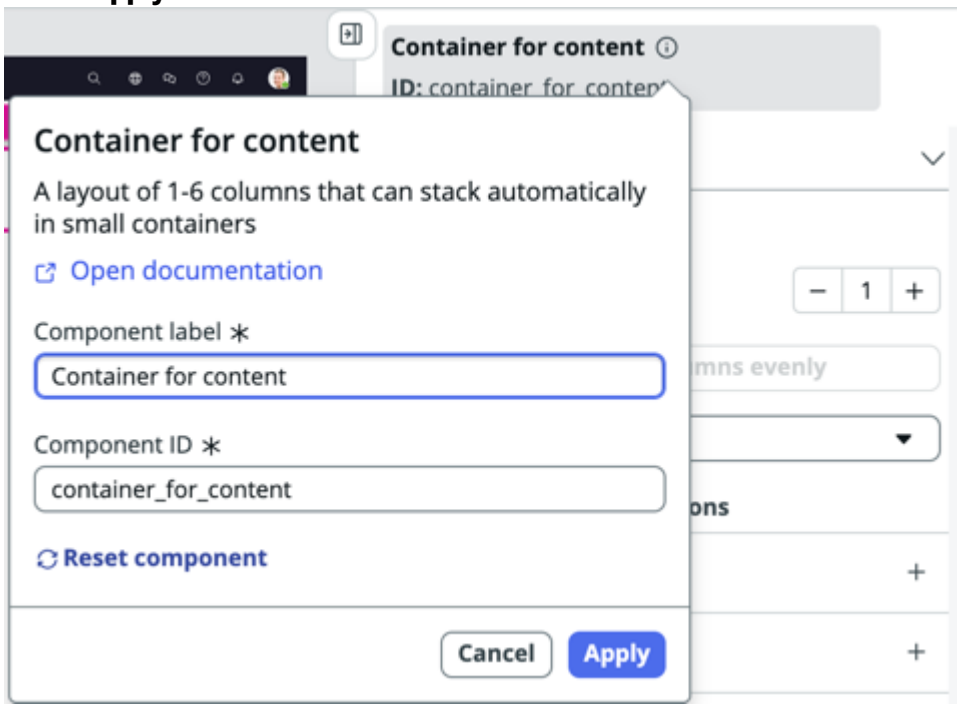


d. Select the component name in the configuration panel.

e. In the **Component label** field, type `Container for content`.

f. In the **Component ID** field, type `container_for_content`.

g. Select **Apply**.



h. Select **Save**.

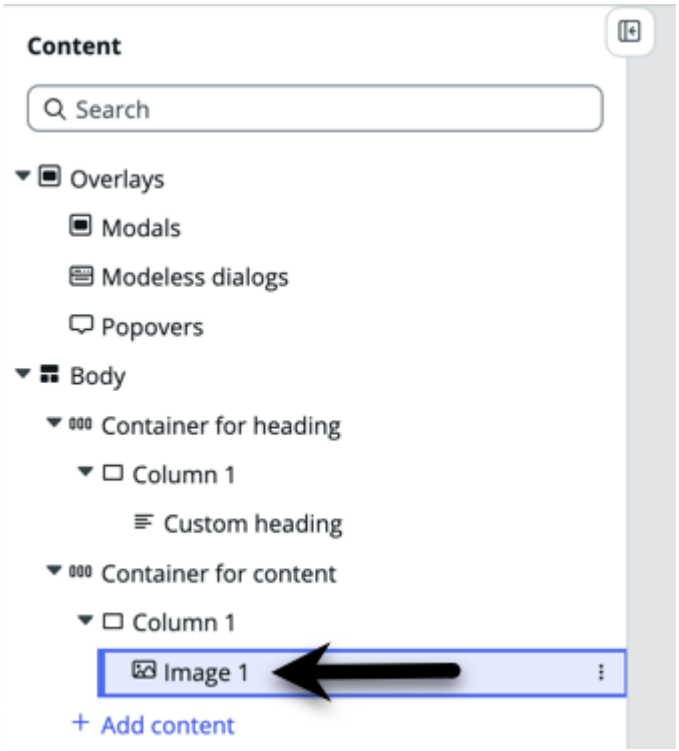
You've successfully added your second container component to your page.

9. Add an image component to your page.
An image is a nice way to add a visual for your page. In this quick start, you add a stock photo that comes with UI Builder. But you can add any image that is available to you.

a. Select **+ Add content** in the content tree below the container created in step 8.

b. Type `image` into the search and select the **Image** component.
A default image is loaded in the image component. You can add your own image by adding a URL to the image. You can use images hosted on the internet or images in the ServiceNow AI Platform®. If you use an image hosted on an external site, you must use the `https` protocol for security.

c. Select the **Image 1** component in the **Content** tree to highlight the image.



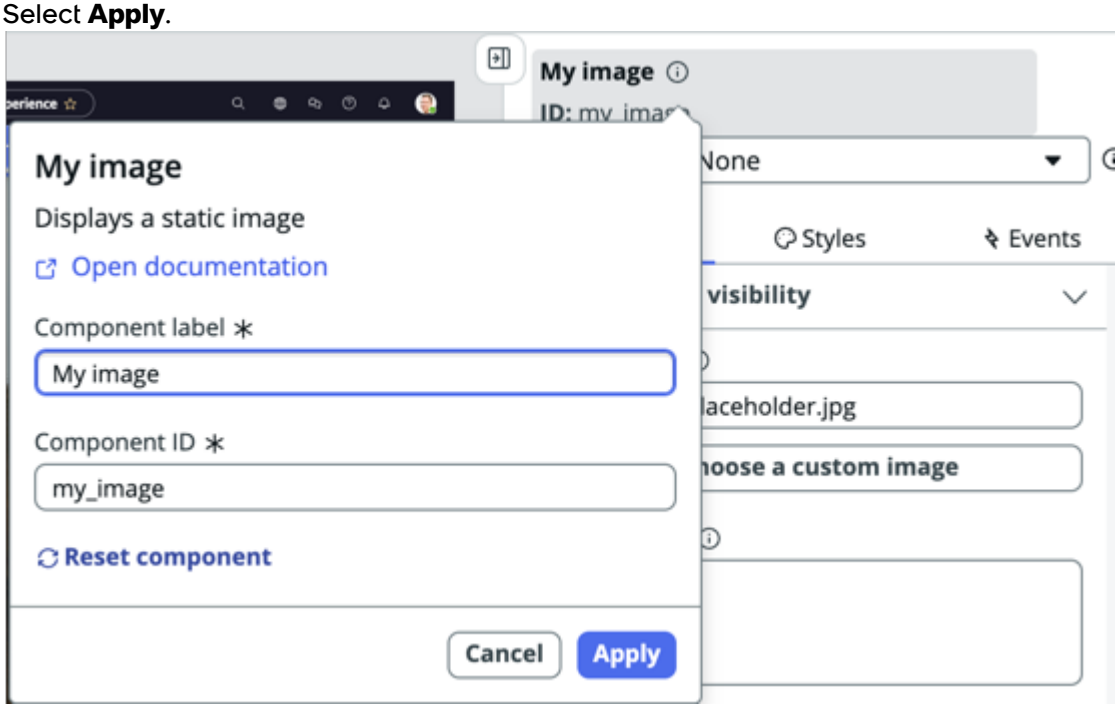
d. Select **None** in the configuration panel.

e. Select the component name in the configuration panel.

f. In the **Component label** field, type `My image`.

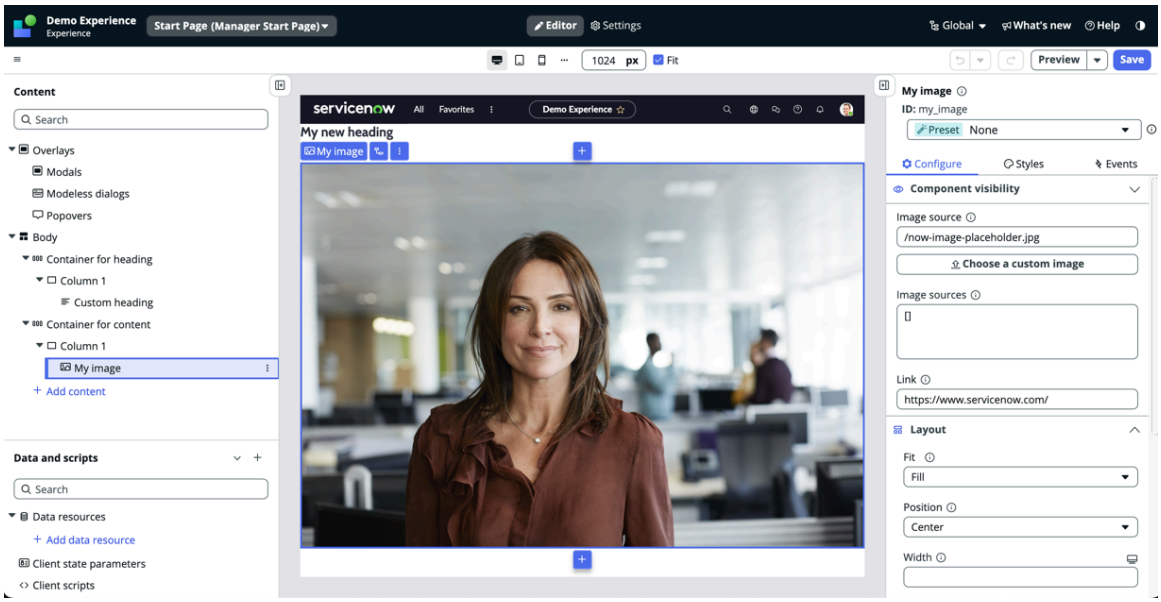
g. In the **Component ID** field, type `my_image`.

h. Select **Apply**.

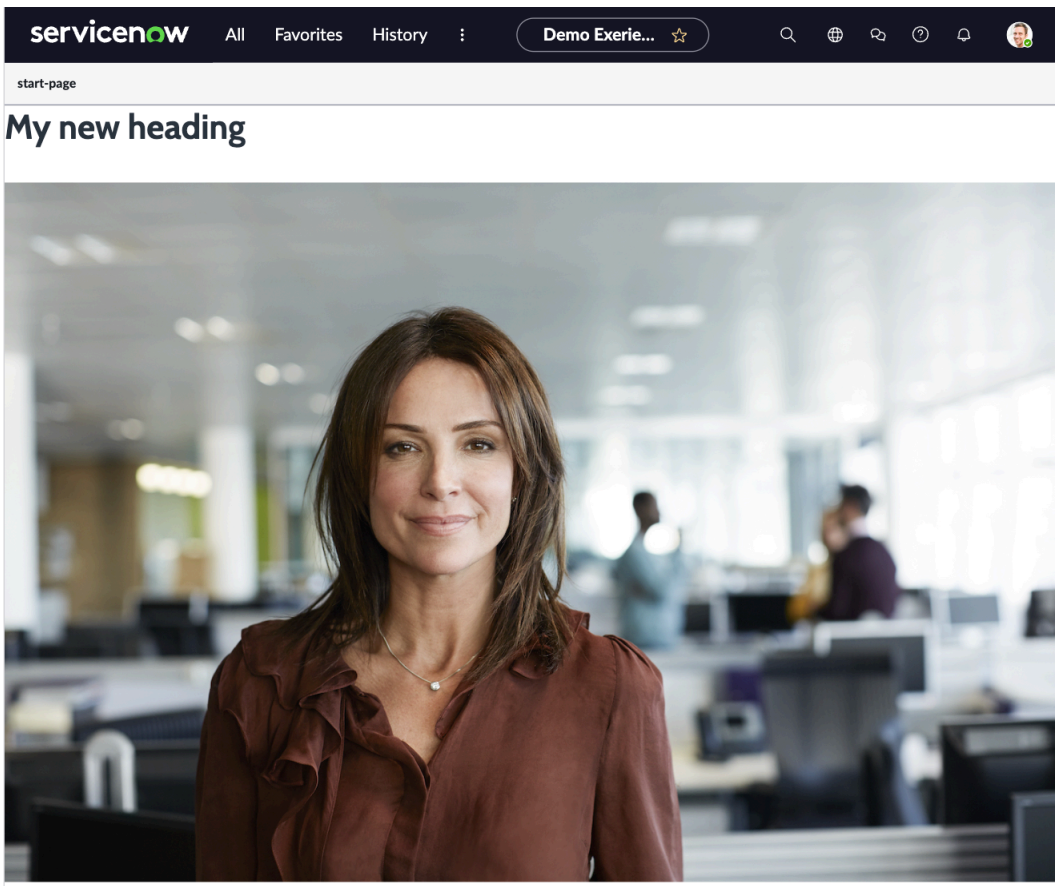


i. Select **Save**.

You added an image component to your page.



10. Select **Preview** in the UI Builder header to preview your page.
Congratulations! You completed the UI Builder quick start. Your page generates a preview of how it looks in your Workspace or portal experience.



Result

1. Created your first page in UI Builder.
2. Added your first container component to the page. You changed the label for the container.
3. Added a **Heading** component as a title to the container. You changed the text of the heading. You also changed the label.
4. Added a second container component to your page. You changed the label for the container.
5. Added an **Image** component. You changed the label of the image component.
6. Saved your new page.
7. Previewed your page in the browser.

You successfully completed the UI Builder quick start!

UI Builder tutorial

Learn how to use the basics of UI Builder to create a page called My Tutorial.

Before you begin

Role required: ui_builder_admin

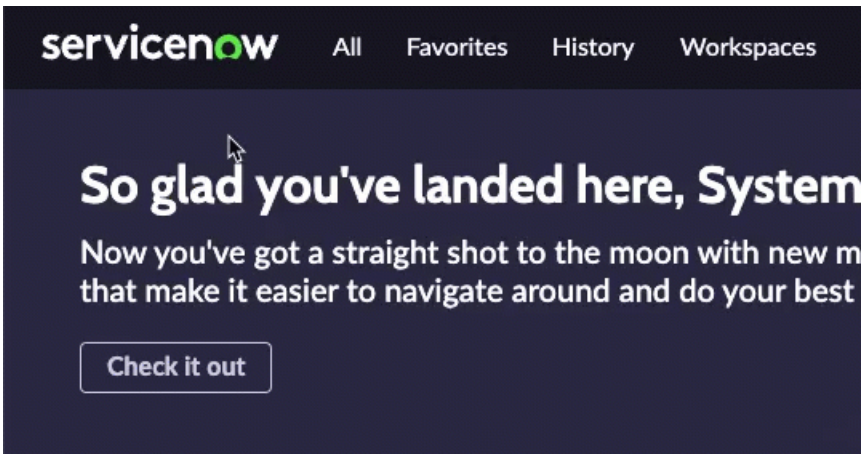
In this UI Builder tutorial, you perform the following tasks to build a page in UI Builder:

- Start UI Builder.
- Create a page for your workspace or custom portal experience. For more information about creating pages, see Create a page in UI Builder.
- Change the layout of the page to have two columns. For more information, see Organize components in UI Builder pages.
- Build your page by adding two container components.
- Rename your container components in the content tree.
- Add a **Heading** component and a **Button** component to the first container. Add a **data visualization component** to the second container. For more information about components, see Customize UI Builder pages using components.
- Configure your components as follows:
  - Link the button to the ServiceNow® website.
  - Connect the **Data visualization** component to a data source to display task data. For more information about data resources, see Dynamically expose data in UI Builder pages (advanced feature).
- Save your page often.
- Preview your page to see how it looks in a browser.

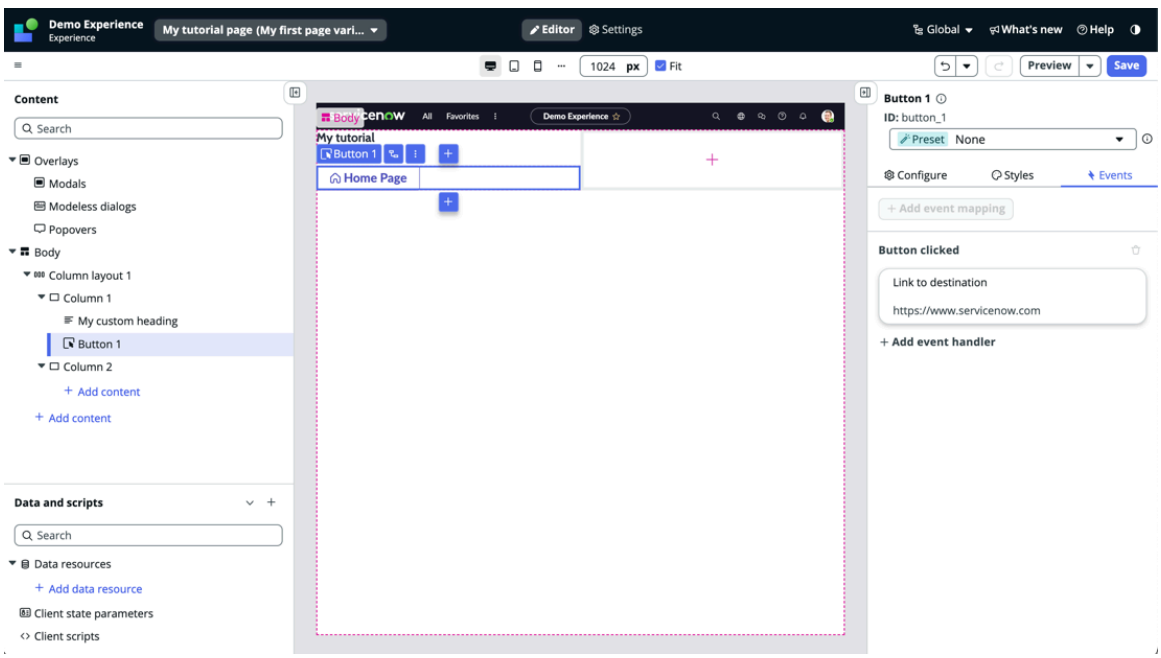
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
   You can also type `UI Builder` directly in the **Filter navigator**.

Start UI Builder



2. From the UI Builder home page, choose an experience you want to work in.
   If you don't see any experiences listed in which to work, contact your administrator to get access to an experience or create an experience. For more information, see Configure how users interact with your applications in UI Builder.
3. Create a page.
   Congratulations! You created your tutorial page! The page is empty of content. You add components to the page to build functionality to it. Components are the building blocks of a page. UI Builder comes with many components ready for you to add to your page. Components can be as simple as a **Heading**, or as complex as a **List**.
4. Change the layout of the page to a two-column layout.
   For more information about layouts, see Organize components in UI Builder pages.
5. Add a **Heading** component, and a **Button** component to the left column.
   You added and configured the **Heading** and **Button** component for your page.



6. Add a **Data visualization** component to the container.

A data visualization component contains data that you display in a visual manner. Configure the data visualization component parameters. Then, add a data resource to it.
You've added a **Data visualization** component to the right column and configured it. You also added a data resource to bring in customer account data.

7. Save the new page one last time.

8. Now preview your page to see what it looks like in a browser.
Congratulations! You completed the UI Builder tutorial.

Result

1. Created a page in UI Builder.
2. Changed the layout of your page to have two columns.
3. Added a **Heading** component as a title to your left column.
   - Changed the text of the heading.
   - Changed the label of the heading.
4. Added a button component below your heading component.
   - Configured your button.
   - Added an event handler for your button.
   - Added an event handler to set up a link to a web page. When you click the button, it opens the web page.
5. Changed the label for the right column.
6. Added a **Data visualization** component.
   - Configured the data visualization component.
   - Added a data resource to bring in task data.
7. Saved your new page.
8. Previewed your page.

UI Builder and configurable workspaces

Use UI Builder to create pages for your configurable workspace experiences.

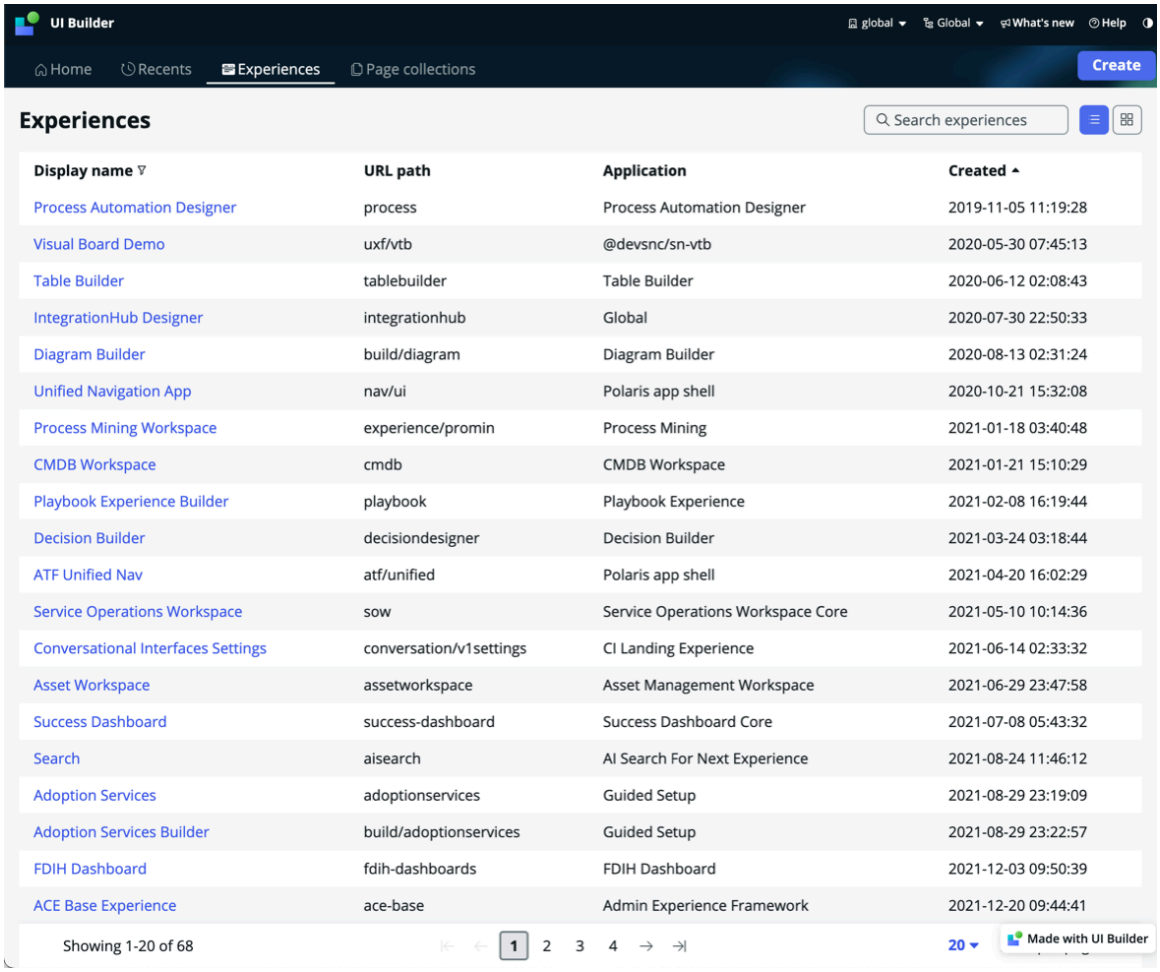
A workspace is a collection of tasks and workflows in a single focused working area that enables a user to efficiently complete an entire job. It includes tools that a user can employ to quickly and easily assist customers and resolve questions and issues. A workspace also includes features that enable a user to be more efficient, including a multi-tab interface for managing multiple cases and a contextual display that provides quick orientation to the current task.

You can use UI Builder to create pages for your workspace experience. UI Builder provides base system components that you can use to build workspace pages.

To create a page in UI Builder for a Workspace experience, you navigate to **All** > **Now experience framework** > **UI Builder** to open UI Builder.

The UI Builder home displays your available experiences under the **Experiences** tab. If you are working in a Workspace experience, you should see the experience listed here.

UI Builder home



Experience settings for UI Builder

Learn about the UI Builder experience settings to build your own workspace and custom portal experiences.

By changing the settings of the workspace or portal experience that you're working in, you can affect how your users interact with the experience, how the experience looks, and how users navigate to and from the experience.

Before you can edit the experience settings, you must be in the correct application scope. If you're in a different scope, the experience settings are read-only. To change your application scope, go to the main header, select the application picker (

), and then select the application scope that you want. For more information about the application scope, see Learn about security and roles.

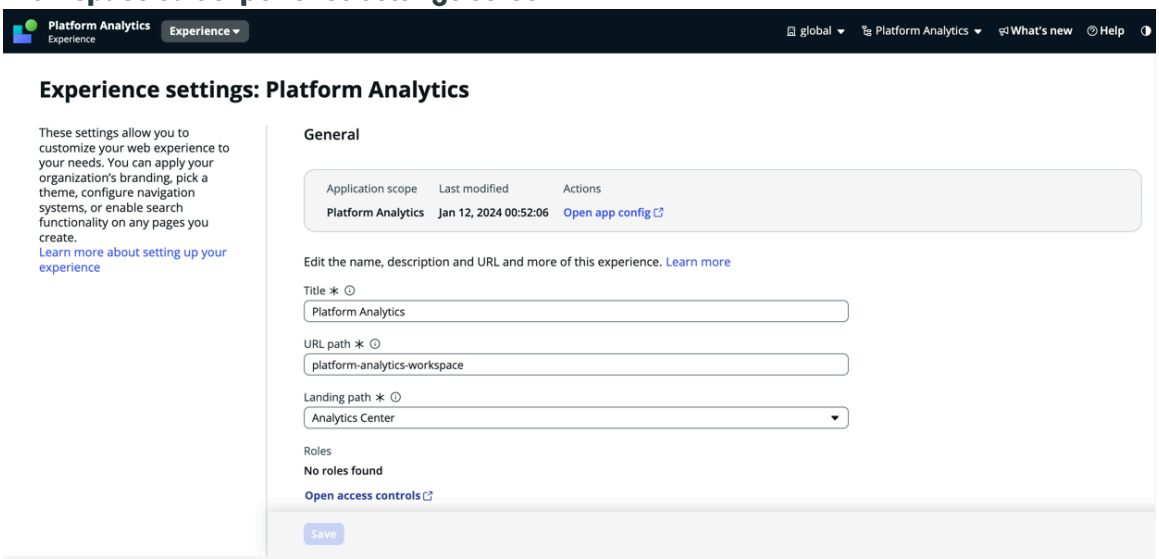
Workspace experience settings

You can change these settings for your workspace experience:

- Modify or add some general settings for your experience. For example, you can change the title, description, and the path of the workspace or portal.
- Apply your organization's branding to all pages in your experience or to a page within your experience.
- Configure your side navigation settings to add pages to the side navigation in your experience.

- Turn on or turn off the notifications for your experience.
- You can change the global search settings for your experience. For example, you can choose to show or hide the search bar in your experience or change the search source that determines where the search results come from.

Workspace edit experience settings screen



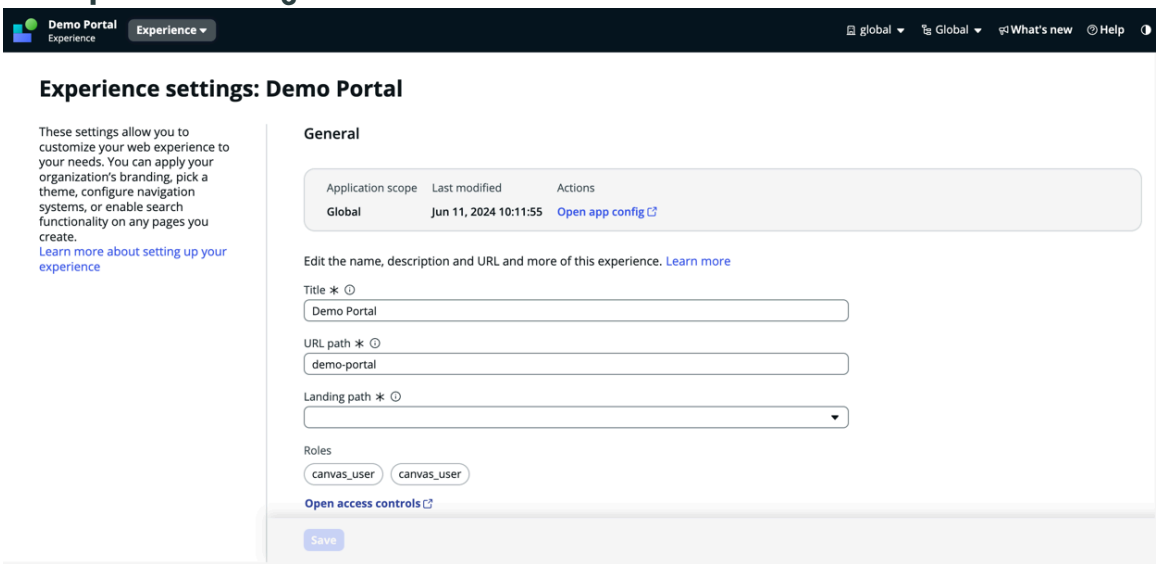
To learn more about the workspace experience settings, see Configure UI Builder workspace experiences.

Portal experience settings

You can change these settings for your portal experience:

- Modify or add general settings for your experience. For example, you can change the title, the description, and the path of the workspace or portal.
- Apply your organization's branding to all of the pages in your experience or to a page within your experience.
- Set up the navigation and menu settings in the app shell of your portal experience. The app shell, which is the wrapper of the portal contents, is similar to a Google Chrome web page.
- Control whether the global search functionality is visible to the users of your portal experience. For example, you can enable or disable the global search for either your public or private pages.

Edit experience settings screen



To learn more about the portal experience settings, see Configure UI Builder portal experiences.

UI Builder glossary

Learn about the terms and concepts used in UI Builder (UIB).

action

Actions in UI Builder are specifically activity on a page or within a page component. Events and event handlers are used to add actions. For example, add a button component to a page and then add an event handler to apply an action for the button, such as opening a web page.

app shell

App shells are the static elements of a web experience (for example, the header, footer, and menu navigation) that stays with the end user as they navigate throughout the experience. App shells are primarily used and supported in Workspace and Portal experiences.

binding

See data binding.

client script

Client-side JavaScript that interact with components and client state parameters on a page. Client scripts are mapped to events as event handlers in UI Builder.

client state parameter

Page variables that are defined for a page to store a piece of data (a client state) only for that page. For example, create three client state parameters to store the input needed to create a record and specify when to refresh the list. Page variables can be updated using client scripts and events to make a page dynamic.

component (UI Builder)

Use the UI Builder library of components to build pages. Components have an interface that an end user can view and interact with. Components can talk to each other through events and properties. Commonly used components include Heading, Image, List, Form, and Button.

component id

Used to reference a component when adding a script or binding data to the component. A component ID is automatically created (based on the component label) when you add a component to a page, but the component ID can be edited.

component preset

Use to apply predefined configuration values and event mappings to components. Presets apply prebuilt configurations to component properties and events handlers. Presets are only available for certain components.

component properties

Available in the Configuration panel and used to configure a component. Each component has unique properties. Component properties are specified within each tab on the Configuration panel: Config, Style, and Event. Some components have presets available. Use the component presets to set component properties automatically.

controller

A type of data resource that includes data and event logic and enables component presets. Controllers are added automatically when using a page template. There are two types of controllers:

- Data controllers contain data resources and can be manually added to a page
- UI controllers are added to pages when using page templates and can't be added manually. Creating controllers isn't supported currently.

data binding

The process of associating data with a UI element that displays the information.

data resource

A dynamic, reusable way of defining what data to fetch for a page's components.

entity view action mapper

Also known as EVAM. Standardizes the format for displaying data in cards and lists.

event (UI Builder)

Action a user takes (such as selecting a button) or an occurrence that happens on a page. Most UI Builder components, pages, and data resources have default-associated events. Use event handlers with the events to add additional actions to pages.

event handler

An action performed when an event occurs.

event mapping

The process of identifying an event handler to run when an event occurs.

macroponent

Core data structure that drives UI Builder pages. Fields contain JSON that builds the page.

modal

User experience window that overlays another content window and takes control of the user experience.

now code editor

A rich-text editor that supports CSS, HTML, JavaScript, XML, and JSON. Use Now Code Editor to change UI configuration, data resource configuration, styles, events, client-side scripts, and server-side scripts in Next Experience UI Builder components.

page

Collection of column layouts, columns, and components. Create or customize multiple UI Builder pages for workspace and portal experiences.

page collection

Group of pages that can be reused in experiences within tabs or modals.

popover

A page overlay that enables users to continue using the rest of the page. Popovers can be configured just like UI Builder pages with text, components, images, fields, and menu items.

repeater

In UI Builder, a repeater is a component that acts as a basic loop that repeats the data you provide in multiple components. Repeaters use an array or an array of objects. Repeaters bind values to a data array property. For example, [{"task": "A"},{"task": "B"}], repeats the content inside it two times.

UI Builder (UIB)

A WYSIWYG web user interface builder. UI Builder enables developers to build new pages or customize existing pages for Agent Workspace and portals using Now Experience UI Framework components.

variant

Version of a UI Builder page with access controlled by role or condition. Create variants of pages to target experiences for different audiences. For example, create a home page for agents and a variant for managers at the same URL. Alternatively, create a page variant that users see under different conditions.

Learning UI Builder

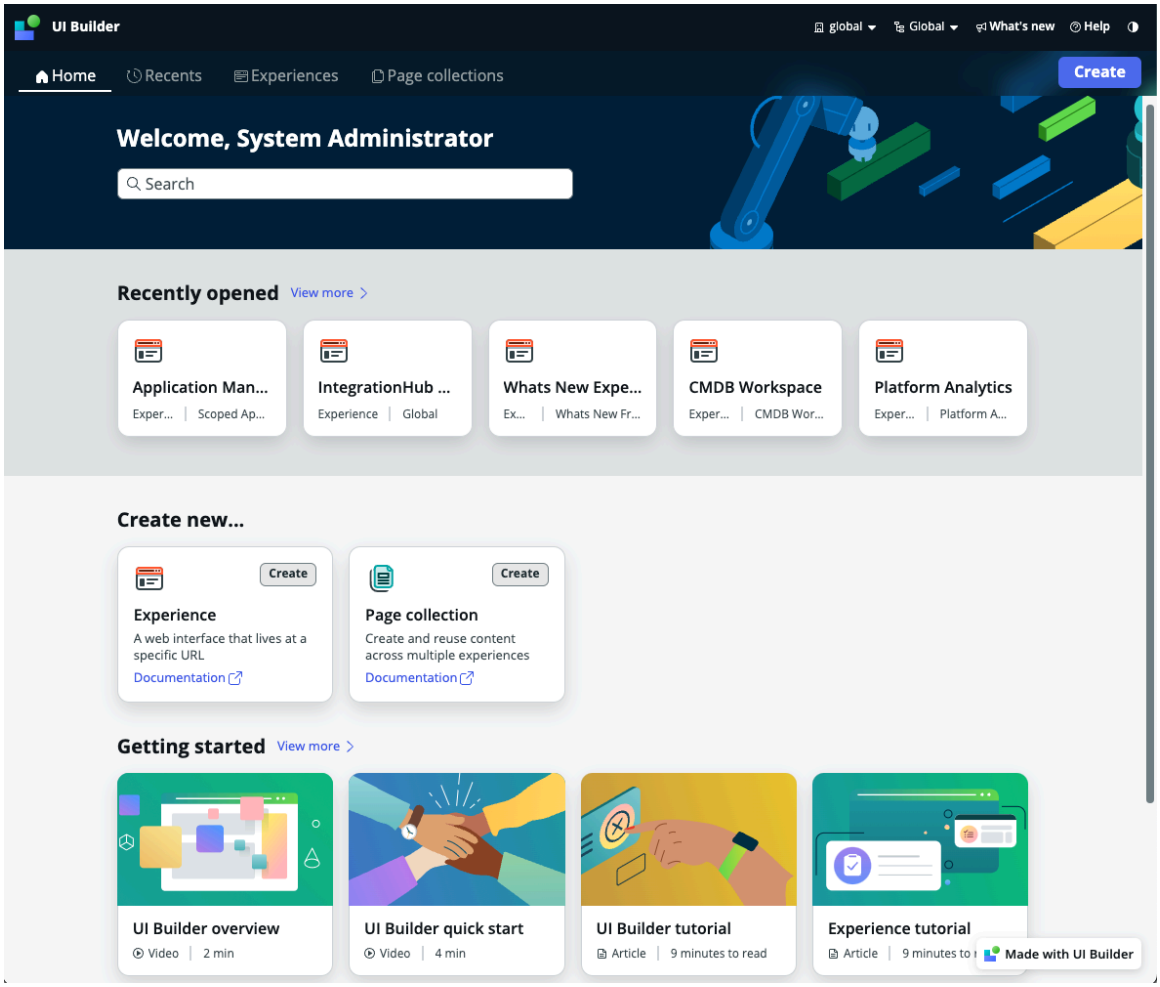
Learn how to navigate the UI Builder and explore some common use cases to learn by example.

Navigate the UI Builder application

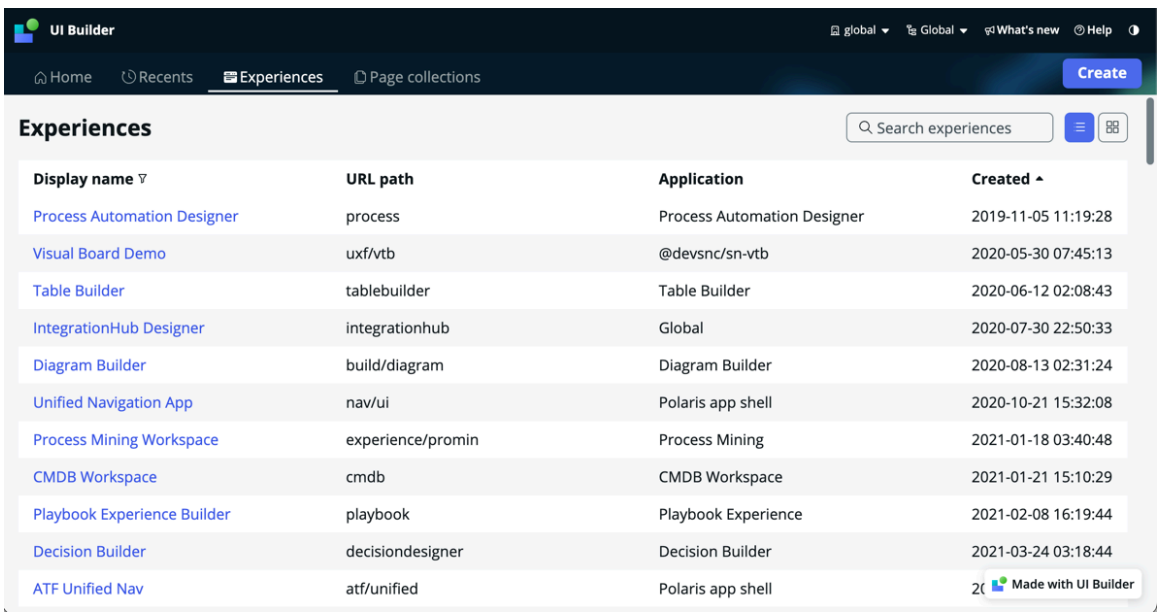
Learn how to navigate through UI Builder.

UI Builder home

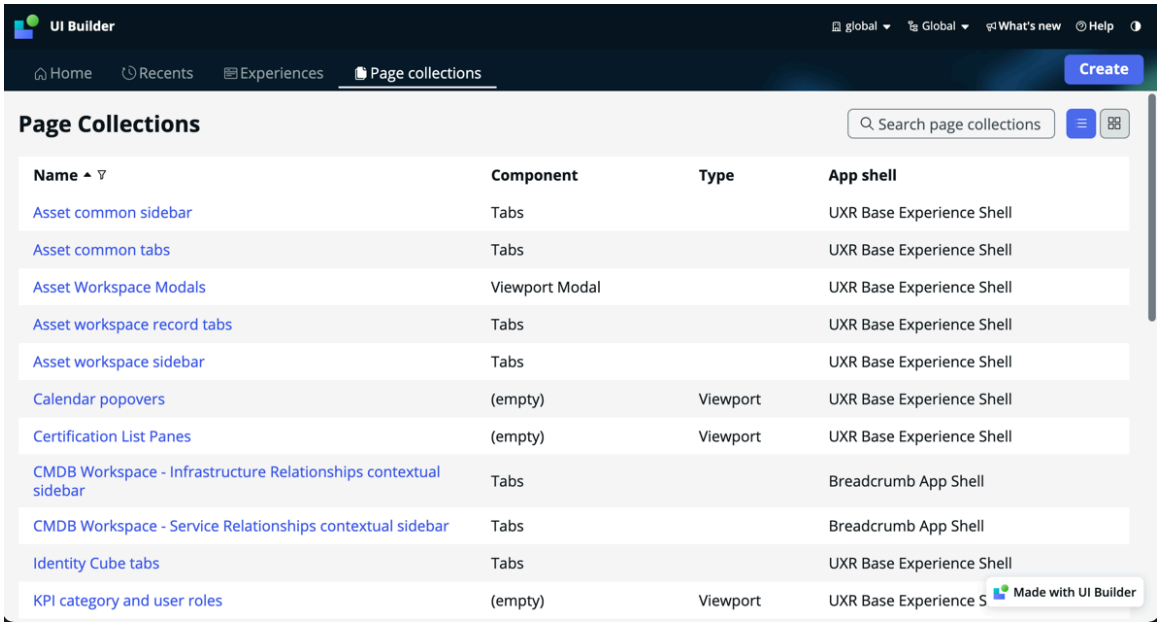
The UI Builder home page provides easy access to recently opened experiences, options for creating experiences or page collections, and help for getting started using UI Builder. You can also use the search bar to quickly find your experiences by name.



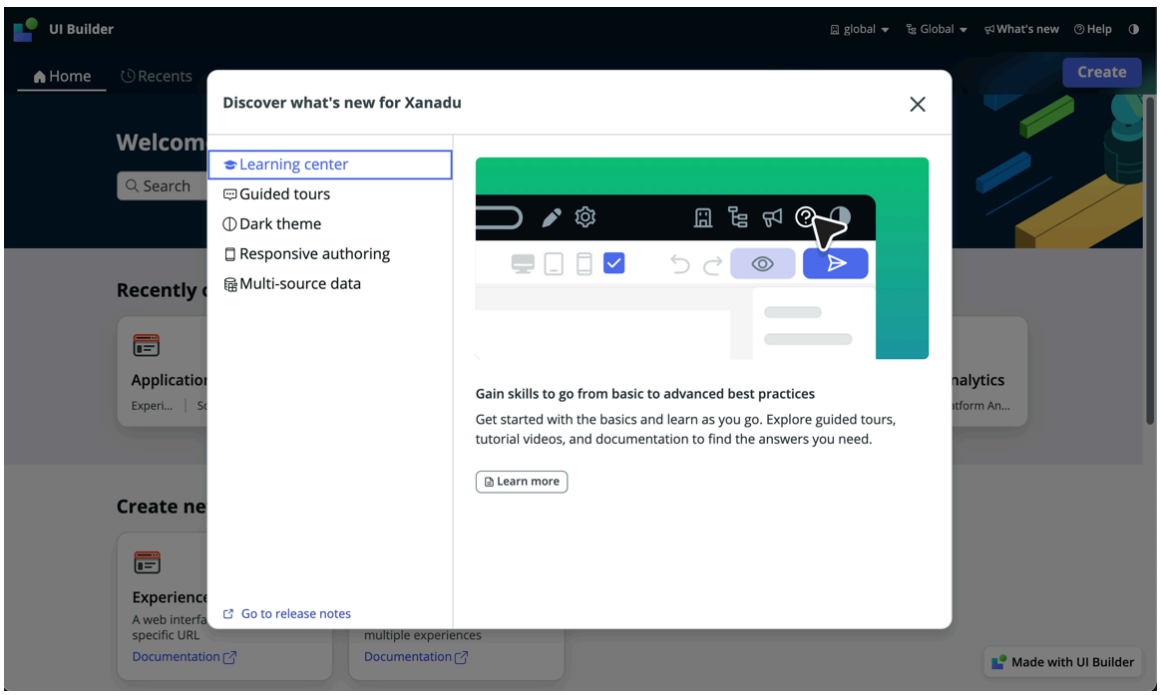
You can access the list of experiences in the top by selecting the **Experiences** tab in the top bar.



You can access the list of page collections in the top by selecting the **Page collections** tab in the top bar.



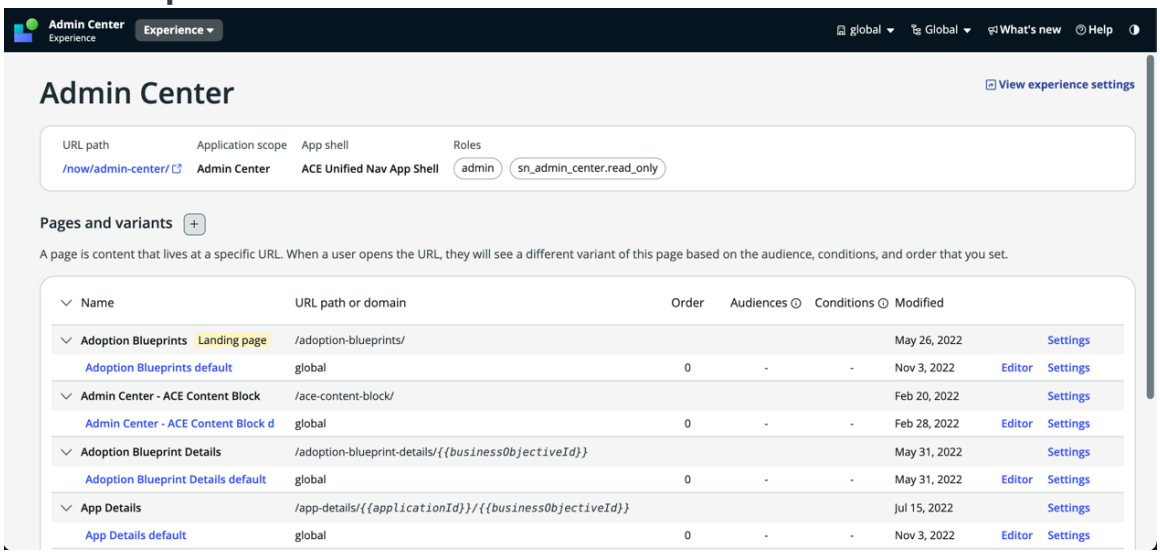
Select the **What's new** link in the header to view the latest changes to UI Builder.



UI Builder experience view

The UI Builder experience view is a central place to view and understand the structure and details of an experience. Use the **Page and Variants** list to select a page to edit.

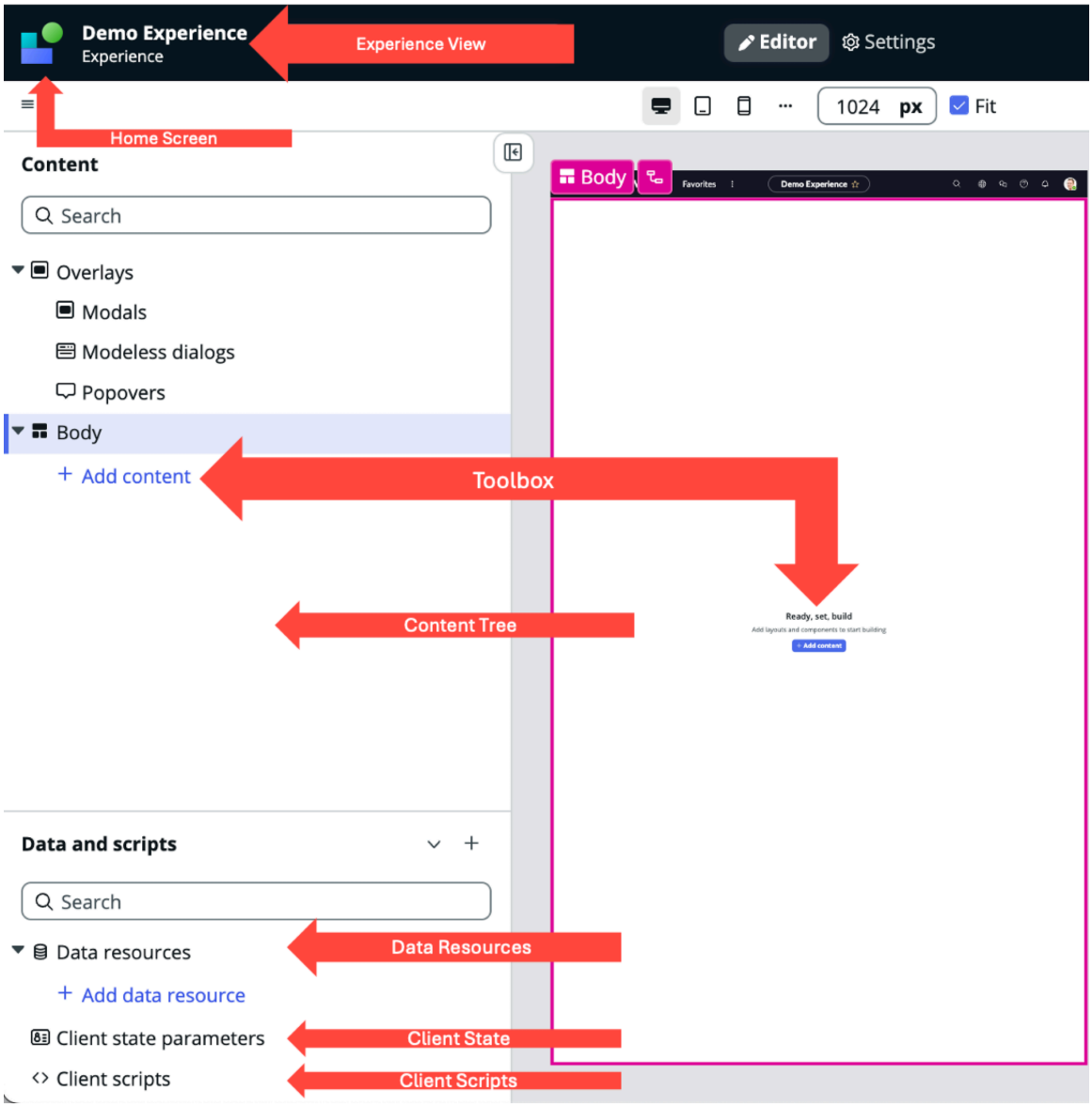
UI Builder experience view



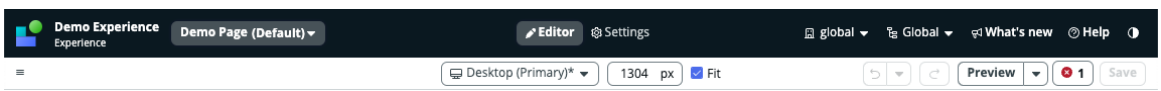
Structure of UI Builder

UI Builder is separated into the following areas.

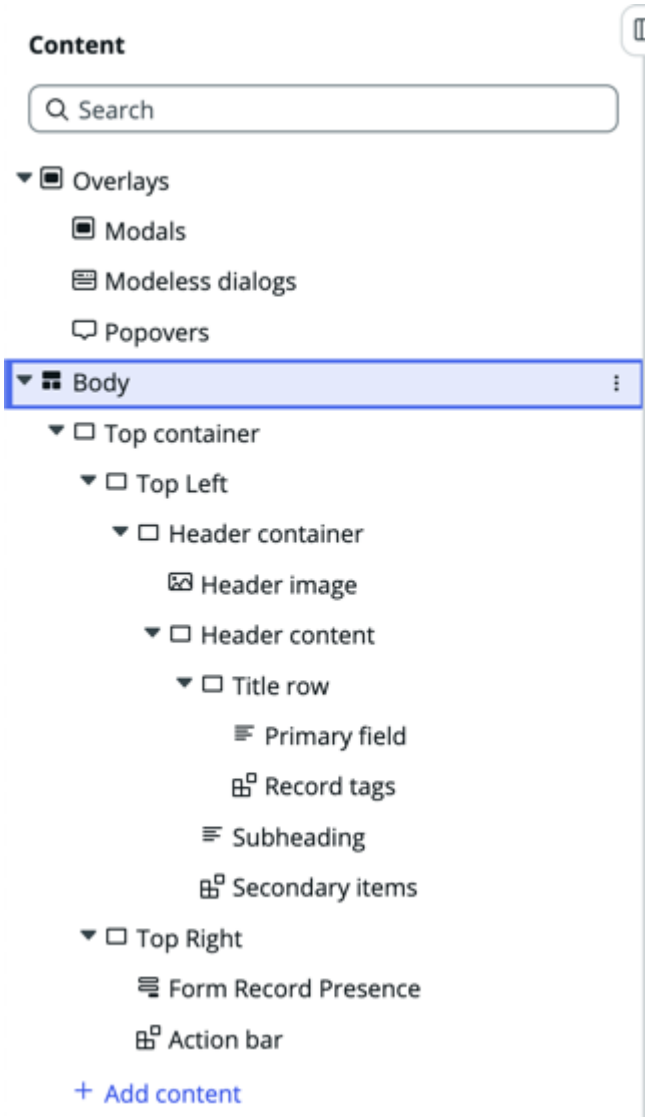
- Use the left vertical bar for access to the following in UI Builder:



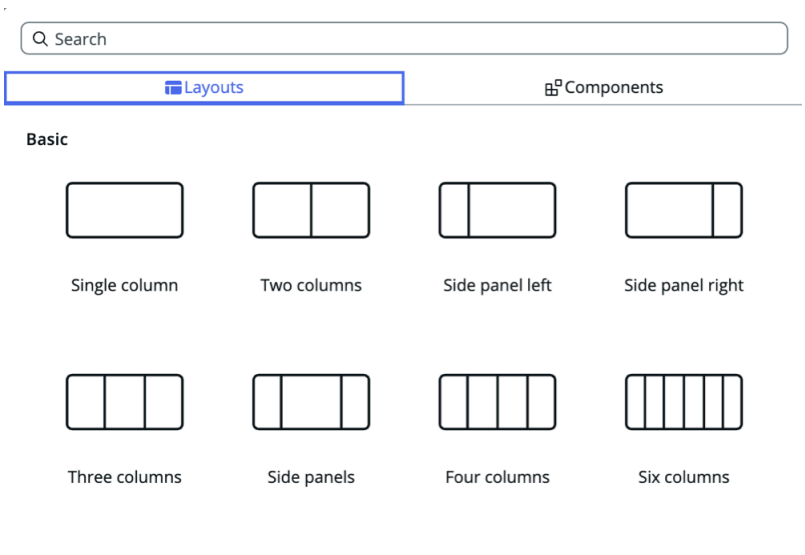
- Use the top bar to access pages in the experience and view Help. Change page settings, such as availability, order, and audience. Preview and open the URL path for the page you're editing to see what the page looks like for users. You can also select a domain and set application scope. View page errors and warnings by selecting the icon next to the save button. And finally, click the **Save** button often to save your page.

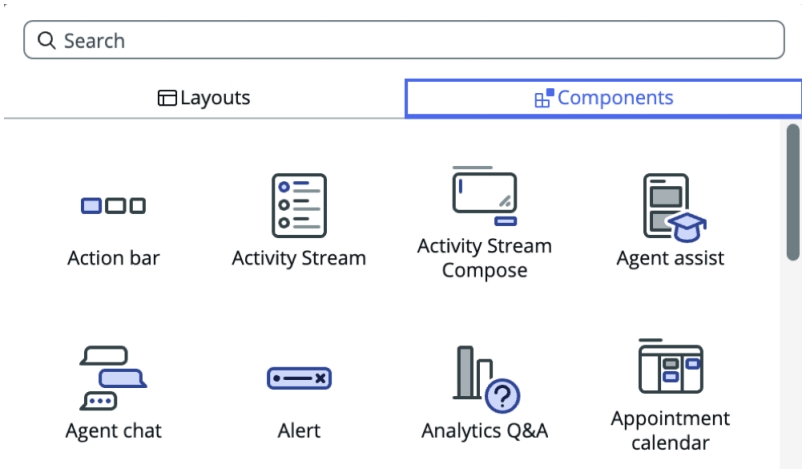


- Use the content tree to view and edit column layouts, columns, and components on your pages.

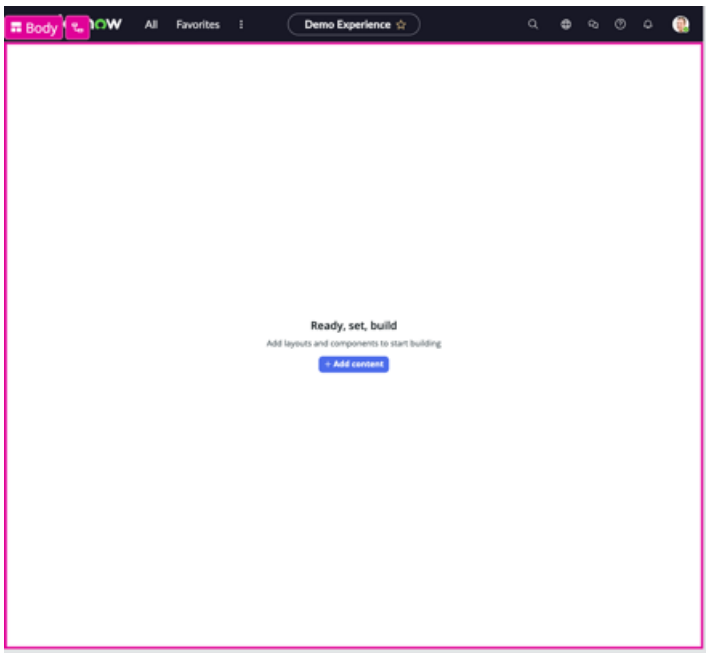


- Add column layouts and components to your page using the UI Builder toolbox.

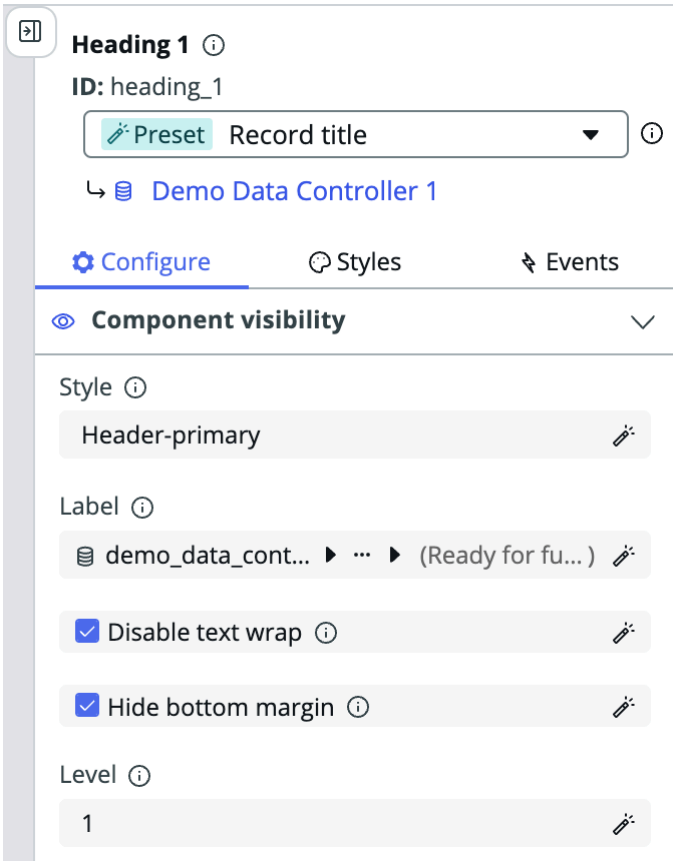




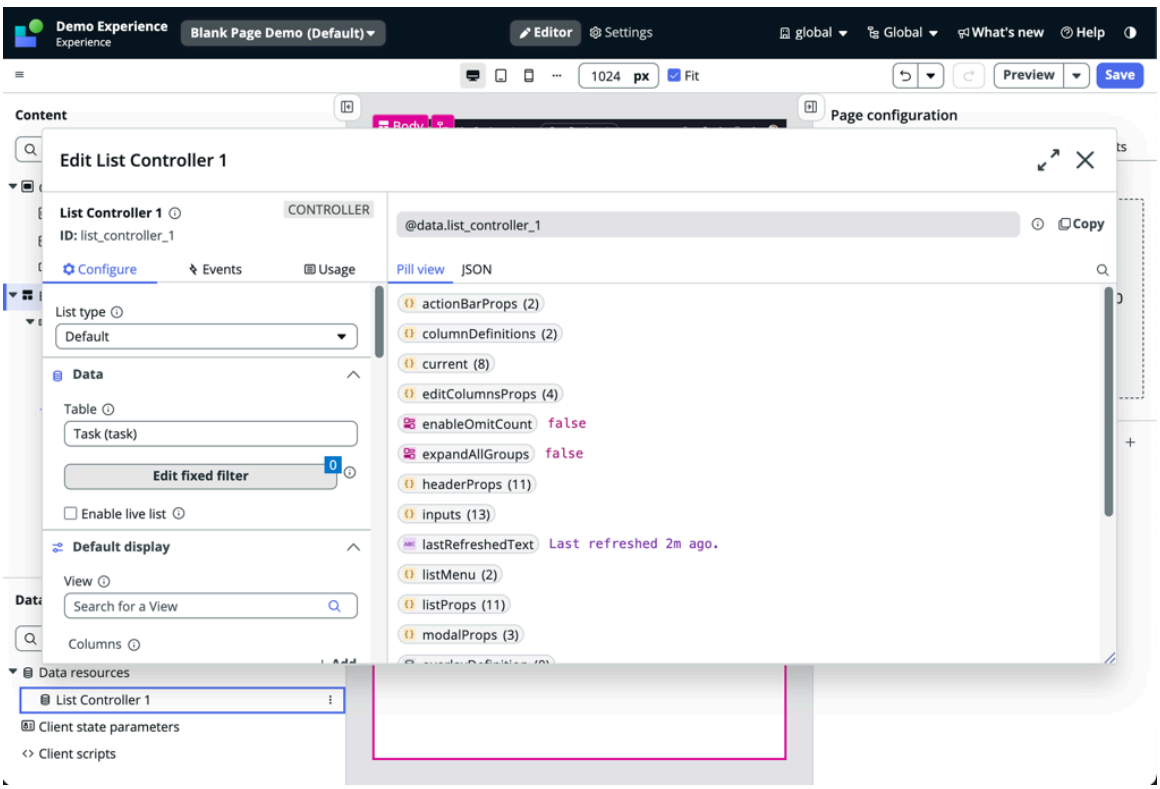
- Work in the UI Builder staging area to build and modify your pages.



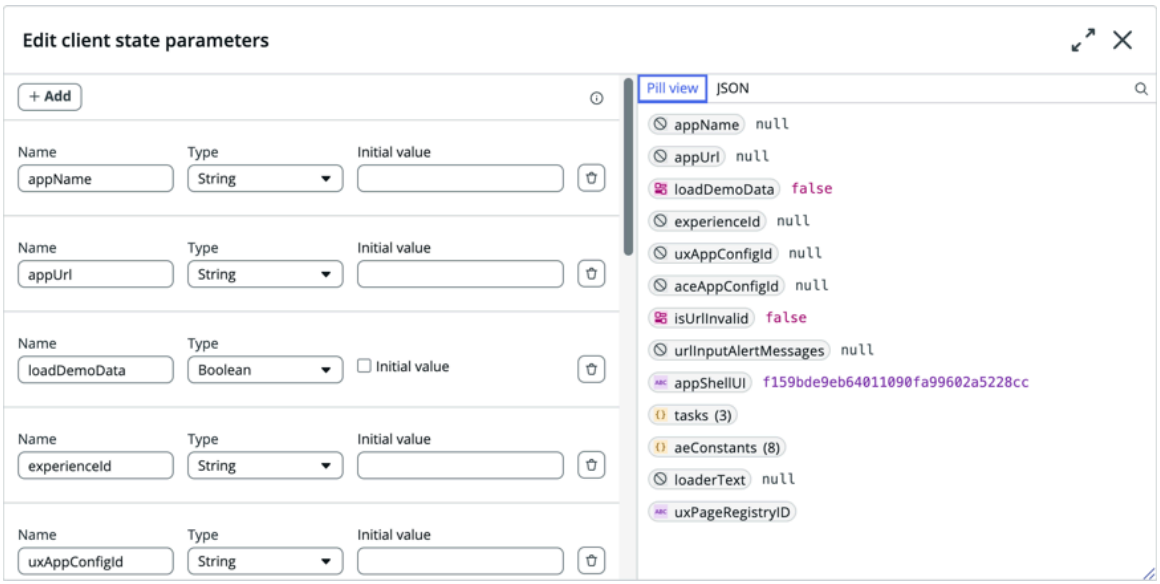
- Go to configuration panel to modify column layouts, columns, components, styling, and events.



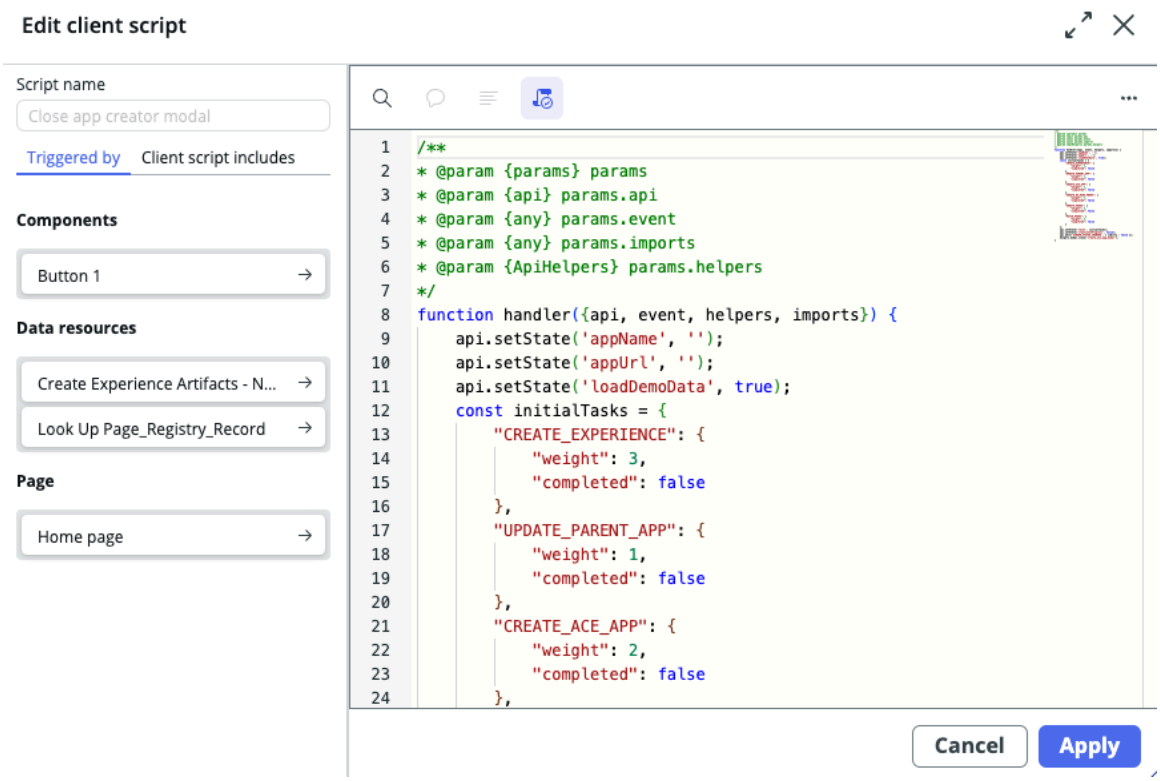
- Data resources: Bind data to your components using data resources to dynamically expose your data from tables, records, or other elements on your page. Data resources enable you to reuse your components. See Connect data to your components for more information.



- Client state: Set the parameters for the client state for your page. The client state parameters consist of a name, a type, and an initial value. You can see the preview of the client state parameters.



- Client scripts: Scripting automates aspects of the page, like event handling.

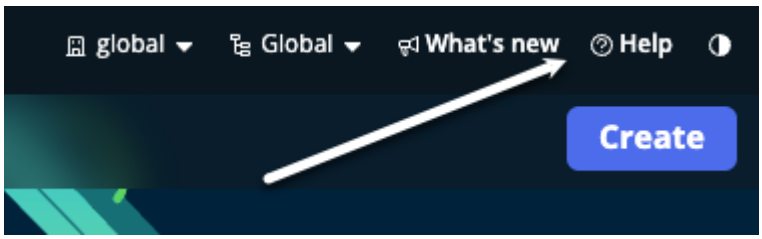


UI Builder Learning Center

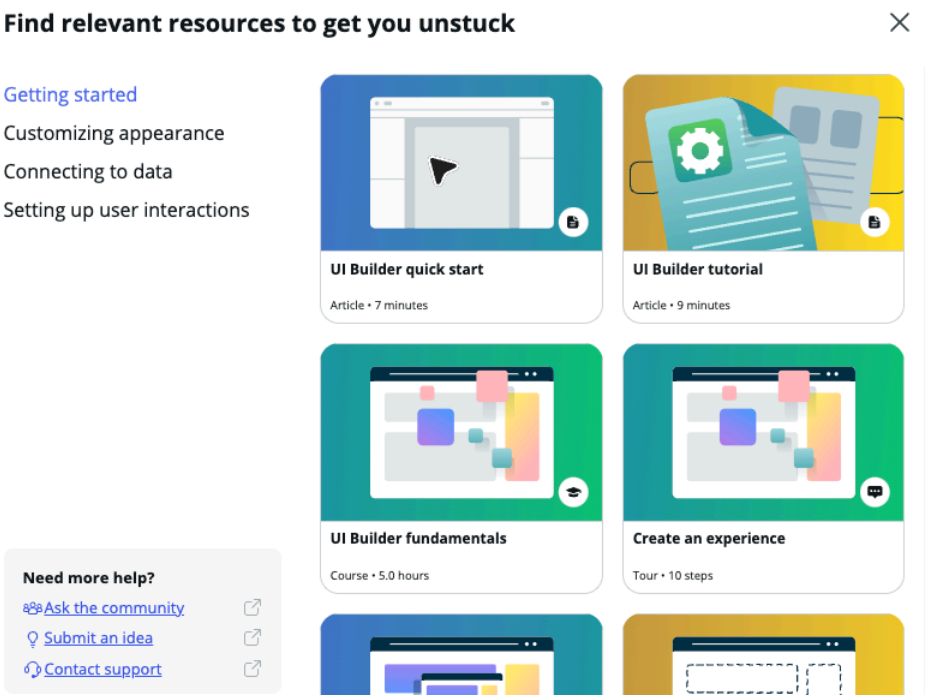
Explore the UI Builder Learning Center for articles, videos, courses, and guided tours to help you get started and build knowledge. The Learning Center is a one-stop shop for learning fundamental features and basic guidelines in UI Builder.

Open the Learning Center

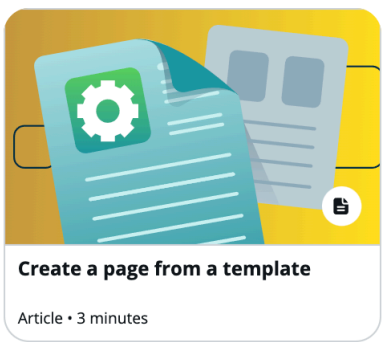
Navigate to Now Experience Framework > UI Builder and select **Help**.



The Learning Center opens in an overlay.



Available information is listed on tiles. Each tile contains a title, the information type, and the time to complete. For example, a tile linking to a product documentation article that takes three minutes to read.



Select a tile to open a resource. Resources are in the following locations:

- Articles: https://servicenow.com/docs
- Videos: created by ServiceNow and available on public, online video platforms

- Courses: https://learning.servicenow.com
- Tours: embedded in your ServiceNow instance

Guided tours in UI Builder

Learn about UI Builder guided tours, including how to use them to build your knowledge and practice using UI Builder.

What guided tours are

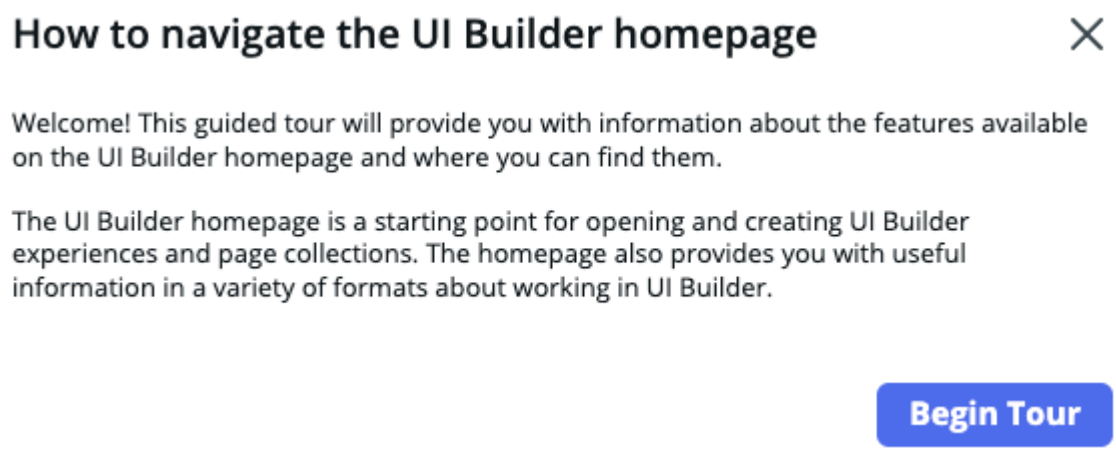
Guided tours are part of the default ServiceNow platform. There are guided tours available to take and you can use the Guided Tours application to create custom tours yourself in your own instances.

Tours contain interactive steps to help train and teach users. Some tours show various features in the user interface, such as an overview of a homepage. Other tours help complete a task, such as creating an experience or previewing an experience page. For more information about what guided tours are and how to create them, see Exploring Guided Tours.

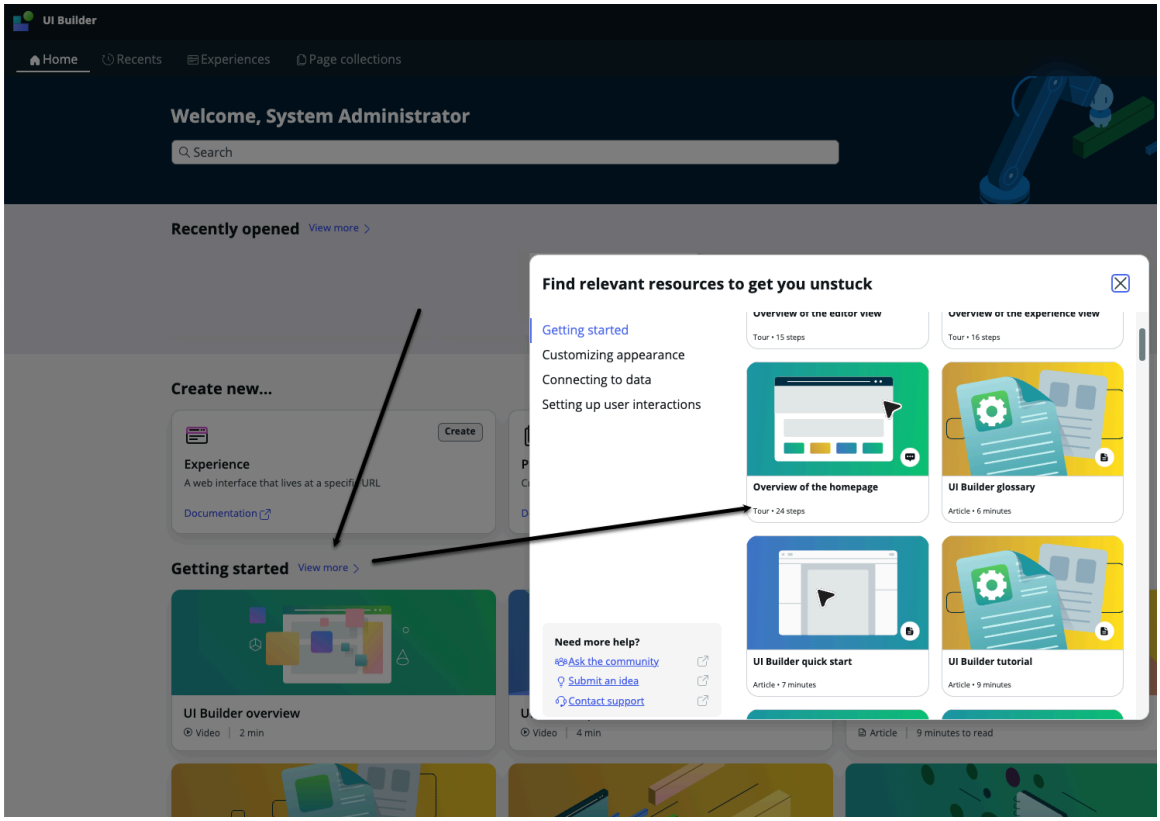
How to access and use UI Builder guided tours

Guided tours about UI Builder may launch automatically. You can also start them manually from the **Getting started** section of the UI Builder homepage or the UI Builder Help panel.

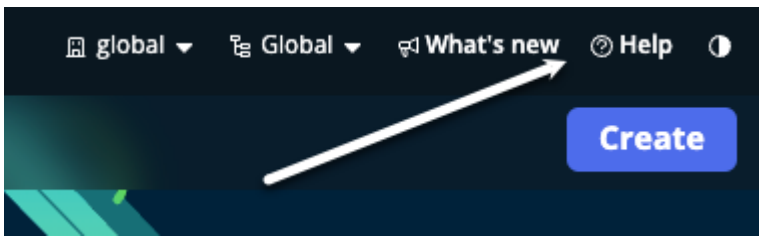
When a guided tour starts automatically, an introduction is displayed. Select **Begin Tour**.



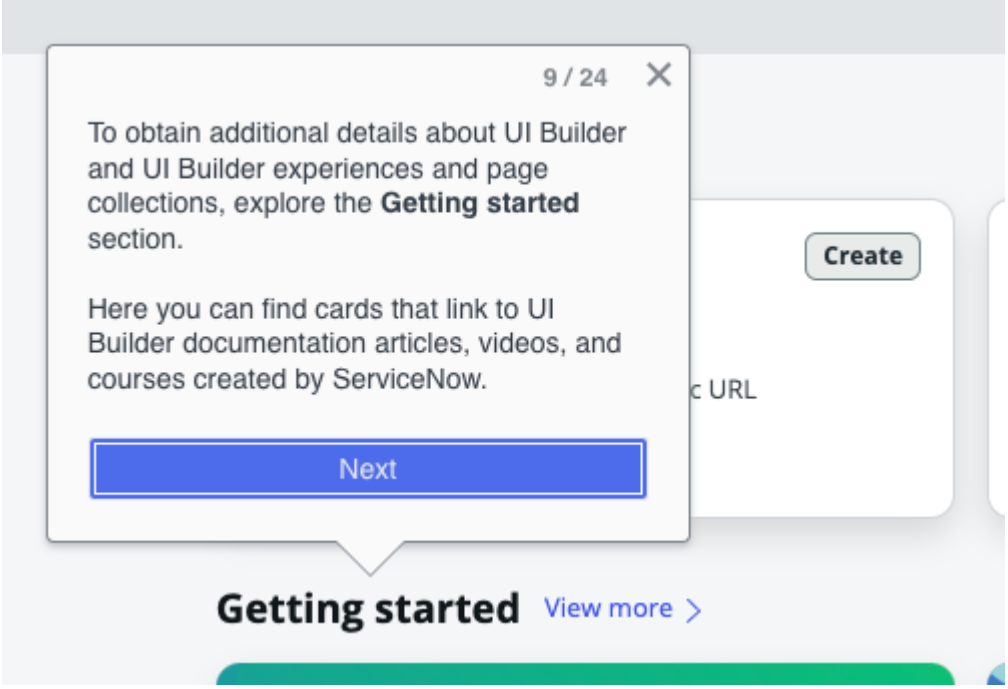
To begin a tour manually from the homepage, navigate to Now Experience Framework > UI Builder and select a tour in the **Getting started** section. If no guided tours appear by default in the section, select **View more** and then select a tour on the **Find relevant resources to get you unstuck** pop-up.



Alternatively, select **Help** and then select a tour on the **Find relevant resources to get you unstuck** pop-up.

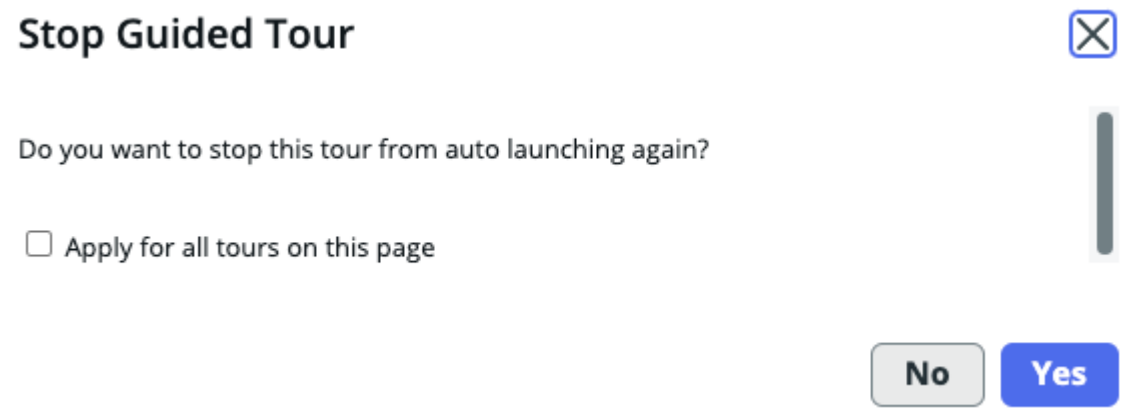


After a tour starts automatically or manually, you're led through a series of steps. Read the text for each step and perform the required actions or select **Next** to continue the tour.



To stop a tour at any time, select the **X**.

Select **Yes** or **No** to stop the tour from auto-launching again. You can also apply your decision to all tours that start on the same page.

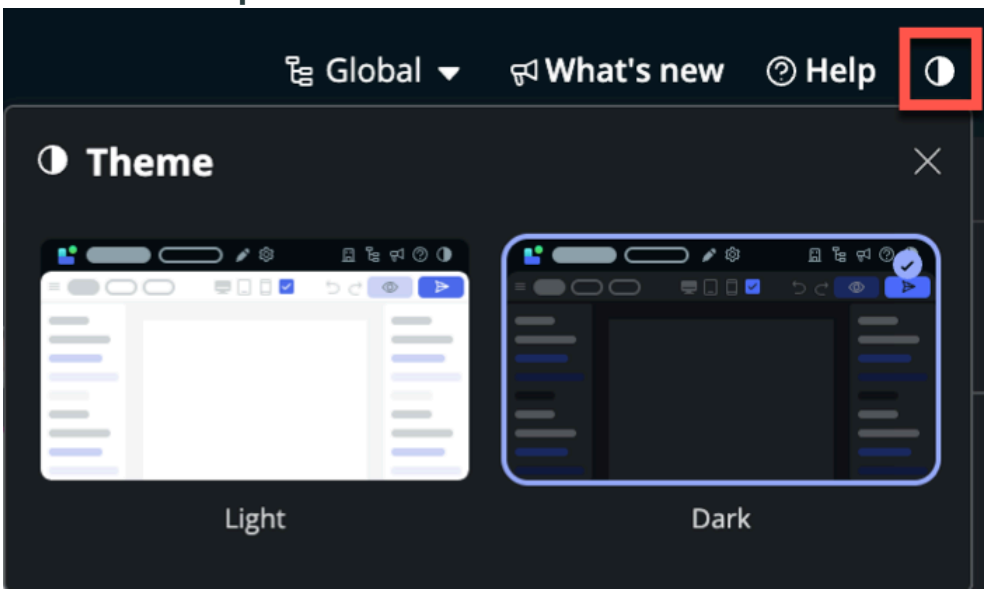


Dark theme in UI Builder

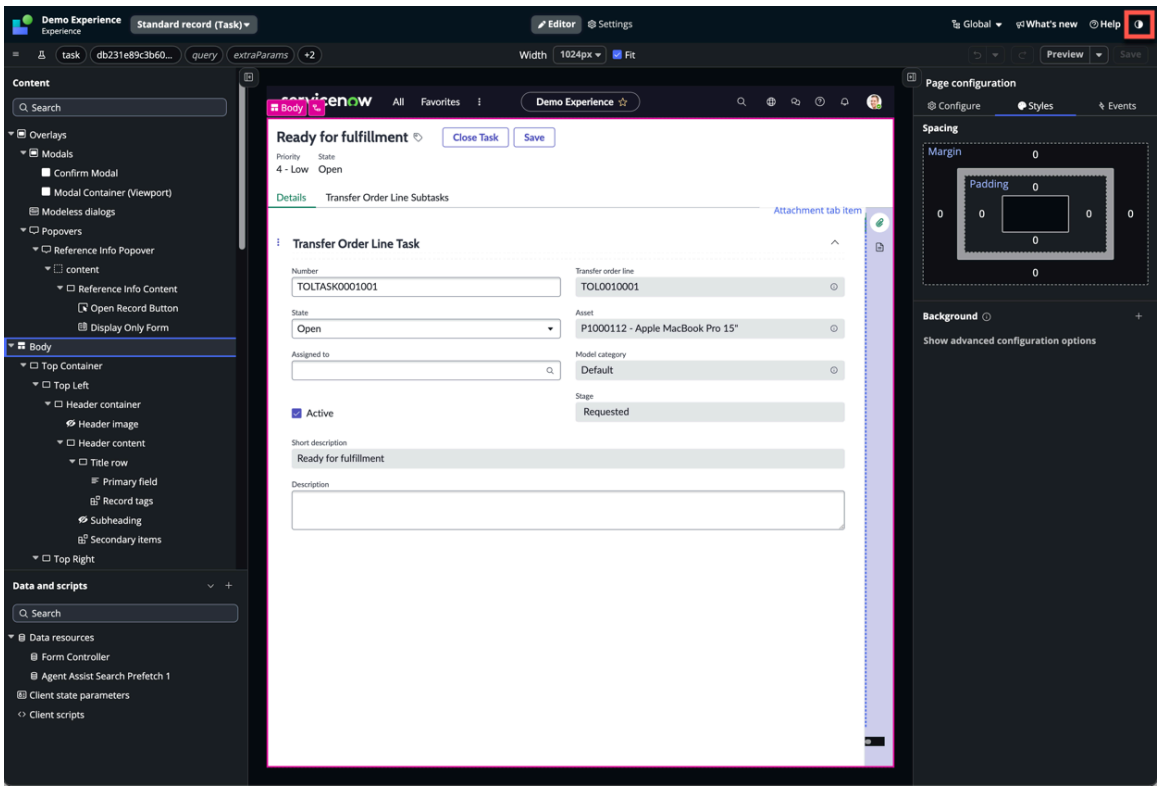
Learn how to switch to the dark theme in UI Builder.

UI Builder includes a dark theme option, offering an eye-friendly alternative to the default light theme. You can enable dark theme by selecting the Choose theme icon ( ) in the banner.

UI Builder theme options



The Dark theme is instantly applied to the UI Builder interface.



Note:
The UI Builder stage uses the theme set in your Platform theme settings. For example, the stage might remain white even when the rest of the interface is dark. To enable UI16 pages to render with the dark theme on the stage, select the **Dark** option in the theme menu of your user preferences. For more information on navigating to the theme menu, see Open the theme menu.

Learn UI Builder by example

Exploring common use cases can be a great way to learn how to use UI Builder. Consider performing each of the tasks in this section in the order presented.

The following use cases are described here.

Create a demo experience to explore UI Builder

An experience in UI Builder is a collection of web pages for users to interact with an application. The experience includes routes, page variants, and the audience and conditions required for each variant, as well as experience settings.

Before you begin

Role required: ui_builder_admin

About this task

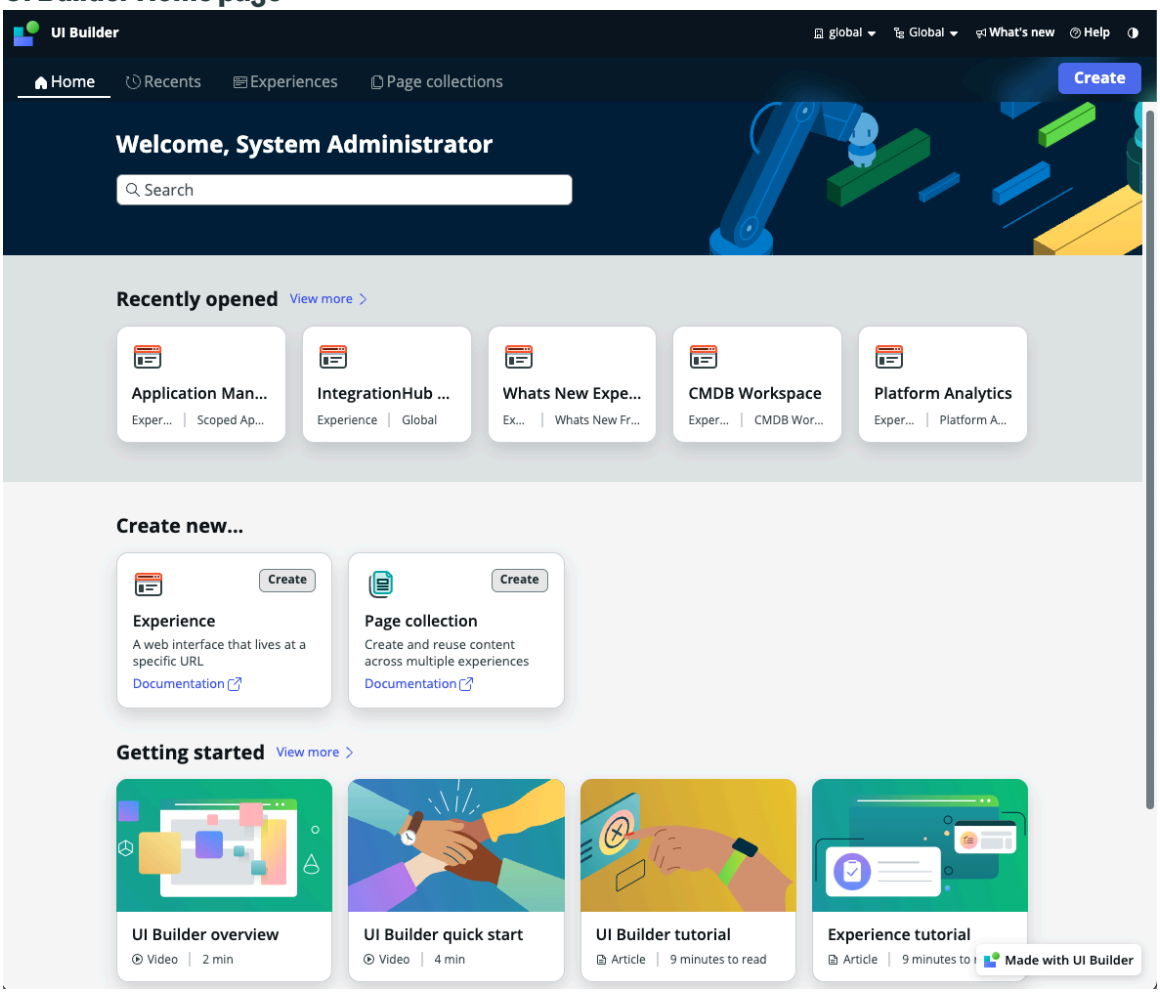
The UI Builder experience view is a central place to view and understand the structure and details of an experience. Use the experience view to see the structure and hierarchy of your experience. You can access UI Builder experience view by selecting an existing experience from the UI Builder home screen or by creating a demo experience, as described here.

Note:
UI Builder is not yet capable of building or configuring ServiceNow base system service portals, such as the Employee Center. For service portals, continue to use the Service Portal Designer.

Procedure

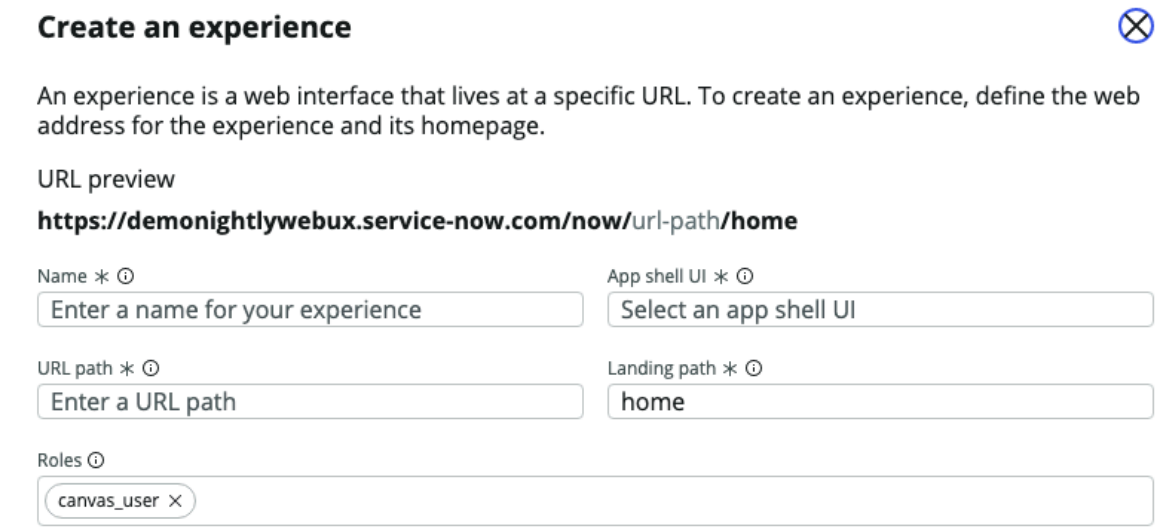
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.

UI Builder Home page



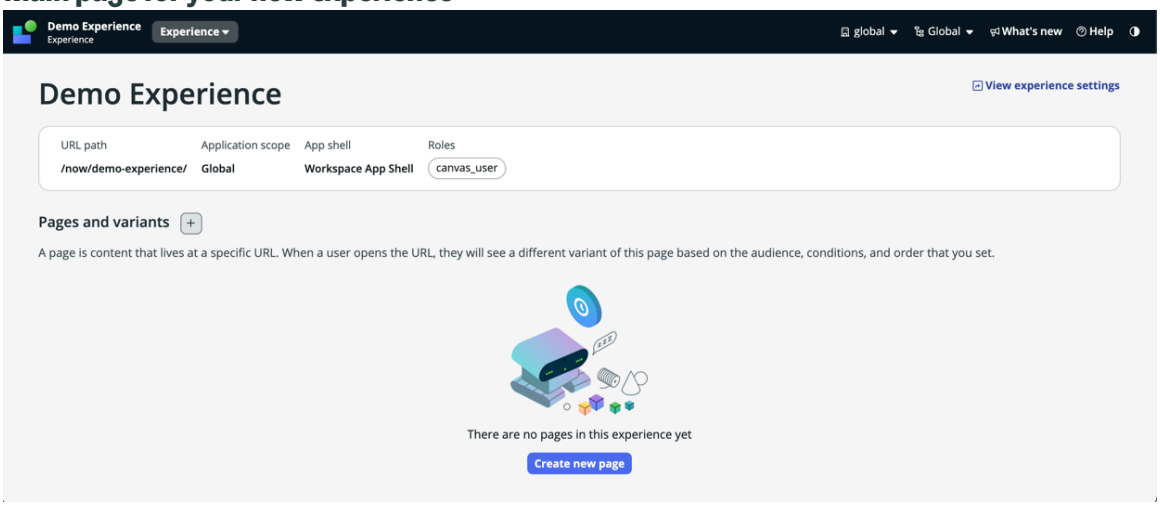
2. From the UI Builder Home page, select **Create**.
3. Select **Experience**

Create an experience dialog box



4. Enter a **Name**; for example, **Demo Experience**.
5. Select an **App shell UI**; for example, **Workspace App Shell**.
6. Select **Create**.
7. Select **Open experience** to view the main page for your experience.

Main page for your new experience



Note:
In addition to the name of your new experience, the screen includes the URL, application scope, admin panel, and the roles of users who can view the experience. Any pages or variants you create for the experience appear in the **Pages and variants** section.

What to do next

Select the **Next topic** link to learn how to create a blank page for this experience.

Create a blank page

After creating a demo experience, you can create a blank page that you can then build on to define what the experience provides to your users.

This video shows you how to perform the following procedure. This video shows you how to create a blank page in a UI Builder experience.

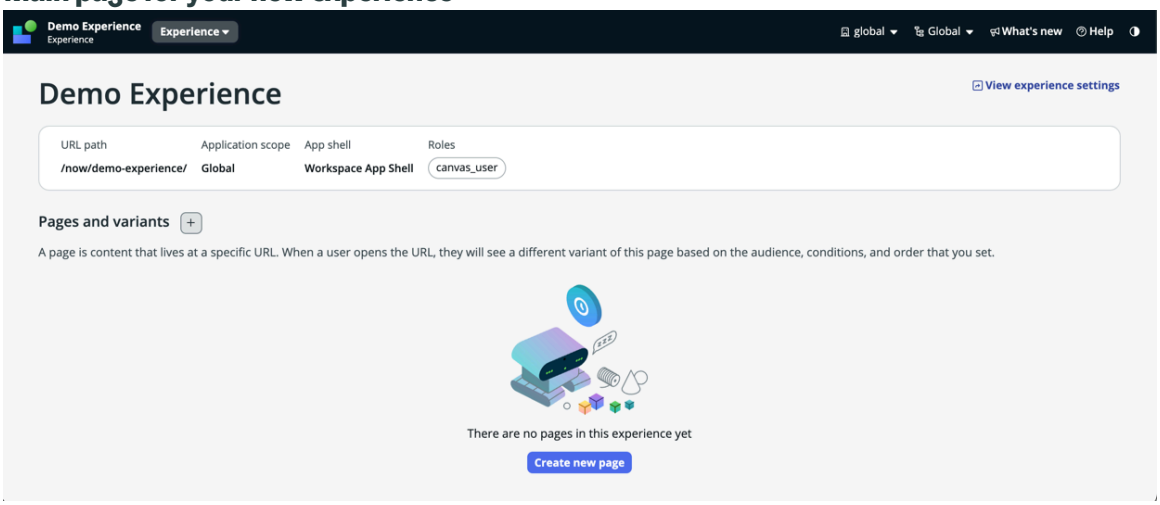
Before you begin

Role required: ui_builder_admin

Procedure

1. If you aren't already viewing the experience you created in the previous procedure, navigate to **All** > **Now Experience Framework** > **UI Builder**, and select the experience you created.

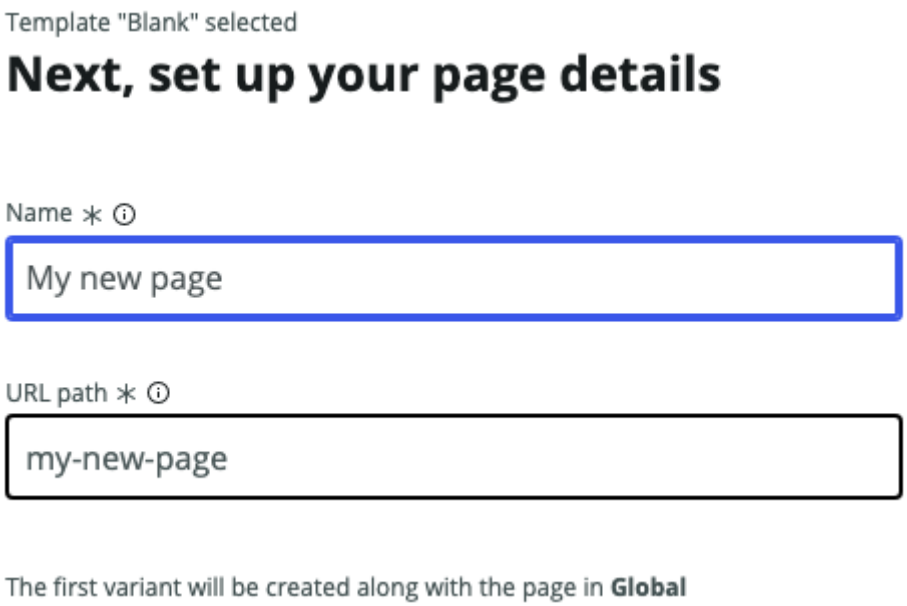
Main page for your new experience



2. From the main page for your experience, either select **Create new page** or select the plus (**+**) sign next to **Pages and variants**.
3. On the **First, select a template** screen, select **Create from scratch instead**.

   There are templates you can use, but we are going to create a page from scratch.

Page details

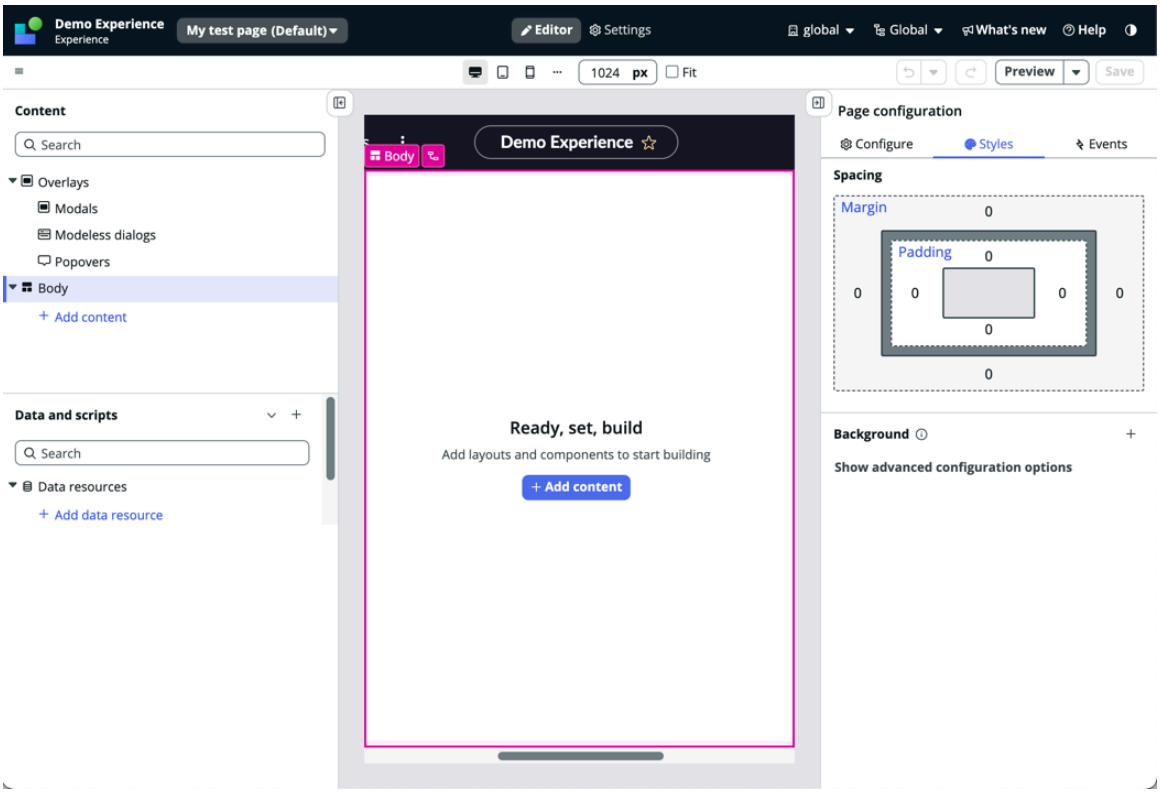


4. In the **Name** field, type **My test page**, and select **Continue**.

Note:
The **URL path** shows the URL that users can navigate to access the page.

5. Select **Looks good** on the next screen.
6. On the **Tell us about your variant** screen, do not make any changes.
7. Select **Continue**.
8. Select the **Build responsive** option (default) for greater control of how the page appears at different screen widths.
9. Select **Create**.
   The visual editor opens. Here, you can begin editing your new page.

UI Builder visual editor



What to do next

Select the **Next topic** link to learn how to create a button that opens a modal.

Create a record page using a template

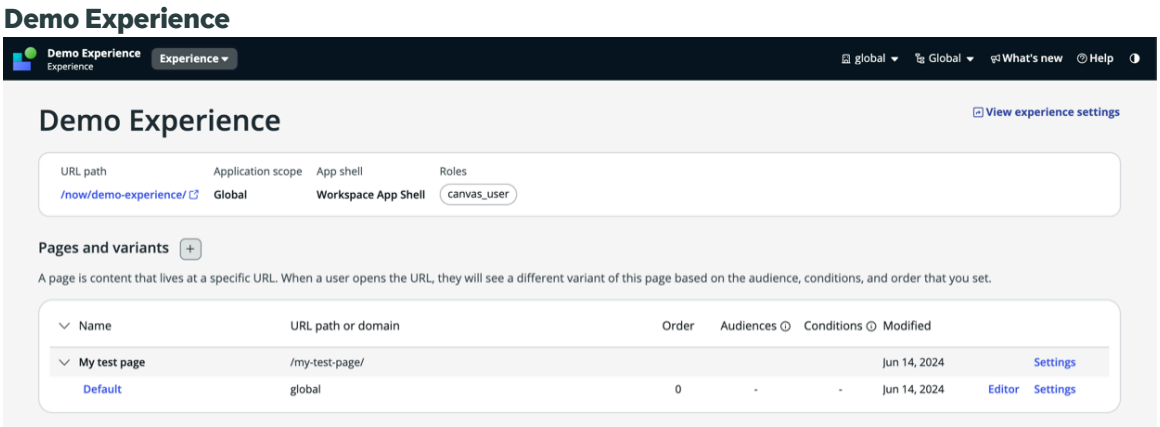
After you've created your demo experience, you can create a record page from a template. A record page shows data from a table.

Before you begin

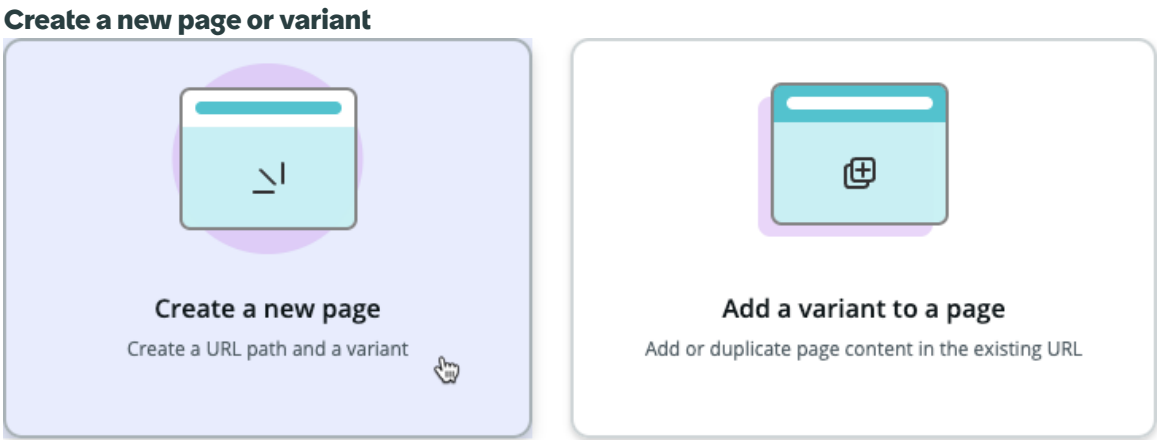
Role required: ui_builder_admin

Procedure

1. Open the main page for your demo experience.

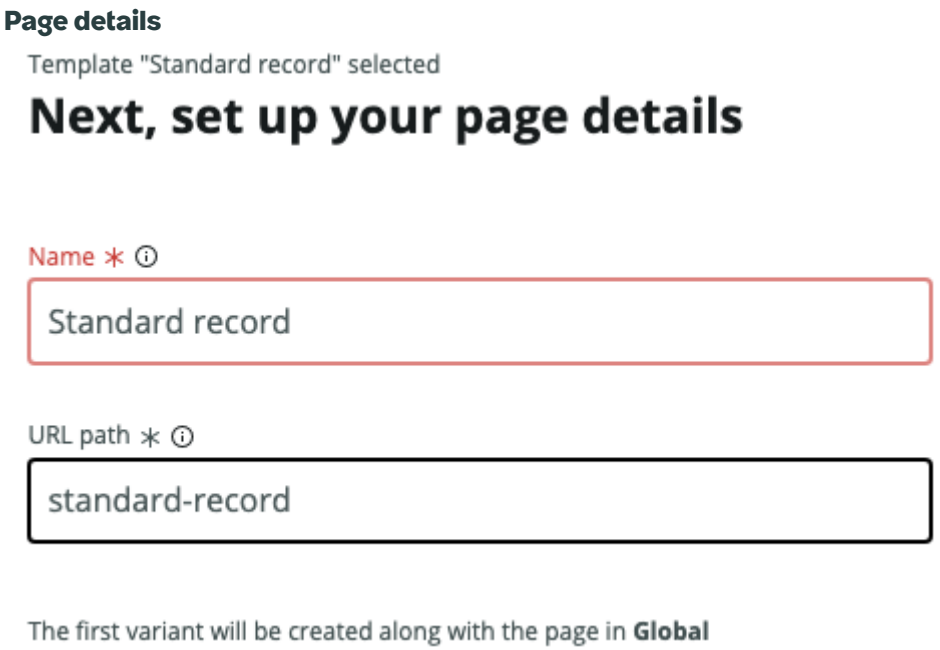


2. From the main page for your experience, select the plus (**+**) sign next to **Pages and variants**.

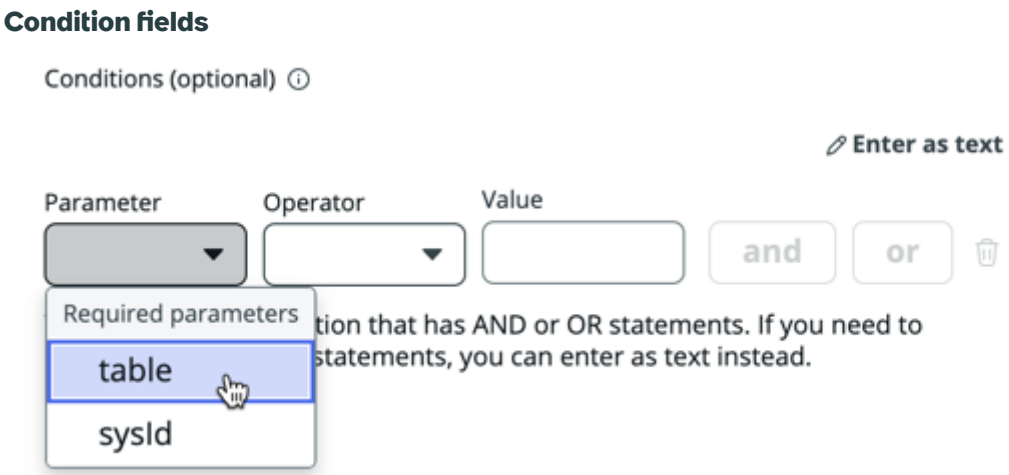


3. Select **Create a new page** to create a page that resides at a different URL.
4. On the **First, select a template** screen, locate the **Standard record** template and select **Use template**.

> **Note:**
> Optionally, you could select **Learn more** to read about the template before selecting it.

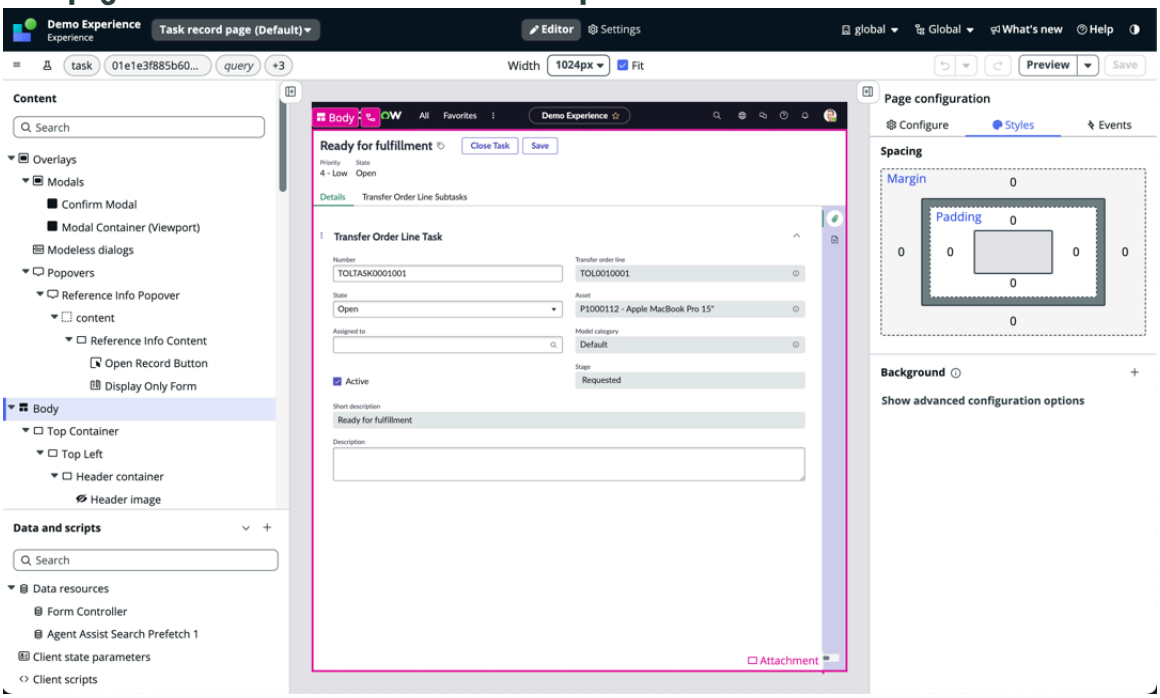


5. In the **Name** field of the Page details screen, type `Task record page`, and select **Continue**.
6. Select **Looks good** on the next screen.
7. On the **Tell us about your variant** screen, enter a condition to determine when the page is visible.
   a. In the **Parameter** field, select **table**.
   b. In the **Operator** field, select **is**.
   c. In the **Value** field, enter `task`.



   This page is visible to users accessing a record from the Task table.
8. Select **Create**.
   The visual editor opens. Here, you can begin editing your new page.

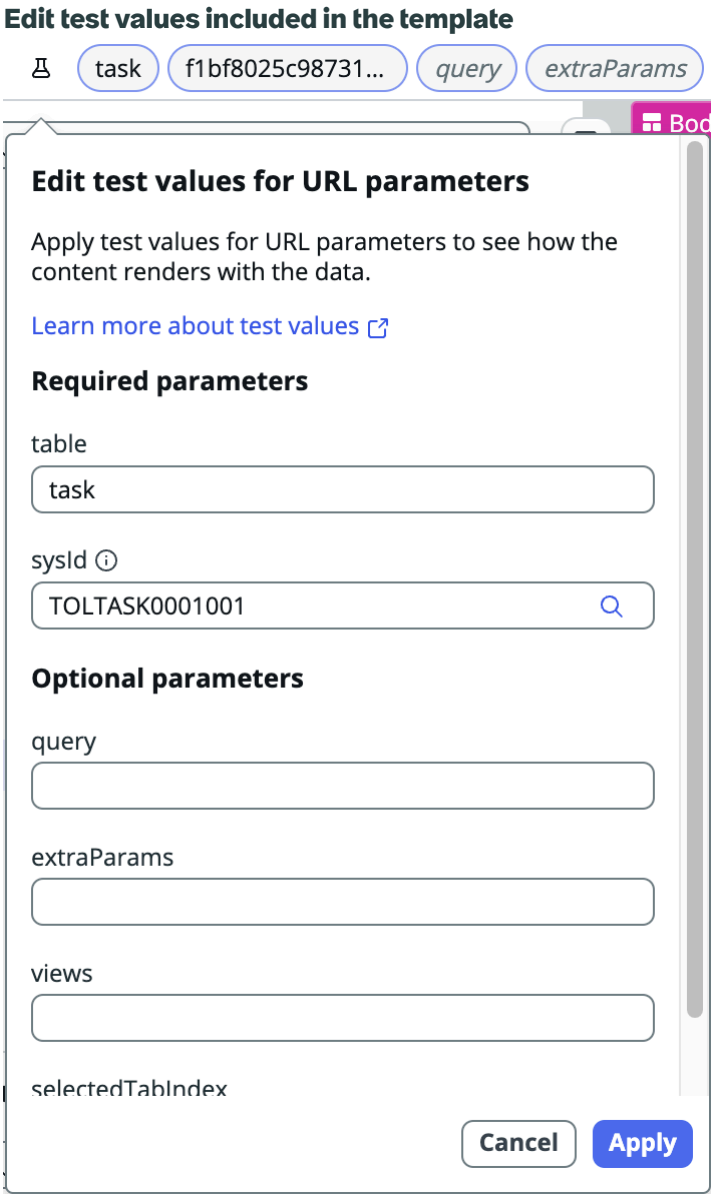
New page created from Standard Record template



Note:

The new page includes LOTS of layout, preconfigured components, data, and modals. Additionally, it includes test values you can modify for your own particular use cases. Using templates can be a significant time-saver when creating new pages.

9. You can select the area above the Page content pane to view information about test values included in the template.



What to do next

Select the **Next topic** link to learn how to define audiences who can view your pages in UI Builder.

Create a button that opens a modal

After you've created your demo experience and added a blank page, you can edit the page variant as needed. For the sake of this demo, you can create a button and a modal, and configure the button to open the modal.

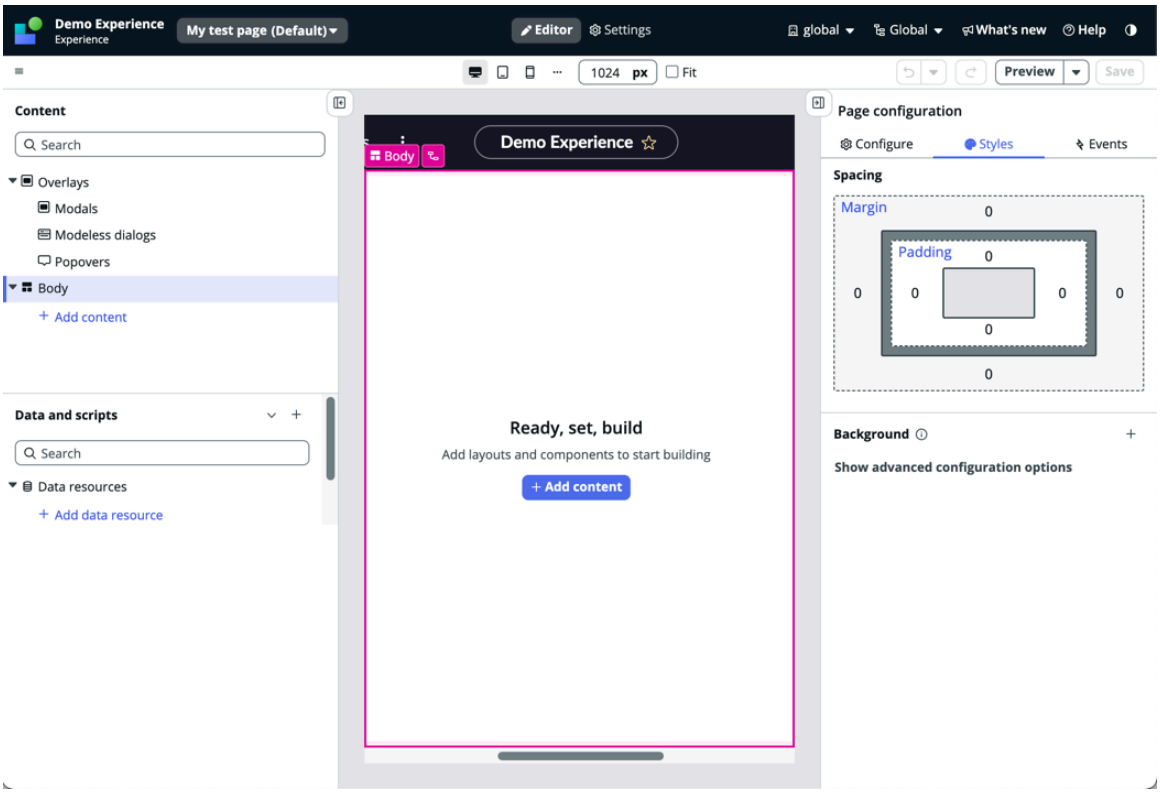
Before you begin

Role required: ui_builder_admin

Procedure

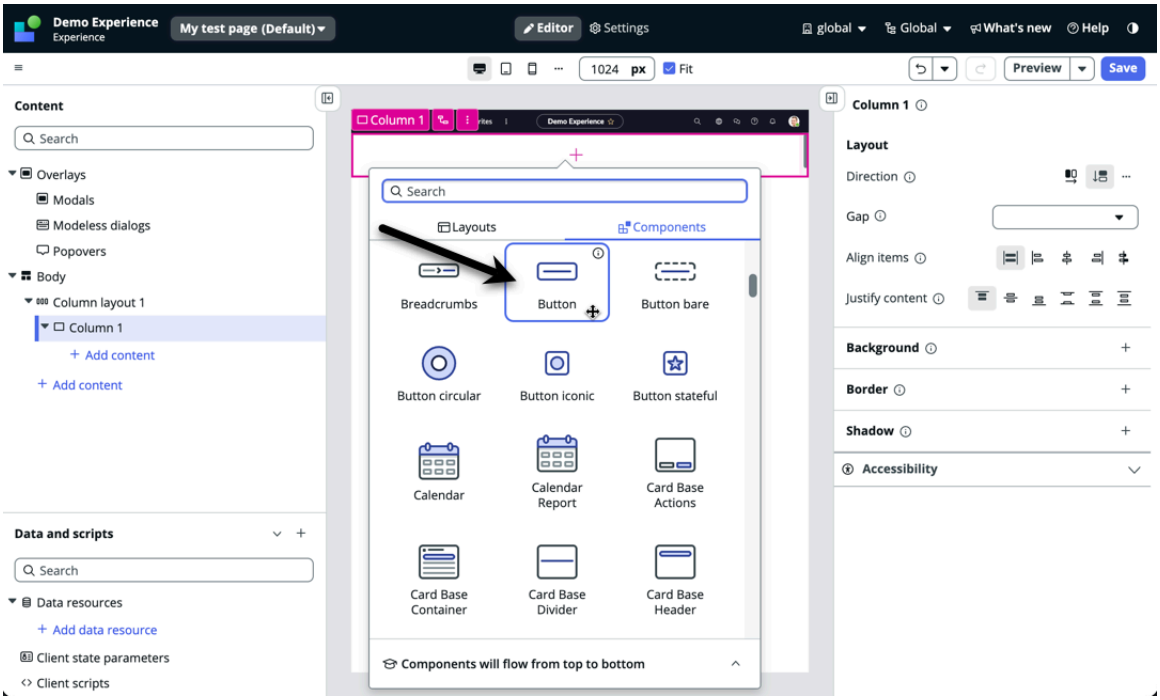
1. Open the UI Builder page for your demo experience.

UI Builder visual editor



2. Click the **+ Add content** button on the stage to open the toolbox.
3. Select a **Single column** layout.
4. Next, click the **+** icon in the column to open the toolbox.
5. Select the **Button** component to add it to the stage.

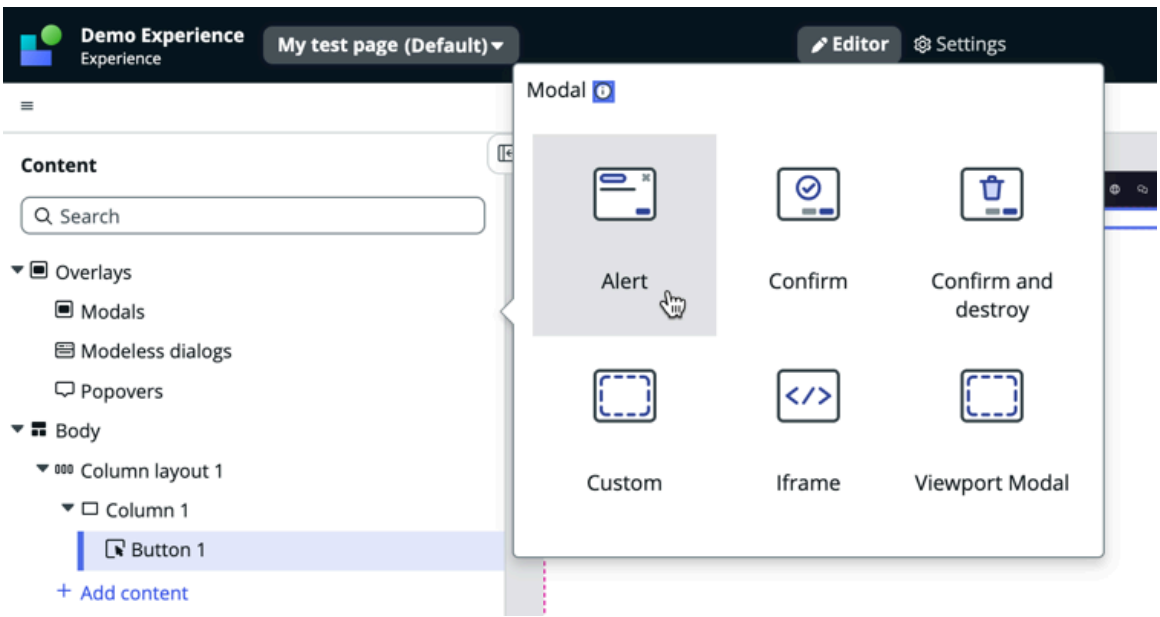
Add a button



Note:
When you have selected the component, the Page configuration pane includes some presets you can use to automatically configure components on compatible pages. For the sake of this exercise, however, you will be configuring the component manually. For more information on presets, see Customize UI Builder pages using components.

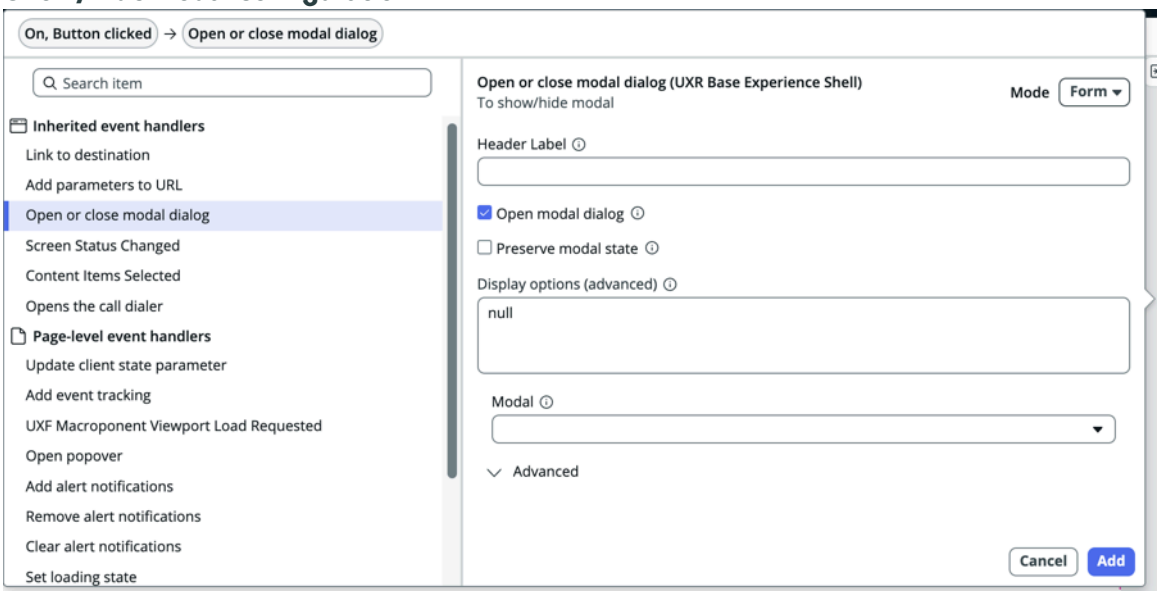
6. In the Page configuration pane, select **Configure the component manually**.
7. In the Page content pane, select **Button 1** and, in the Configuration pane, change the button label to **Open modal**.
8. Select **Save**.
9. In the Page content pane, click the plus icon next to **Modals** and select an **Alert** modal.

Add an Alert modal



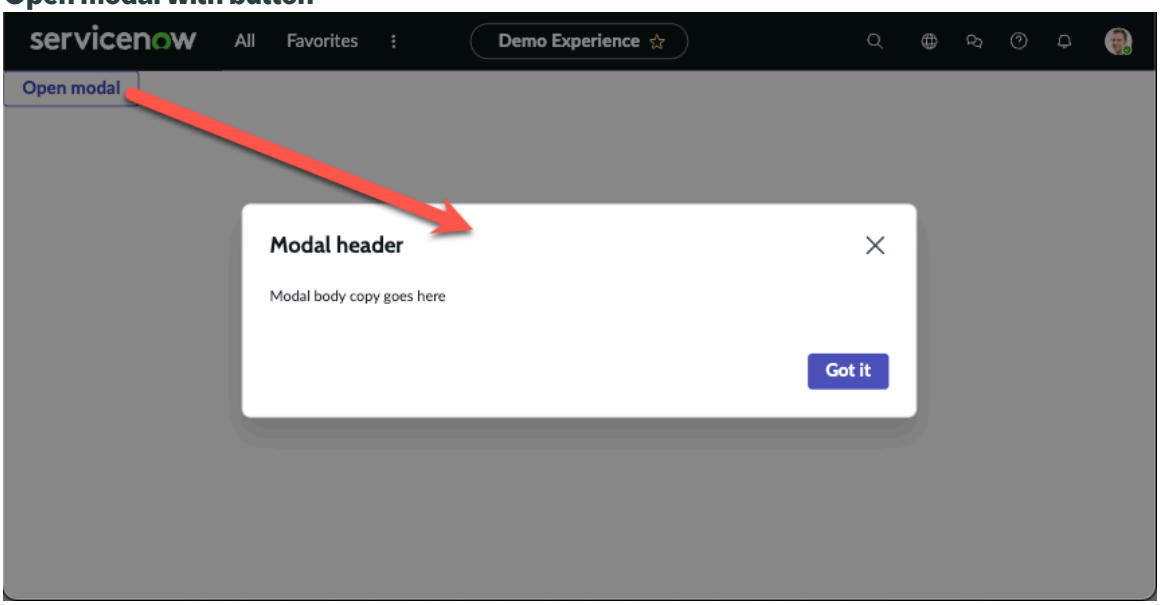
10. Select **Save**.
11. In the Page contents pane, select **Button 1** and, in the Configuration pane, select the **Events** tab.
12. Select **+ Add event handler** and, under **Inherited event handlers**, select **Open or close modal dialog**.

Show/hide modal configuration



13. Activate the **Open modal dialog** and, select **Alert 1** in the **Modal** field, and select **Add**.
14. Select **Save**.
15. Select **Preview**.
16. When the preview opens, select **Open modal**.
    The modal you defined opens.

Open modal with button



17. Select **Got it** in the modal, and then select the browser back button to return to the experience main screen.

What to do next
Select the **Next topic** link to learn how to create a page using a template.

Define an audience for your variant

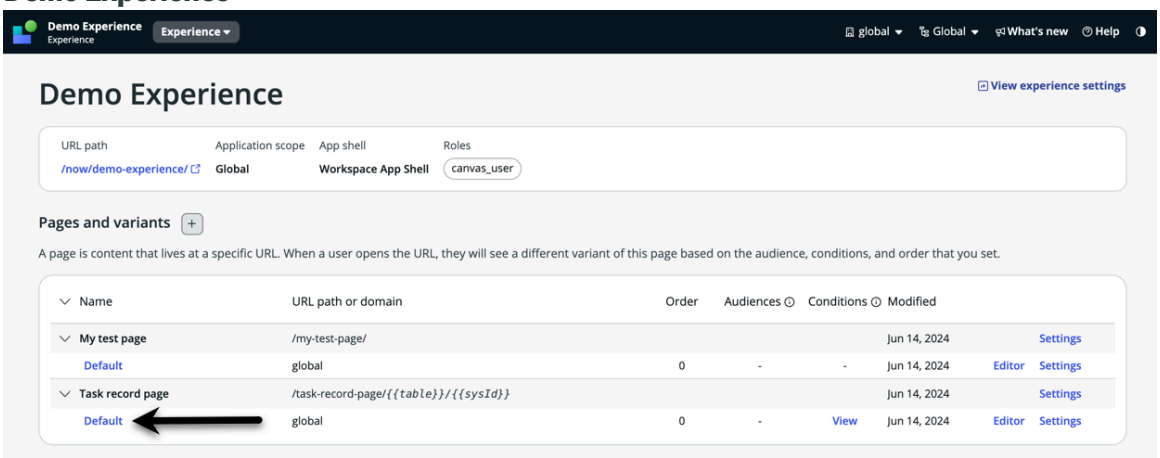
An audience represents a group of users in your organization. You can define who can access this page variant by adding one or more predefined audiences.

Before you begin
Role required: ui_builder_admin

About this task
In the previous procedure, you created a page variant that can be viewed by anyone accessing a record from the Task table. For this procedure, you will duplicate the variant and configure it so that it is accessible only to audiences with the Admin role.

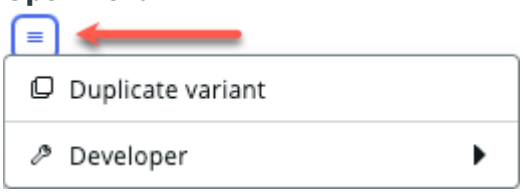
Procedure
1. Open the main page for your demo experience.
2. Select the experience you created, and select **Default**.

Demo Experience



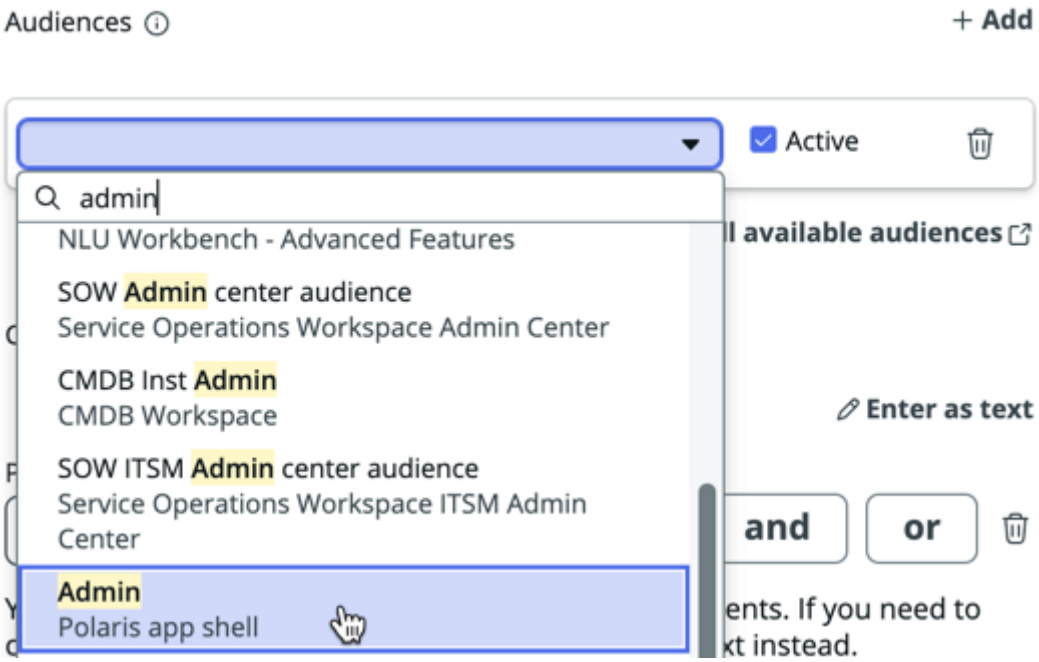
3. Select the **Open menu** icon.

Open menu



4. Select **Duplicate variant**.
5. On the **Tell us about your variant** screen, enter **Admin only** in the **Name** field.
6. Select **+ Add** next to **Audiences**, and select the **Admin** audience.

Select the Admin audience



7. Select **Create**.
   The variant is duplicated and only users with the Admin role can view this page.

What to do next

Select the **Next topic** link to learn how to apply conditions to the variant so that the variant is visible only when the defined conditions are met.

Define conditions for your variant

You can define conditions to determine when a page variant is shown. The conditions are based on setting an order, and declaring the criteria that must be met for the page variant to display.

This video shows you how to perform the following procedure. This video shows you how to define conditions for your page variant in a UI Builder experience.

Before you begin

Role required: ui_builder_admin

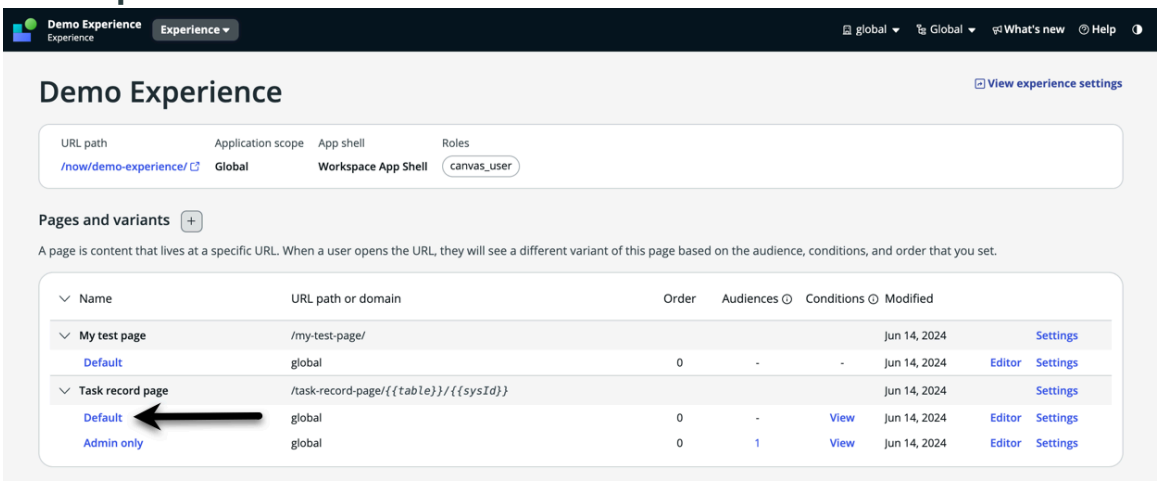
About this task

If you have multiple page variants that all have the same conditions, the variants go by the order setting.

Procedure

1. Open the main page for your demo experience.
2. Select the experience you created, and select **Default**.

Demo Experience



3. On the Default variant under the **All users record page**, select the **Menu** icon (⋮), and select **Duplicate variant**.
4. On the **Tell us about your variant** screen, enter **Incident record page** in the **Name** field.
5. Since the intent of this task is to restrict this page to users who are opening a record in the Incident table, enter the following under Conditions:

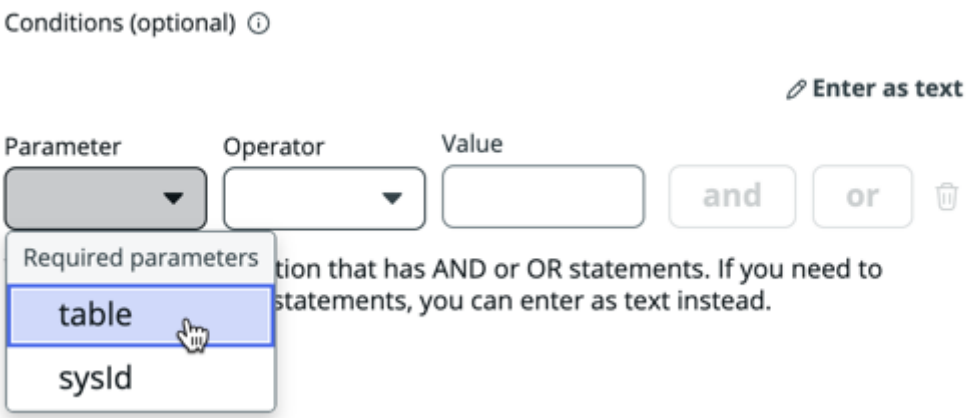
   a. In the **Parameter** field, select **table**.

   b. In the **Operator** field, select **is**.

   c. In the **Value** field, replace the text with `incident`.

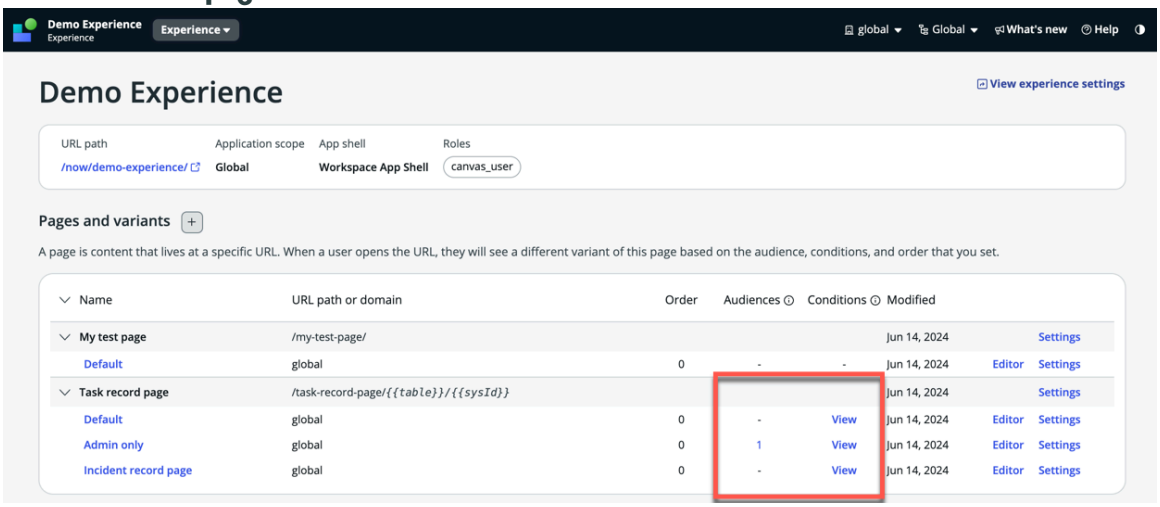
   When you created this page earlier in the series, you set this condition to the Task table. Now, you are configuring the page to display only to users accessing a record from the Incident table.

Condition fields



6. Select **Create**.

Incident record page



Note:
You can view the structure of your pages and variants in this experience. The **Conditions** column shows a **View** link for each page. You can select **View** to see the condition for that variant (that is, table=incident). The **Audiences** column for the Admin only variant shows a 1. You can select the **1** to view the role required to view that variant.

Customize forms within a form component

Customize your form components by accessing Form Builder in UI Builder.

Before you begin

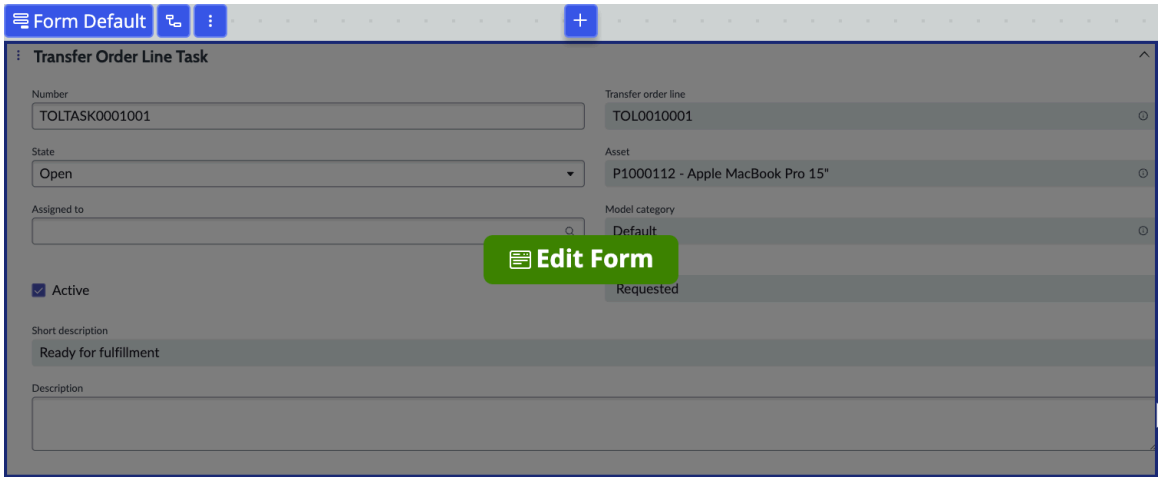
Role required: admin

About this task

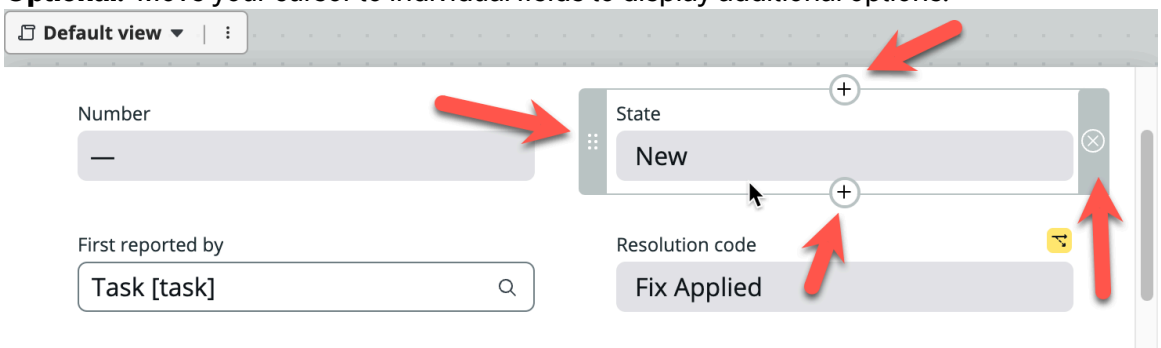
You can edit form components without leaving UI Builder. Access Form Builder from within a form component on the UI Builder stage. After you save the form, your edits remain, and anyone who views the form sees the modified version. Changes made in Form Builder are reflected on the stage.

Procedure

1. Add a form component to your page.
   a. In the content tree, select **+ Add content**.
   b. Select **Form**.
2. On the stage, move your cursor to the form component to display the **Edit Form** button.



3. Select the **Edit Form** button.
   Form Builder opens in a full-screen modal.
4. **Optional:** Move your cursor to individual fields to display additional options.



5. **Optional:** Insert fields by moving your cursor to the **+** button, which displays the **+ Add** button, then selecting **+ Add**.
   You can also drag additional fields onto the form from the **Fields** column.

   Example
   For example, you can add the **Due date** field below the **Configuration item** field, then select **Save** and **Preview** to save and preview the form.
6. **Optional:** Move a field by dragging the edge of a field to another place in the form and then selecting **Save**.

   Example
   For example, select the **State** field and place it below the **Duplicate of** field, then select **Save**.
7. **Optional:** Delete a field by selecting the **X** button on the field.
8. Select **Save**.
9. Close Form Builder to see your edits appear on the stage.

What to do next

You can enhance the functionality of a custom form by configuring additional features. For more information, see Forms in Table Builder.

Learn UI Builder using other ServiceNow resources

Learn more about UI Builder using resources outside of the product.

Learn more about UI Builder	Additional ServiceNow resources
UI Builder is a web user interface builder. Use UI Builder to build pages for CSM Configurable Workspace, App Engine Studio generated workspaces and portals, or custom web experiences using Next Experience Components and custom web components.	ServiceNow Dev Program YouTube channel
	Community YouTube channel
	ServiceNow Community site
	Developer Advocate blog
	ServiceNow Developer site
	Components documentation

Learn about audiences

Learn how to apply the correct audiences to your UI Builder pages.

Audiences

Understanding your audiences in UI Builder is an important part of creating pages. The audience defines who can see your pages. You create pages for targeted experiences tailored to audiences and roles such as agents and managers. For example, you may want to create a page for agents to solve issues for your employees. You want to ensure that only people who have the agent role can see the page.

Audiences are generally made of allow/deny lists based on role and domain. They can also target the following:

- Role
- Group
- User

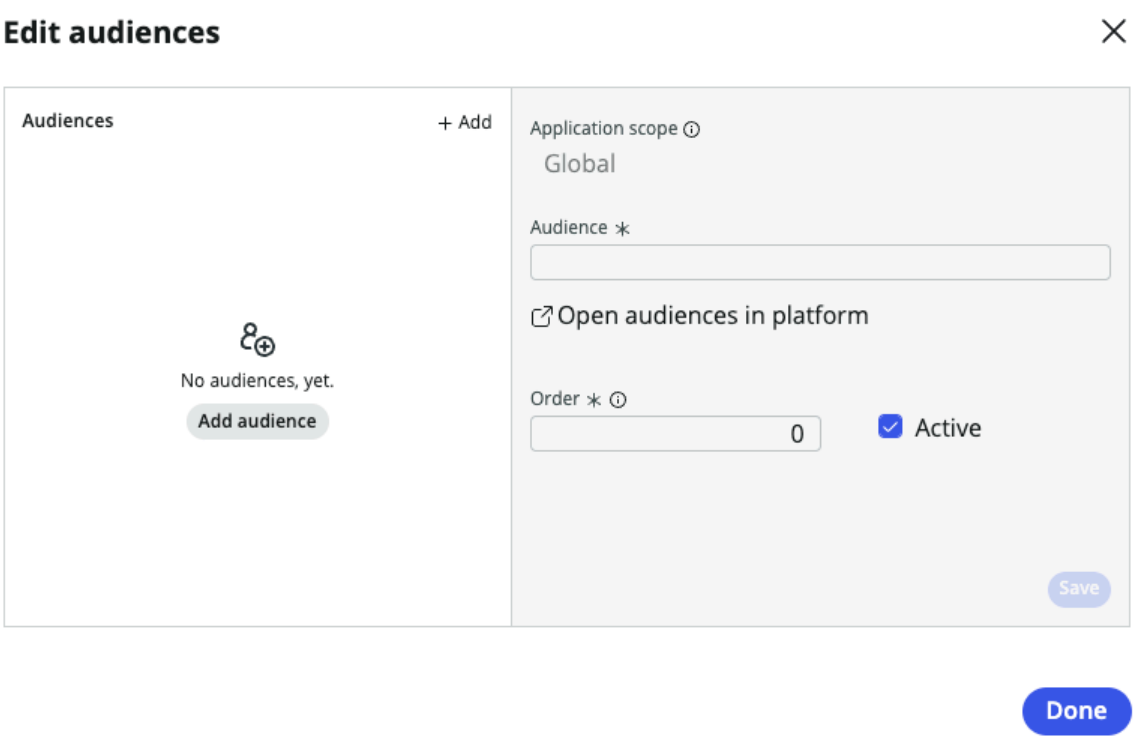
- Company
- Department
- Location
- Script

You can set audiences to fit a specific role based on one or more criteria. For example, you could create an audience for an ITSM user in Europe who is not a manager.

The *glide.ux.user_criteria_enabled* property needs to be set to **true** to configure access for users based on role, department, group, location, or company. See Enable the user criteria property, for more information.

Set an audience in UI Builder when you create your page or page variant. You can also set the priority of the audience record. The lower the number the higher the priority. Ensure the **Active** check box is selected to make this page active.

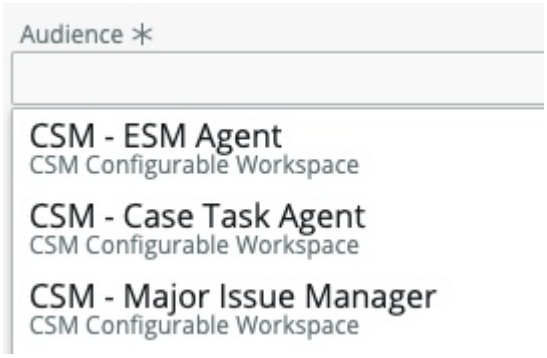
Add audiences modal



If you do not see an audience that you need, you can click **Open audiences in platform** to define an audience in the ServiceNow AI Platform®.

- Choosing an audience from the list in the **Audience** field.

Audience list



- Selecting **Open audiences in platform** to edit or create an audience record in the ServiceNow AI Platform®.
- If you have multiple audiences, define the **Order** for each audience record. The order defines the importance of each audience record. For example, a CSM Manager could have a higher priority than a CSM Consumer Agent. To give higher priority to an audience, enter a lower number.
- Ensuring the **Active** check box is selected to make this page active.

Learn about security and roles

Set up the security and roles for your UI Builder instance.

Security and roles in UI Builder are controlled through your application scope, domain separation, and protection policy settings.

Roles

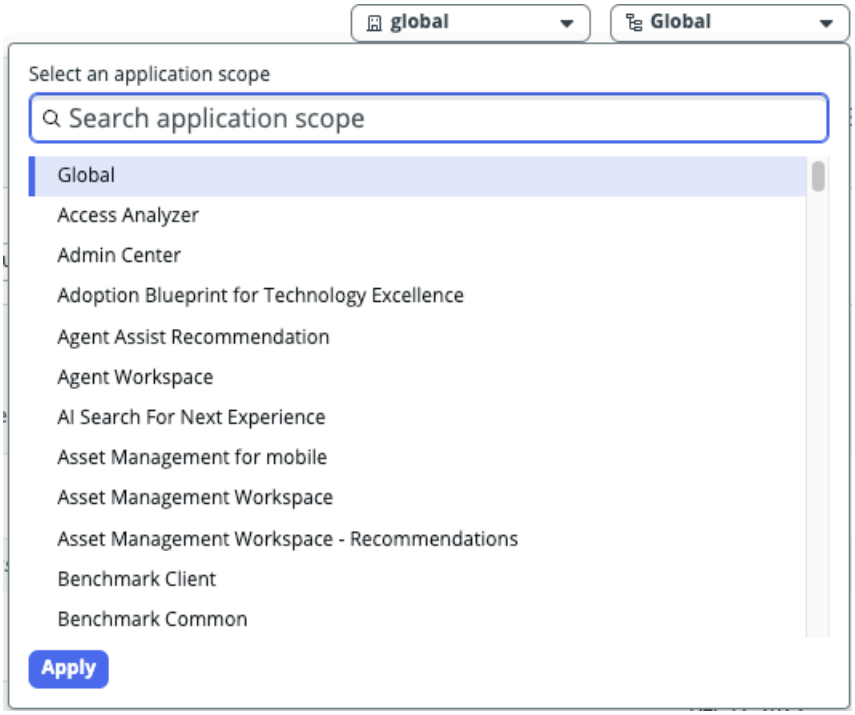
Roles control access to features and capabilities in UI Builder. The admin role provides access to all features and capabilities. After access has been granted to a role, all the groups or users assigned to the role are granted the access. Roles can contain other roles, and any access granted to a role is granted to any role that contains it. For more information, see Managing roles.

Application scope

Application scoping protects applications by identifying and restricting access to application files and data. Administrators can specify what parts of an application are accessible to other applications, which helps to protect data and application files. In UI Builder, System administrators and developers set application scope when creating a page. Note, you cannot change the application scope after creating the page, so choose your scope carefully.

Changing the scope in UI Builder also changes the scope in the ServiceNow AI Platform®, and vice versa.

When creating a page, it's important for admins and developers to be aware of the scope they are in for the workspace or portal experience. Choose the correct application scope for your experience. The scope picker is to the right of the URL field. The scope defaults to the scope that the user is currently in within the ServiceNow AI Platform®.



If you change to a different scope while in a page, you're notified that you are in a different application scope from the one the page was created in. For more information, see Application scope.

Delegated developers for UI Builder

If you have the application-specific admin role or the system-level admin role, you can delegate application development in ServiceNow® Studio to designated developers at the UI Builder application level.

Protection policy

A protection policy prevents anyone from modifying and/or copying an application file or its related record. A protection policy is typically used when the author of an application is different than the company that uses the application. UI Builder notifies you if you try to modify a protected record. For more information, see Application file protection policy.

Learn about domain separation

Domain separation is supported for UI Builder . Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Support level: Standard

- Includes all aspects of **Basic** level support.
- Application properties are domain-aware as needed.
- Business logic: The service provider (SP) creates or modifies processes per customer. The use cases reflect proper use of the application by multiple SP customers in a single instance.
- The instance owner must configure the minimum viable product (MVP) business logic and data parameters per tenant as expected for the specific application.

Sample use case: An admin must be able to make comments required when a record closes for one tenant, but not for another.

For more information on support levels, see Application support for domain separation.

Overview

UI Builder is a web user interface builder. UI Builder enables developers to build new pages or customize existing pages for web-based workspace experiences using Next Experience Components and custom web components. In addition, UI Builder supports Domain Separation, which is the ServiceNow® instance-wide multi-tenant architecture.

Enable developers or dashboard builders in domain-separated environments to safely create UI application screens or dashboards while in the same browser window. Domain Separation in UI Builder works similarly to application scope to help administrators safely create or edit in a multi-tenant environment.

It's important to understand a key principle to maintaining a stable, healthy, and scalable ServiceNow® instance, where Domain Separation is installed.The primary principle is standardization. Standardization means a common configuration that most the instance operates by. When an instance has hundreds or thousands of domains, managing them successfully requires rigorous governance. Domain-specific configurations should be used only if they are deemed necessary by the instance owners. Generally, most instances should follow the common instance configuration. Doing so provides a more uniform experience across the instance. It also lets instance owners minimize code sprawl that slows the adoption of new ServiceNow® features included as part of release upgrades.

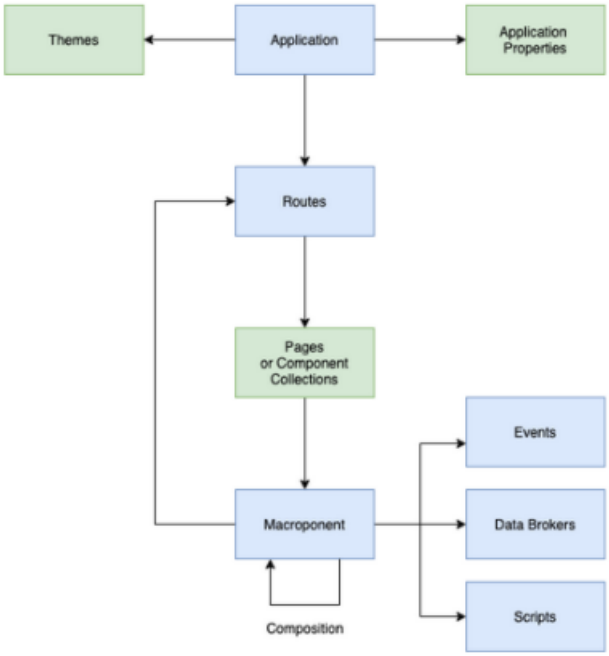
How domain separation works in UI Builder

Domain separation in UI Builder works similarly to application scope to help administrators safely create or edit in a multi-tenant environment.

UI Builder is compromised of a framework of interlocked components that you use to create web-based workspaces, dashboards, or portals. While the application supports domain separation, it does not mean every component or table is domain separated, which is important for instance owners to understand.

If the current domain does not match the domain of the variant or dashboard, the record is read-only. If a user has access to the domain, they can choose to switch their domain to the domain of the record. Alternatively, users can edit the record. Editing the record temporarily forces the user session into that record's domain. They can then make edits without fear of accidentally creating an override.

The following diagram shows what is (in green) and is not (in blue) domain separated in UI Builder.



Not shown in the diagram are viewports, declarative actions, and screen applicabilities, which are domain supported as process.

Data and Process/UI separation are important when considering domain separation architecture. UI Builder fully supports data and process/UI separation, and any data (records) displayed in the web-based workspace, dashboard, or portal experiences.

For example, a change request that belongs to the domain of Acme only shows for users who have access to the domain of Acme in an experience built using UI Builder. Conversely, if an application does not support data separation, its records won't be domain separated by the workspace or portal experience.

Process/UI separation tables that form the underpinning framework in UI Builder are process separated, and a sys_override column exists on those tables. For example, if a page is created in Global, any changes to the logic created and saved in a sub domain results in an override.

For items that are not domain separated, any change to the logic globally affects any page or dashboard that references its content. Understanding domain separation is critical when interacting with these elements.

Domain Selection menu, messaging, and managing overrides

When designing a workspace, dashboard, or portal experience using the UI Builder (including Dashboard Builder), a system administrator or ui_builder_admin has access to a **Domain Selection** menu in UI Builder. A system administrator or ui_builder_admin should switch to the proper domain prior to creating, editing, or overriding a variant or dashboard page.

By default, the ui_builder_admin role does not have access to the **Domain Selection** menu. The **Domain Selection** menu must be coupled with a role that grants access, such as ITIL, or it can be added via system property. For more information, see Enable domain selection menus in Core UI.

In addition, the **Domain Selection** menu also shows **Expand/Collapse Domain Scope**, that is displayed while the system administrator or ui_builder_admin is in Global. Select **Expand** to show any variant or dashboard that has been overridden, or exists as a standalone in a sub domain. Select **Collapse** to only show variants or dashboards created in Global.

Lastly, domain hierarchy is available from the **Domain Selection** menu. For deep-domain hierarchies, the user may have to collapse the branches of the domain hierarchy to physically select the domain. In these environments, perform a search to find the domain.

UI Builder has governance controls for editing and overriding variants or dashboards, similar to the way application scope is handled. Both application scope and domain scope are handled concurrently in UI Builder.

For example, if a variant was created in Global, but the current domain of the system administrator is set to Acme, then that variant is read-only. As long as that screen is not in a private scope that prevents editing, the system administrator or ui_builder_admin have two options. They can temporarily transact into the global domain if they have access to Global. Or, they can create an override.

You can edit the domain separation to make quick changes to the variant or dashboard and its content. When you edit the domain, you temporarily transact into the same domain scope as the variant or dashboard. Going into the same scope prevents accidental overrides when modifying certain settings such as Name, Order, Event Mappings, Page Definition configurations) tied to the variant. While in edit mode, not all settings are available in page management. For full capabilities, switch into the correct domain prior to editing the record.

Create Override allows a system administrator or ui_builder_admin to create an override of an existing variant or dashboard. Create an override of a variant or dashboard to perform an extensive copy of the page definition content, minus screen conditions and audiences in the user's currently selected domain. The sys_override column is then updated appropriately.

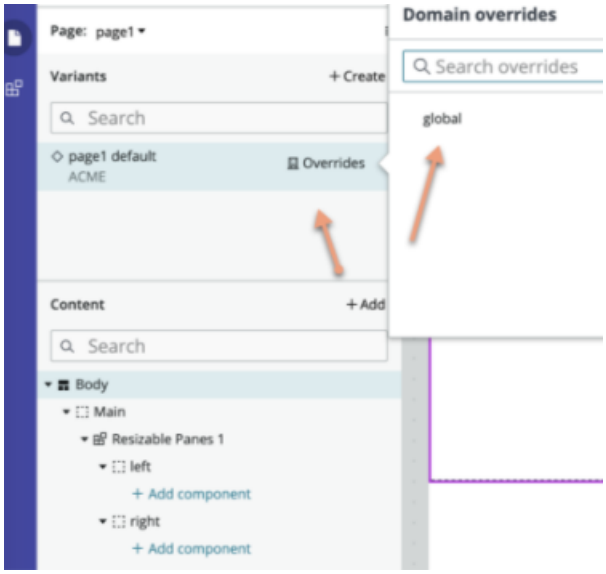
Viewports, which are variants in and of themselves, are domain separated, and are typically nested inside page definition content. Some viewports may not copy over. For example, a viewport (displayed as a tab set) that was created as an override in a domain of a Global viewport would not be carried in the page definition content during the override creation process.

As screen conditions and audiences may be specific to a domain, this content is not carried over during the override creation process. A screen prompts the system administrator or ui_builder_admin to create screen conditions and audiences.

A user cannot create an override of a variant or dashboard in Global if the item exists in a sub domain, or if an override exists for that variant or dashboard in the same sub domain.

After the override and the conditions and audiences are set, the content and configurations can be configured as needed. As standard to domain separation, the override is no longer affected by any changes done to the original variant or dashboard. The workspace, dashboard, or portal experience displays these overridden configurations if the user's current domain session is within the affected domain or sub domains where this override was created. Audiences further determine what a user may or may not see.

In addition, a user can access the domain hierarchy to view existing overrides from higher domains. For example, Global <- Top <- Acme <- Current domain. If no overrides exist, the default variant or dashboard are displayed. The exception is if the default variant or dashboard is in a child domain or a peer domain.



If you select **Expand Domain Scope** while in Global, all variants and overrides in sub domains are shown as previously mentioned.

System administrators and ui_builder_admin can see what has been created in the ServiceNow® platform.

Viewports and domain separation

Viewports are variants that can be nested in page definition content. They can be created as a common configuration in Global, or can be overridden per sub domain.

Declarative actions and domain separation

Declarative Actions can be overridden per domain as well. A system administrator or ui_builder_admin should select the appropriate domain prior to creating a domain specific declarative action override.

Related topics

Domain separation for service providers

Working in UI Builder

Get a full understanding of everything that you can do when creating a page for your workspace or custom portal experience in UI Builder.

Using UI Builder resources

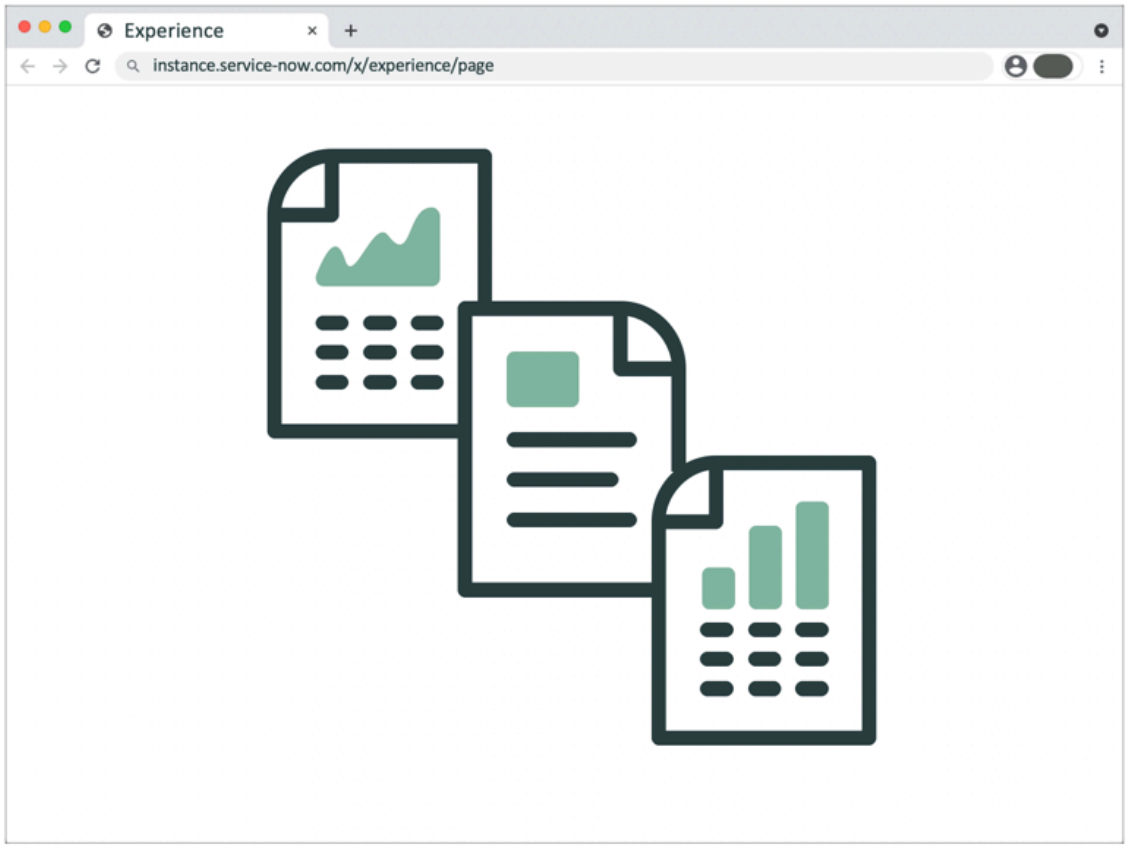
The following information helps you along your UI Builder journey:

Configure how users interact with your applications in UI Builder

Learn what an experience is in UI Builder. Understand what an experience contains.

What is a UI Builder experience

An experience in UI Builder is a collection of web pages for users to interact with an application.



Many websites use the same basic structure on every page. For example, every page contains the same header, the same footer, and the same menus. These common page elements are called a shell. Experiences can have a shell, or wrapper, for each page in the experience. A shell gives an experience a consistent look and feel as users navigate from page to page. Shells are recommended but not required. For more information, see Define UI experiences using app shells.

Ways to create an experience

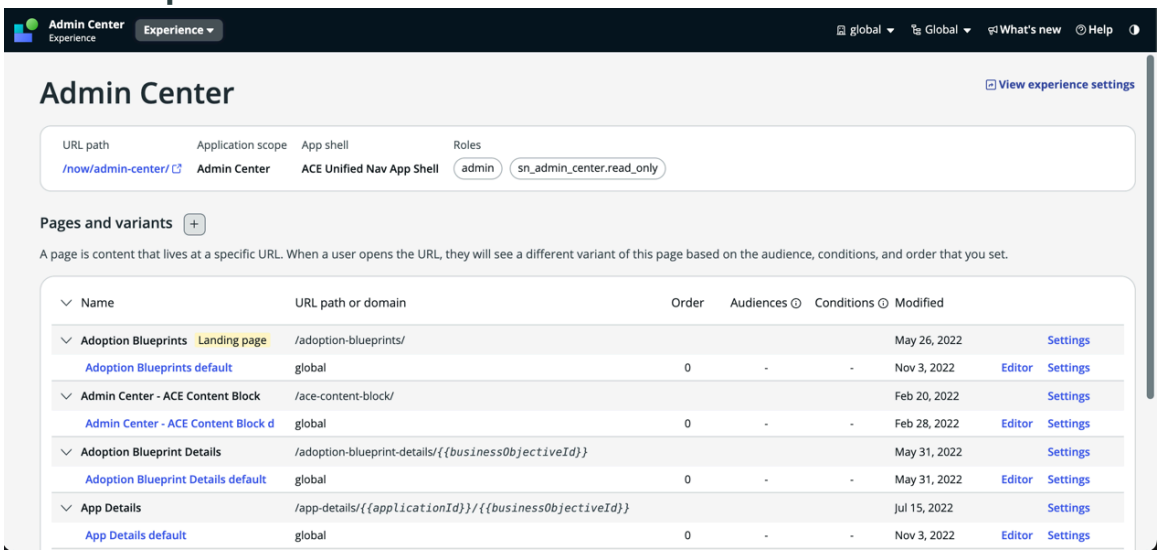
You can Create an experience for UI Builder from scratch or use UI generation to assist you in building an experience.

Experience view in UI Builder

The UI Builder experience view is a central place to view and understand the structure and details of an experience. This includes routes, page variants, and the audience and conditions required for each variant, and experience settings. Use the experience view to see the structure and hierarchy of your experience.

- View the experience URL, application scope, app shell, and roles for the experience.
- Edit experience settings.
- Quickly view and compare variants.
- Create pages and page variants.
- View a list of page collections.
- Edit page and page variant settings.
- Duplicate page variants.

UI Builder experience view



Create an experience for UI Builder

Learn how to create a workspace or portal experience for UI Builder in the ServiceNow platform.

Before you begin

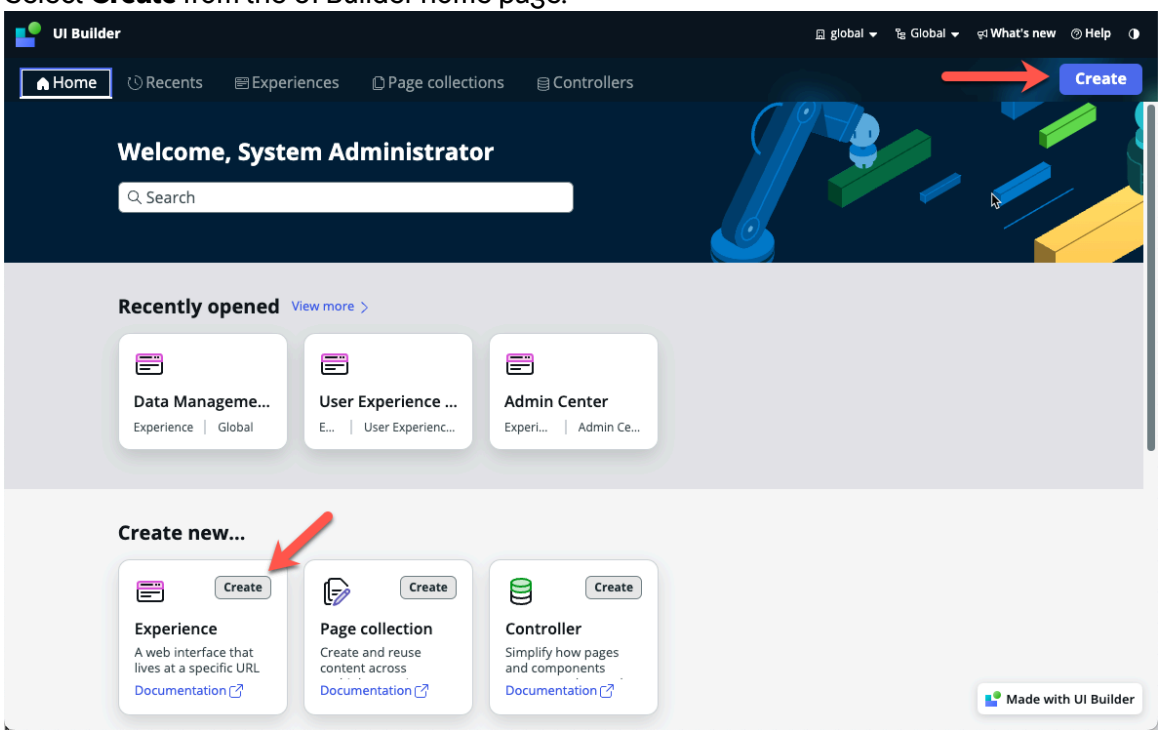
Role required: ui_builder_admin

About this task

This task shows you a basic example of how to create a workspace or portal experience for UI Builder. This example creates an experience called My App to demonstrate the process.

Procedure

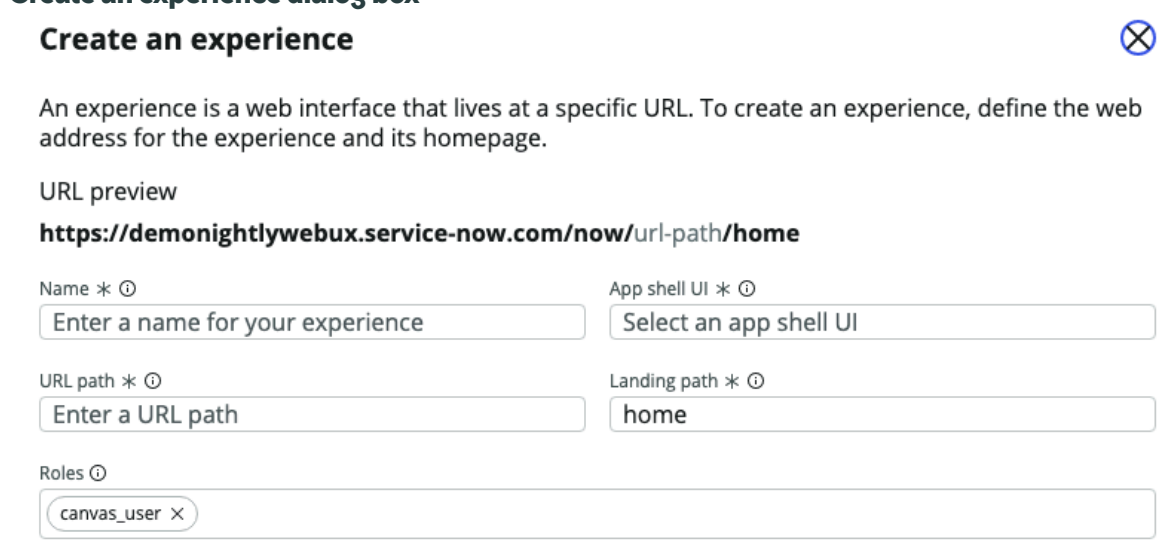
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Select **Create** from the UI Builder home page.



3. Select **Experience**.

4. In the form, fill in the fields.

Create an experience dialog box

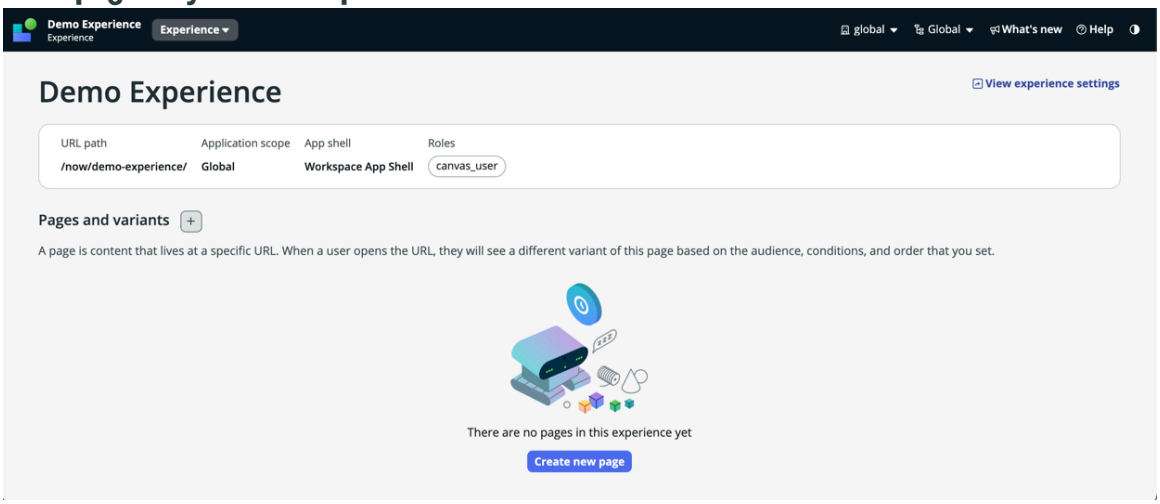


Create an experience form

Field	Description
Name	Add a name to track your experience internally. The experience name is visible to users from the browser tab. For this example, it is `MyApp`.
App Shell UI	You must choose an app shell for the experience to work. For example, you could choose a workspace or portal app shell. The app shell is the wrapper of the page contents, which is similar to the functionality of a web page. The app shell can show things like the logo of your company, user preferences, the search icon. For more information, see Define UI experiences using app shells.
URL path	Add a path for your experience. The path appends to the end of the URL parameter of your experience. For example, if you used `MyApp` as the title for your experience, you could type `my/app` for the path.
Landing path	The landing path is the prefix that people use to reach your experience homepage. To designate a page as the homepage, you must create a page that has a matching path.
Roles	Only users with these assigned roles can access the experience. If you leave this field empty, the experience is open to all logged-in users by default.

5. Select **Create** to create your experience.
6. Select **Open experience** to view the main page for your experience.

Main page for your new experience



Note:
In addition to the name of your new experience, the screen includes the URL, application scope, admin panel, and the roles of users who can view the experience. Any pages or variants you create for the experience appear in the **Pages and variants** section.

What to do next

Now that you have created an experience you can add and customize pages. For more information, see Manage UI Builder pages and page variants.

Configure UI Builder workspace experiences

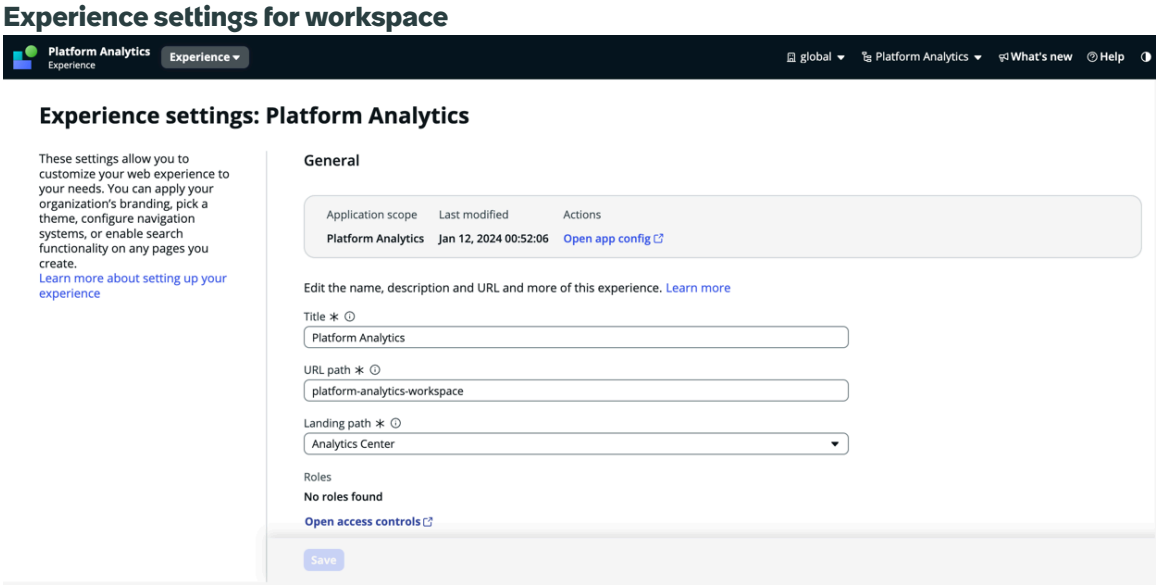
You can change the UI Builder workspace experience settings to fit your company goals.

By changing the experience settings, you can affect how your users interact with your workspace experience, how your workspace looks, and how users navigate to and around your workspace.

Before you can edit the experience settings, you must be in the correct application scope. If you're in a different scope, the experience settings are read-only. To change your application scope, go to the main header, select the application picker (Global), and then select the application scope that you want. For more information about the application scope, see Learn about security and roles.

Learn how you can change these workspace experience settings:

- Change or add the workspace general settings for your workspace experience. For example, you can change the title, the description, and the path of the workspace.
- View the brand and theme setting in your workspace experience to see the theme currently assigned to your experience. The theme sets the look and feel of the experience.
- Change the workspace side navigation settings to add pages to the side navigation in your workspace experience.
- Modify the workspace utility settings to turn on or turn off notifications for your workspace experience. Also, you can change the global search settings for your workspace experience. For example, you can choose to show or hide the search bar in your workspace experience or change the search source that determines where the search results come from.



Change the general settings in your workspace experience

Change the general settings for your workspace experience in UI Builder to fit the needs of your organization. For example, you can modify the title, description, and path for your workspace experience.

This video show you how to perform the following procedure. This video shows you how to change the general setting in a UI Builder workspace experience.

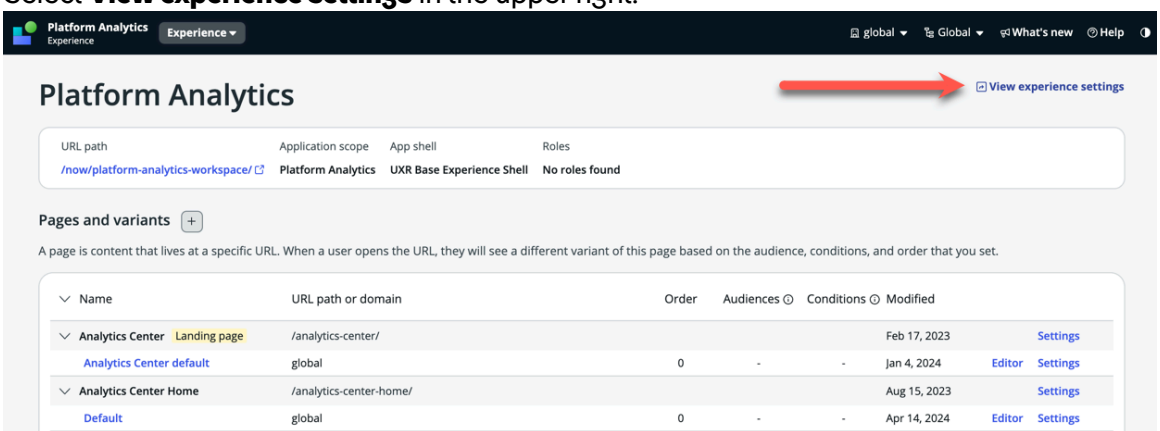
Before you begin

You must be in the correct application scope to edit the experience settings. If you're in a different scope, the experience settings are read-only. To change your application scope, go to the main header, select the application picker (Global), and then select the application scope that you want. For more information about the application scope, see Application scope.

Role required: ui_builder_admin

Procedure

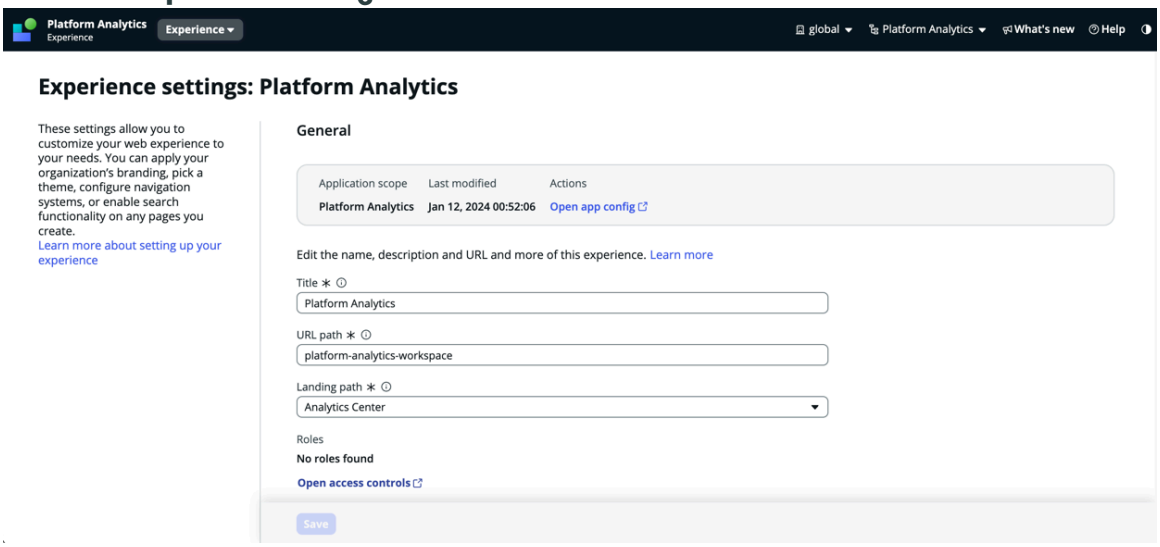
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. For more information, see Configure how users interact with your applications in UI Builder.
3. Select **View experience settings** in the upper right.



4. Change the title, description, and path of your workspace experience.

- The title lets everyone know what the workspace is called. Take care when choosing a name for your workspace.
- The description lets your users know the details of the workspace.
- The path must be unique. The path can include digits (0-9), letters (A-Z, a-z), and a few special characters (`" - "`, `" . "`, `" _ "`, or `" ~ "`) with the words separated by a forward slash or hyphen.

UI Builder experience settings



5. **Optional:** Select **Advanced settings** to edit the workspace record on the ServiceNow AI Platform.

   a. Update the record in the platform when you're finished.
   When you go back to your workspace experience settings in UI Builder, you see the changes that you made to the general settings.

6. Click **Save**.

View the brand and theme setting in your workspace experience

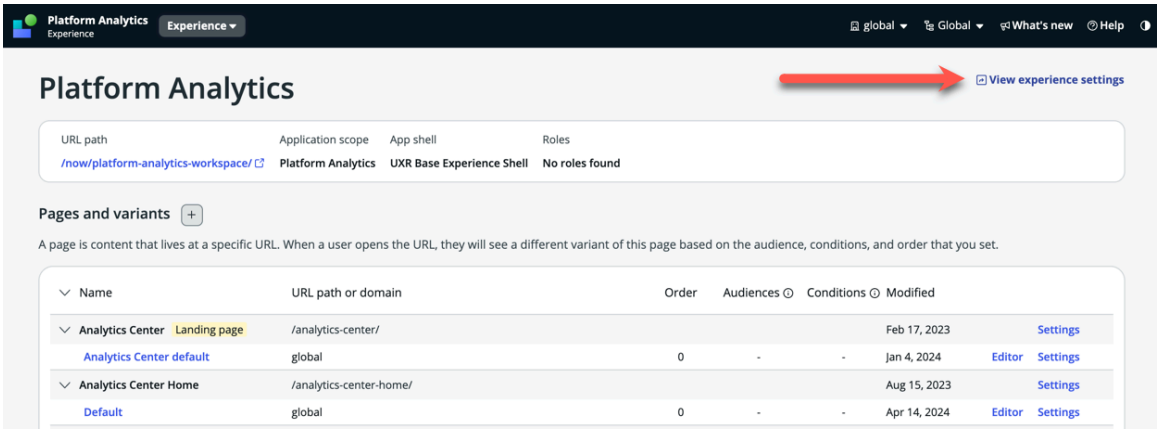
View the theme setting for your workspace experience in UI Builder. The theme sets the visual style of the experience and provides a consistent look and feel across all pages.

Before you begin

Role required: ui_builder_admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Select **View experience settings** in the upper-right.



4. Scroll to the **Branding and theming** section and note what is listed in **Themes**.
   By default, the Polaris theme is applied to UI Builder experiences.

What to do next

If you want to customize the theme, there are two options:

- **Theme Builder** enables you to create, edit, manage, and apply Next Experience themes using a friendly visual interface. For more information, see Configuring Next Experience with Theme Builder.
- If you are an admin user comfortable working in style records and JSON code, you can create a custom theme to override the default Polaris style. This option gives you the most control over typefaces, colors, and images (including the banner and logo) in a theme. For more information, see Configuring Next Experience themes and preferences.

Change the navigation and menu settings in your UI Builder workspace experience

Select pages for side navigation in your UI Builder workspace experience. From any page in the workspace experience, users can navigate to the pages you selected.

Before you begin

This task has the following prerequisites:

- You have an existing workspace experience on UI Builder. For more information, see Configure how users interact with your applications in UI Builder.
- This workspace experience was created with the Workspace App Shell.
- This workspace experience includes at least two pages.
- You want users to be able to navigate from a page to other pages that you specify.

Role required: ui_builder_admin

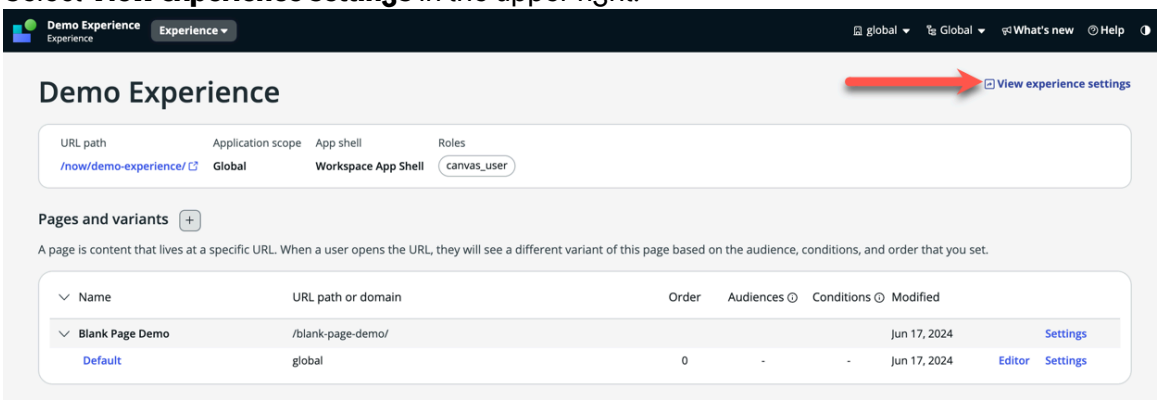
About this task

You can add up to seven pages to your side navigation.

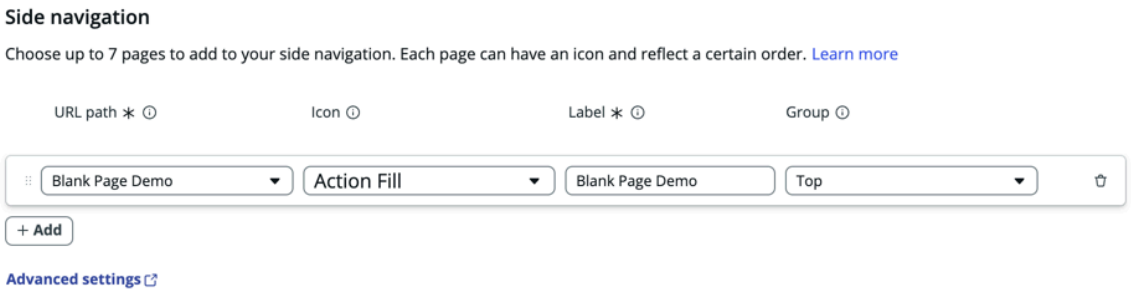
You must be in the correct application scope to edit the experience settings. If you are in a different scope, the experience settings are read-only. To change your application scope, go to the main header, select the application picker (Global), and then select the application scope that you want. For more information about the application scope, see Learn about security and roles.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open your workspace experience in the UI Builder.
3. Select **View experience settings** in the upper-right.



4. Scroll down to **Side navigation**.
5. Select **+ Add**.



6. Fill in the following fields:

Field	Description
URL path (required)	Select the page that you want to navigate to. **Tip:** The landing page is usually the first page that you want a link to.
Icon	Select an icon for the link. Default: No icon
Label (required)	Enter a name for the page, which appears in a tooltip for the link.
Group	Place the link in the top or the bottom group of the side navigation panel. Default: Top

7. Add more pages as needed.
8. Select **Save**.

9. Select the experience name (for example, Demo Experience) in the UI Builder header.
10. Select the **URL path** link to open the experience in a new tab and check that the side navigation is displayed on the left side.

> **Tip:**
> If the side navigation doesn't work as expected, consider reviewing the UX Page Property [sys_ux_page_property] record underlying the side navigation. Reopen the Side Navigation experience settings, as described in Steps 3-4. Select **Advanced side navigation settings** to open the record. Verify that the fields have the correct values, such as Type=`json`.

Set up notifications in workspace utility settings

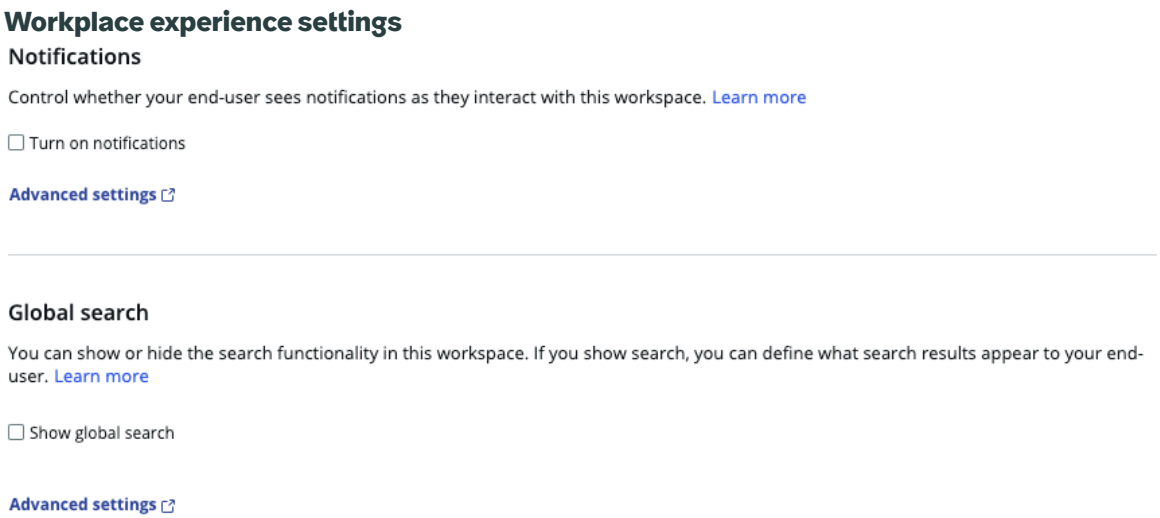
Set up the advanced notification and search settings in your workspace experience so that you can control how your users see notifications and search options.

Before you begin

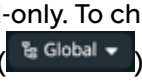
Role required: ui_builder_admin

About this task

Set up the advanced notification and search settings in your workspace experience. You can turn notifications on or off to control how people see notifications in your workspace experience.



You can also control whether the search functionality is visible to the users of your workspace experience. If you choose to display a search option, you can define what search results are returned by choosing a source for the search.

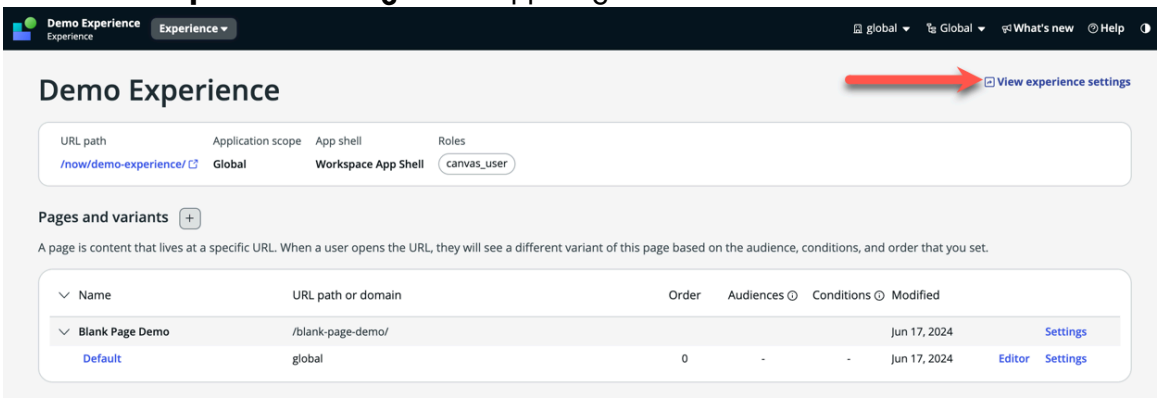
Before you can edit the experience settings, you must be in the correct application scope. If you're in a different scope, the experience settings are read-only. To change your application scope, go to the main header, select the application picker (

), and then select the application scope that you want. For more information about the application scope, see Learn about security and roles.

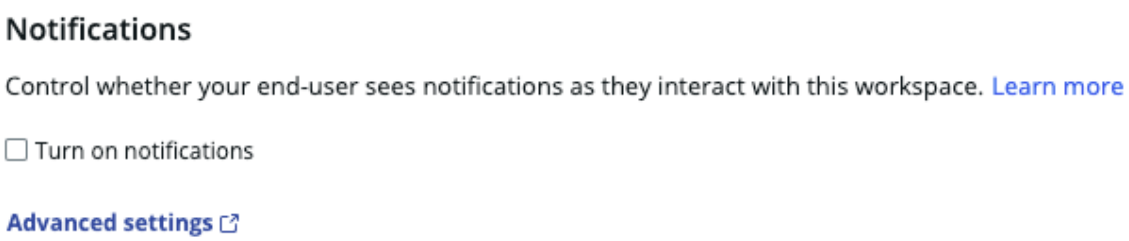
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. For more information, see Configure how users interact with your applications in UI Builder.
3. Open or create a page in your workspace experience.

4. Select **View experience settings** in the upper-right.



5. Scroll down to **Notifications**.



6. Select the **Turn on notifications** option.
7. Click **Save**.
8. **Optional:** Click **Advanced settings** to go to the ServiceNow AI Platform® and edit the JSON values of the record.

> **Note:**
> If you have the Record component on a page, you can have a record open in the workspace experience when a notification is selected. Follow the instructions in Record UIB Setup on the ServiceNow developer site and define a featureRoutes page property.

Display global search in a workspace experience

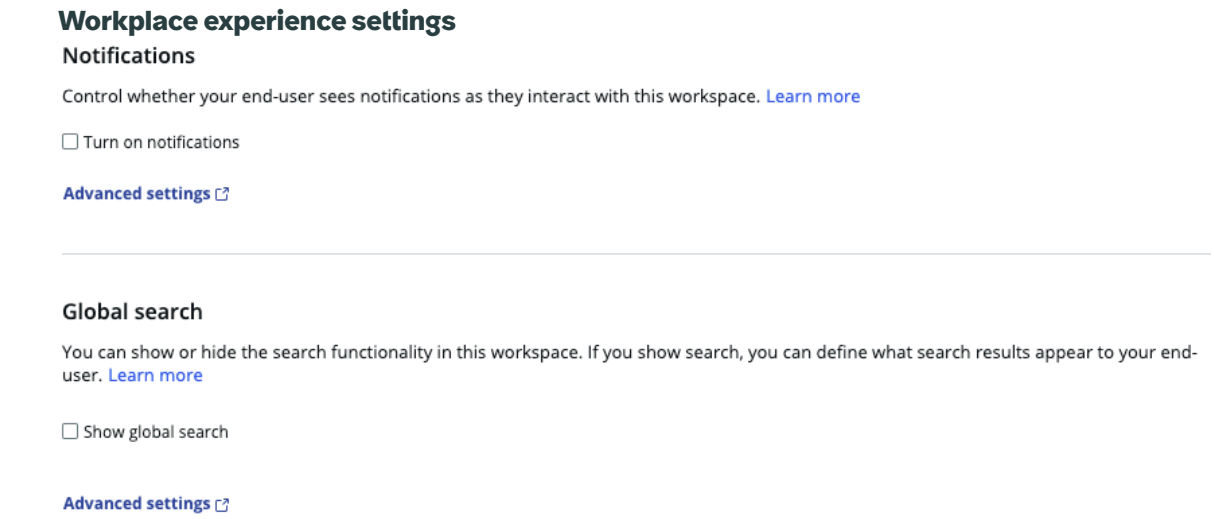
Show or hide the search functionality in your workspace.

Before you begin

Role required: ui_builder_admin

About this task

You can control whether the search functionality is visible to the users of your workspace experience. If you choose to display a search option, you can define what search results are returned by choosing a source for the search.

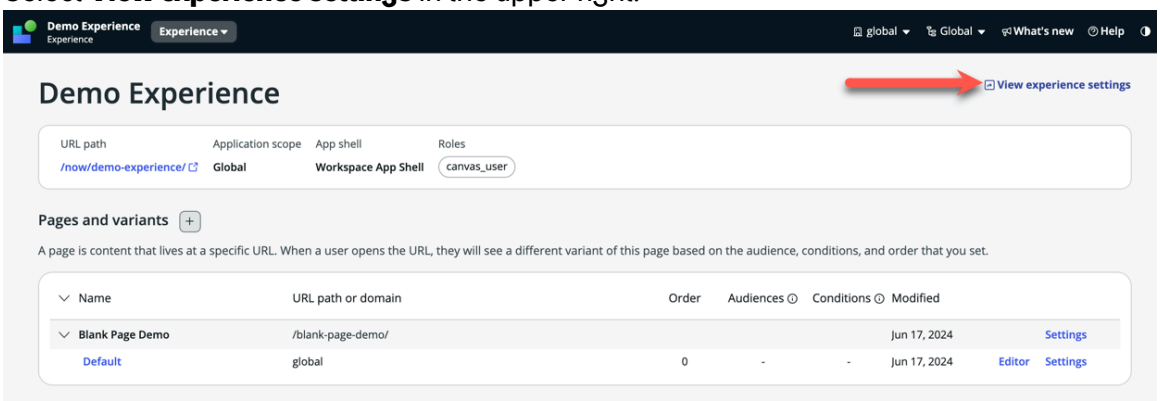


Before you can edit the experience settings, you must be in the correct application scope. If you're in a different scope, the experience settings are read-only. To change your application scope, go to the main header, select the application picker (

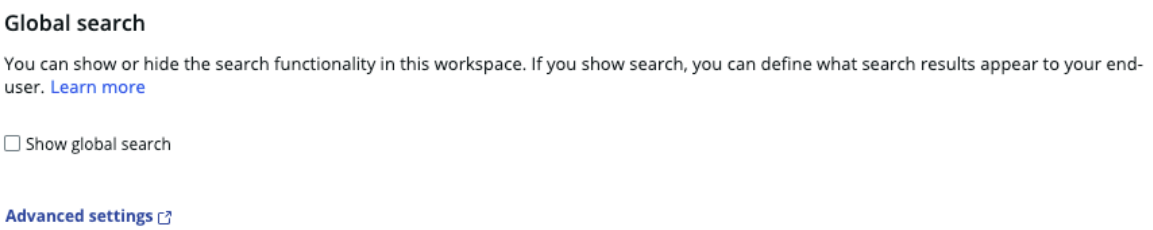
), and then select the application scope that you want. For more information about the application scope, see Learn about security and roles.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. For more information, see Configure how users interact with your applications in UI Builder.
3. Open or create a page in your workspace experience.
4. Select **View experience settings** in the upper-right.



5. Scroll down to **Global search**.



6. Select the **Show global search** option.

7. Click **Save**.
8. **Optional:** Click **Advanced settings** to go to the ServiceNow AI Platform® and edit the JSON values of the record.

Configure UI Builder portal experiences

You can change your UI Builder custom portal experience settings to affect how your users interact with your portal experience.

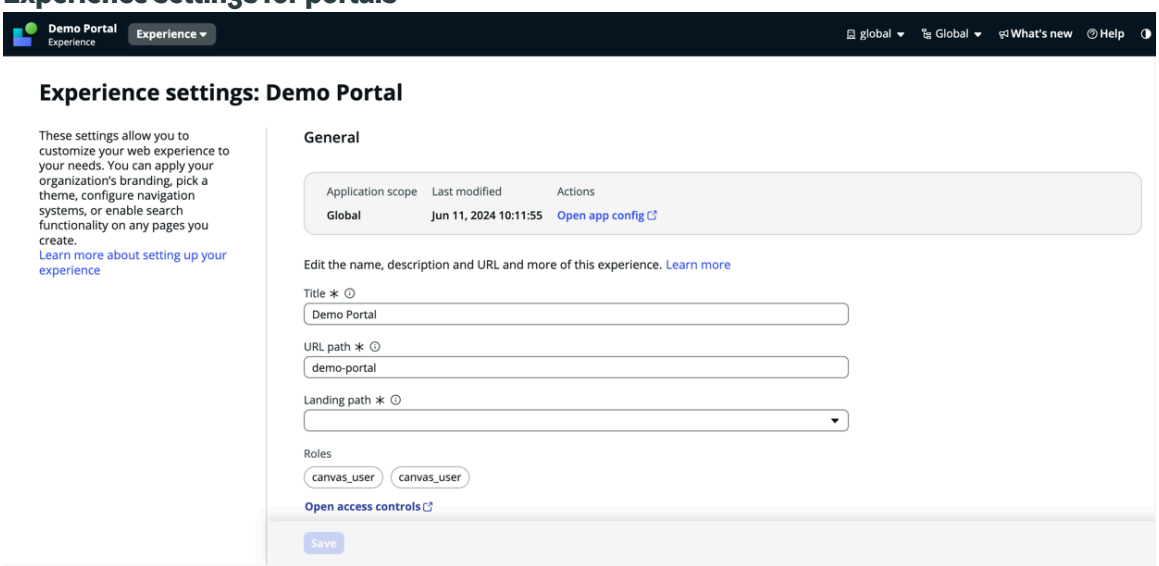
Use experience settings to change the settings for the custom portal experience that you are working in. These settings affect how your users interact with your portal experience, how your portal looks, and how users navigate to and around your portal.

You must be in the correct application scope to edit experience settings. If you are in a different scope, the experience settings are read-only. To change your application scope, in the main header, select the application picker (Global), and then select the application scope that you want. For more information about the application scope, see Learn about security and roles.

Learn how to change the following portal experience settings.

- Modify or add portal general settings for your portal experience. Change the title of your portal, the description, and the path of the portal.
- View the brand and theme setting in your portal experience to see the theme currently assigned to your experience. The theme sets the look and feel of the experience.
- Use portal navigation and menu settings to set up the navigation and menu settings in the app shell of your portal experience. The app shell is the wrapper of the portal contents. For example, the app shell can show things like the logo of your company, user preferences, the search icon, the configuration icon, and the user menu. For more information about app shells, see Define UI experiences using app shells.
- Modify the portal search settings to show or hide the global search functionality on public or private pages in your portal experience.

Experience settings for portals



Change the general settings in your portal experience

Change the general settings for your custom portal experience in UI Builder. For example, you can modify the title, description, and path for your portal experience.

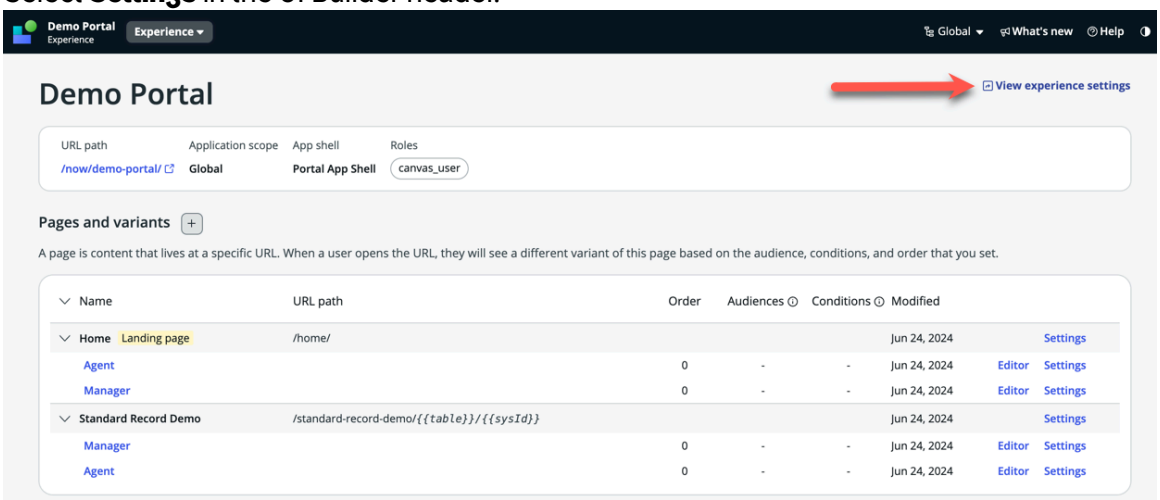
Before you begin

You must be in the correct application scope to edit the experience settings. If you're in a different scope, the experience settings are read-only. To change your application scope, go to the main header, select the application picker (Global), and then select the application scope that you want. For more information about the application scope, see Learn about security and roles.

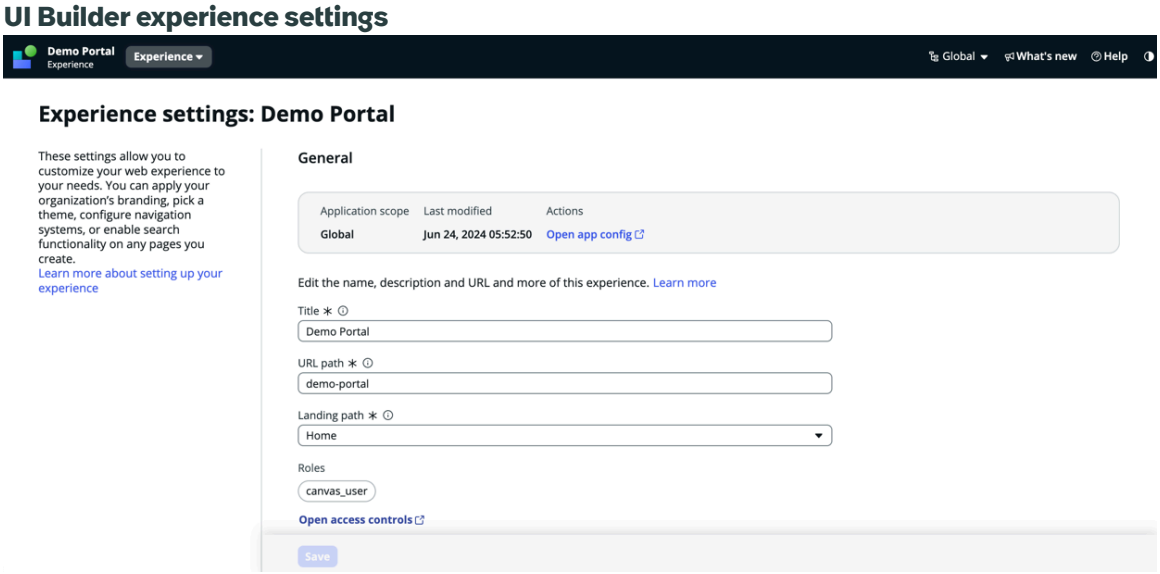
Role required: ui_builder_admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open a portal experience to work in or create an experience by selecting **Create** > **Experience**. For more information, see Configure how users interact with your applications in UI Builder.
3. Select **Settings** in the UI Builder header.



4. Change the title, description, and path of your portal experience.
   - The title lets everyone know what the workspace is called. Take care when choosing a name for your workspace.
   - The description lets your users know the details of the workspace.
   - The path must be unique. The path can include digits (0-9), letters (A-Z, a-z), and a few special characters (`" - "`, `" . "`, `" _ "`, or `" ~ "`) with the words separated by a forward slash or hyphen.



5. **Optional:** Select **Advanced settings** to edit the workspace record on the ServiceNow AI Platform.

   a. Update the record in the platform when you are finished.
   When you go back to your workspace experience settings in UI Builder, you see the changes that you made to the general settings.

6. Click **Save**.

Result

When you go back to your portal experience settings in UI Builder, you see the changes that you made to the general settings.

View the brand and theme setting in your portal experience

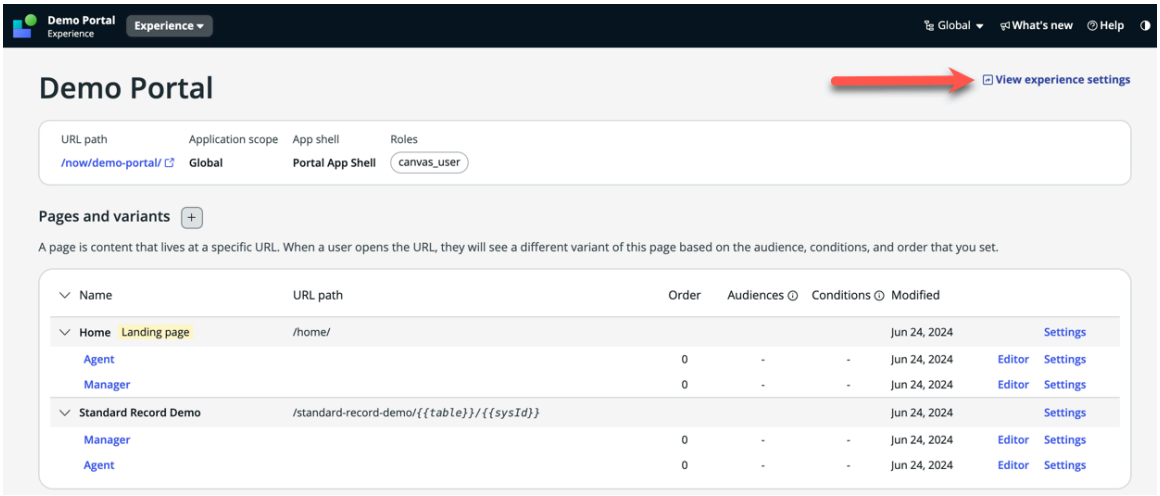
View the theme setting for your portal experience in UI Builder. The theme sets the visual style of the experience and provides a consistent look and feel across all pages.

Before you begin

Role required: ui_builder_admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open a portal experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Select **Settings** in the UI Builder header.



4. Scroll to the **Branding and theming** section and note what is listed in **Themes**.
   By default, the Polaris theme is applied to UI Builder experiences.

What to do next

- **Theme Builder** enables you to create, edit, manage, and apply Next Experience themes using a friendly visual interface. For more information, see Configuring Next Experience with Theme Builder.
- If you are an admin user comfortable working in style records and JSON code, you can create a custom theme to override the default Polaris style. This option gives you the most control over typefaces, colors, and images (including the banner and logo) in a theme. For more information, see Configuring Next Experience themes and preferences.

Change the navigation and menu settings in your portal experience

Change the header, footer, and menu settings for your custom portal experience in UI Builder to fit the needs of your organization.

Before you begin

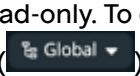
This task has the following prerequisites:

- You have an existing portal experience on UI Builder. For more information, see Configure how users interact with your applications in UI Builder.
- This workspace experience contains at least two pages.
- You want users to be able to navigate from a page to other pages that you specify.

Role required: ui_builder_admin

About this task

Set up the navigation and menu settings in the app shell of your portal experience. The app shell is the wrapper of the portal contents. For example, the app shell can show things like the logo of your company, user preferences, the search icon, the configuration icon, and the user menu. You can configure this app shell in the ServiceNow AI Platform. For more information about app shells, see Define UI experiences using app shells.

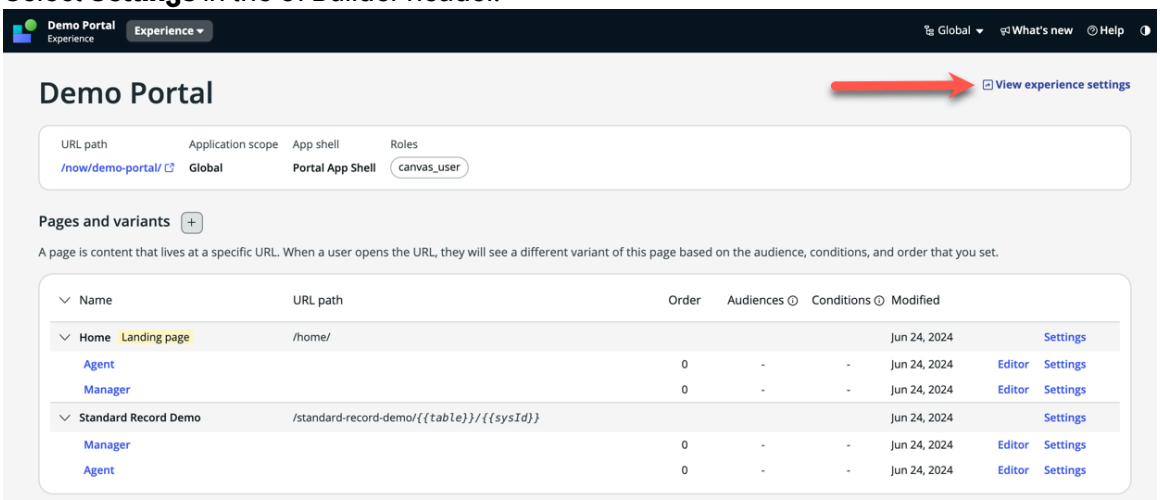
You must be in the correct application scope to edit experience settings. If you are in a different scope, the experience settings are read-only. To change your application scope, in the main header, select the application picker (

), and then select the application scope. For more information about the application scope, see Learn about security and roles.

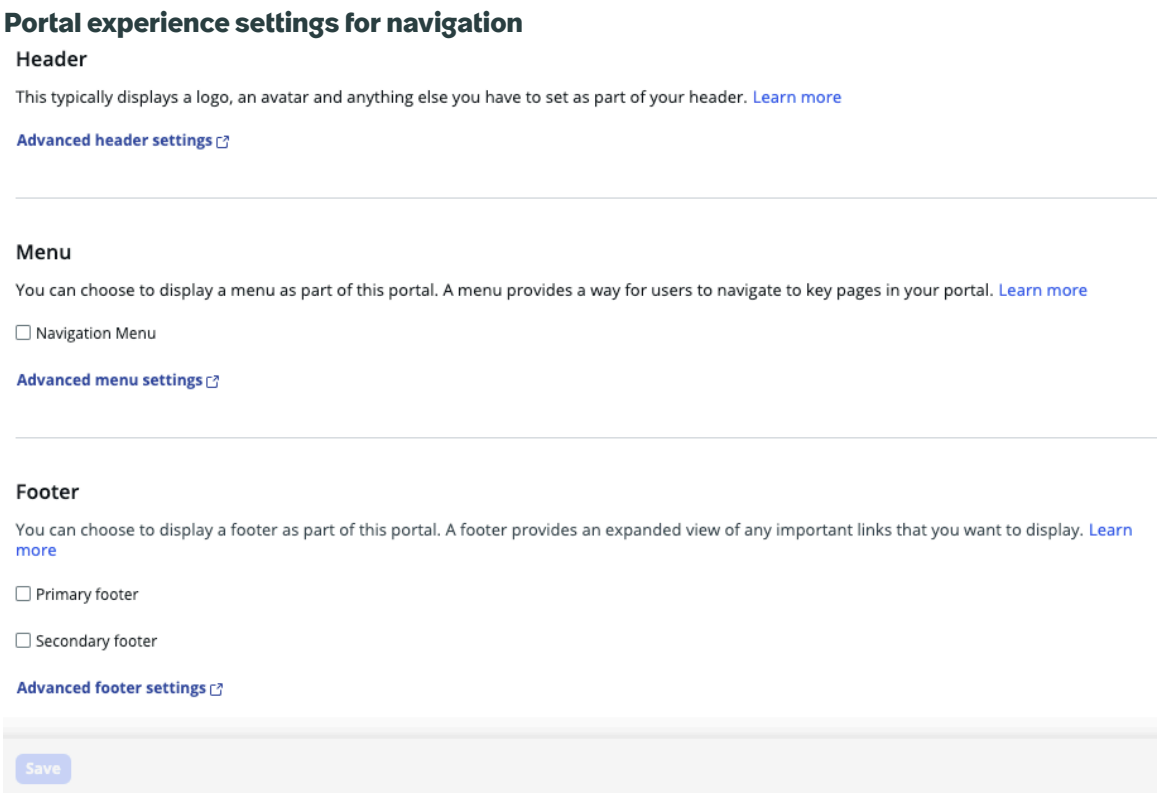
Note:
For advanced information about UX page properties, including JSON examples, see https://www.servicenow.com/community/next-experience-articles/workspace-app-shell-ux-page-properties/ta-p/2331956.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open a portal experience to work in or create an experience by selecting **Create** > **Experience**. For more information, see Configure how users interact with your applications in UI Builder.
3. Select **Settings** in the UI Builder header.



4. To change the settings for the header, footer, and navigation menu for the app shell of the portal experience you're working in, select the advanced settings links.



Action	Procedure
Change the header	To go to the ServiceNow AI Platform®, click **Advanced header settings**. Here, you can change what information shows in the header of your portal experience. You can change things like a company logo, user preferences, the search icon, the configuration icon, and so on.
Show the menu	**a.** To add a menu to your portal experience, select **Navigation Menu**. **b.** To remove the navigation menu, deselect **Navigation Menu**. **c.** To go to the ServiceNow AI Platform, select **Advanced navigation menu settings**. Here, you can change what menu and menu items are displayed in the navigation menu. For more information, see Define UI experiences using app shells.
Display a footer	**a.** To add a primary footer to your portal experience, select **Primary footer**. The footer can show a footer logo, content, and other important links for your portal. **b.** To go to the ServiceNow AI Platform, click **Advanced navigation menu settings**. **c.** To add a second footer to your portal experience, select **Secondary footer**. A secondary footer includes additional text links, social media elements, and copyright information.

5. Click **Save**.

Show or hide the search settings for your portal experience

Show or hide the global search functionality on the public or private pages for your custom portal experience in UI Builder so that your users can search on the portal.

Before you begin

Role required: ui_builder_admin

You can control whether the global search functionality is visible to users of your portal experience. You can enable or disable the global search for either public or private pages.

You must be in the correct application scope to edit the experience settings. If you are in a different scope, the experience settings are read-only. To change your application scope, in the main header, select the application picker (

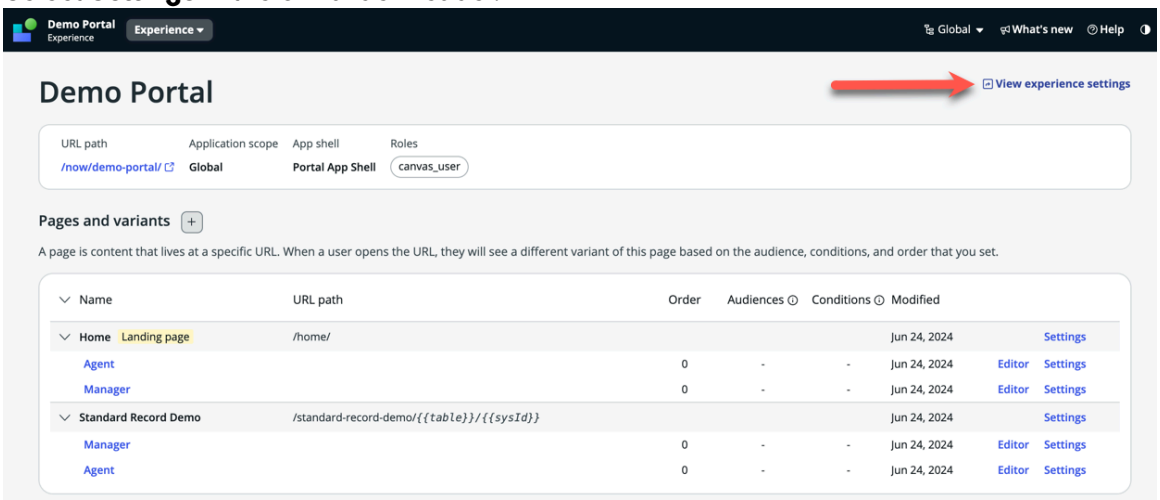
), and then select the application scope. For more information about the application scope, see Learn about security and roles.

Procedure

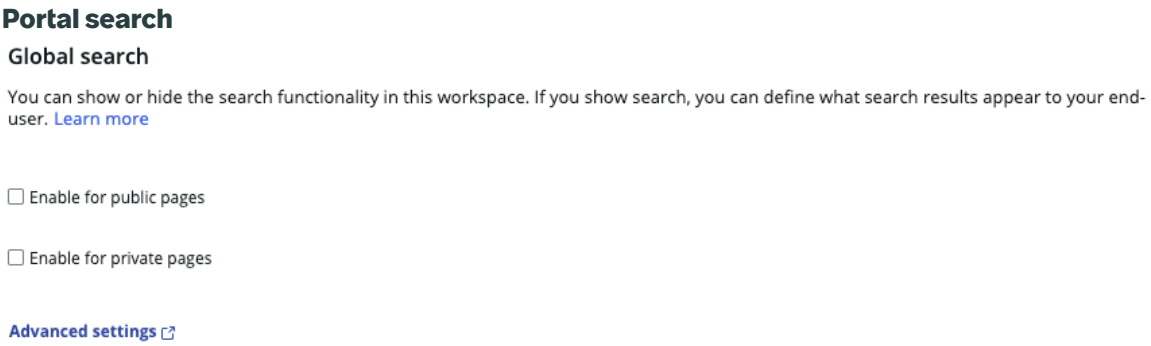
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open a portal experience to work in or create an experience by selecting **Create** > **Experience**.

For more information, see Configure how users interact with your applications in UI Builder.

3. Select **Settings** in the UI Builder header.



4. Scroll down to **Global search**.
5. Select the **Enable for public pages** or **Enable for private pages** to show or hide the global search in public or private pages.



6. **Optional:** Click **Advanced settings** to go to the ServiceNow AI Platform to make changes to the application record on the ServiceNow AI Platform®.
   To enable or disable the search for a page, you must create a separate UX Page property. The UX Page property overrides the global search setting.
7. Click **Save**.

Define UI experiences using app shells

Understand what app shells are, what app shells are available for UI Builder, and why you would pick one over another.

App shells

An app shell is the wrapper of the contents of an experience page. An app shell includes functionality similar to a web page. For example, an app shell can show things in a header or footer of your experience, such as a logo for your company, user preferences, a search icon, configuration icon, user menu, and so on.

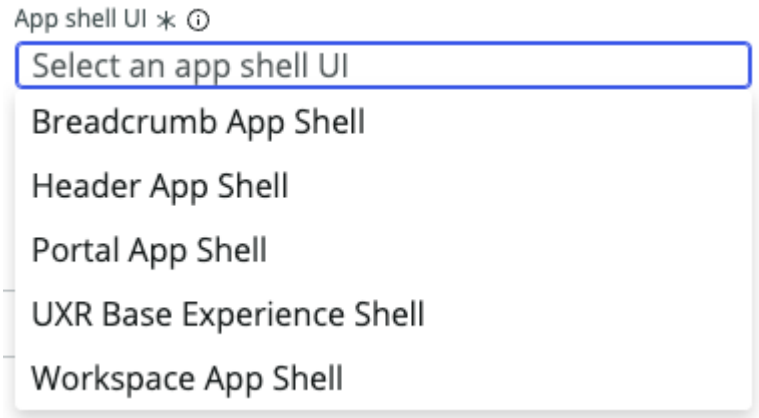
An app shell is required for UI Builder.

The app shell defines whether your experience has a workspace or portal design. A workspace is a graphical user interface that puts multiple tools on one page for handling requests from users.

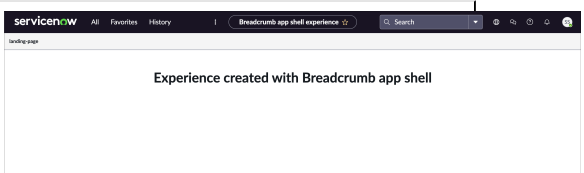
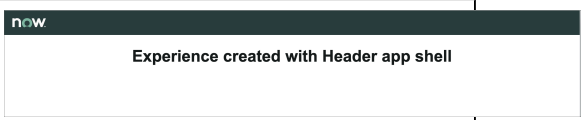
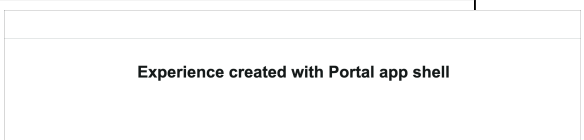
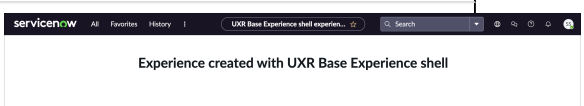
A portal is a page where users can add requests, such as order items, track their tickets, and so on.

You choose which app shell to apply to your experience when you create an experience in the ServiceNow® platform. For more information on how to create an experience, see Configure how users interact with your applications in UI Builder.

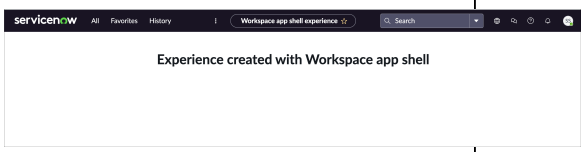
You can choose from various app shells when creating an experience.



App shell descriptions

App shell	Description	Example
Breadcrumb app shell	The breadcrumb app shell includes a header, breadcrumb style navigation, and a primary navigation bar for a workspace experience. Use the breadcrumb app shell for focused workflows and wizard flows that don't require multi-tasking.	
Header app shell	The header app shell includes a header and user menu for a workspace experience. Use the header app shell for experiences that require minimal navigation and few actionable items in the header.	
Portal app shell	The portal app shell includes the header and footer for a portal experience. The portal app shell record is a reference implementation of how the menu, utilities, logo, and login are configured for a portal experience.	
UXR Base Experience app shell	The UXR Base Experience shell creates an experience with the Next Experience banner. The UXR Base Experience shell provides base functionality support for experiences that are configured within a parent app, including experience configuration, context binding, routing, screen loading, and	

App shell descriptions (continued)

App shell	Description	Example
	caching. The UXR Base Experience shell has no visual or experience-specific components. The UXR Base Experience shell can also be used for experiences that have no shell customization. For example, Identity Center directly uses UXR Base Experience Shell as its app shell.	
Workspace app shell	The workspace app shell includes the headers and footers for a workspace experience. Use the workspace app shell to provide tools that users need to find, research, and resolve issues. To manage the visual style of experiences created with this app shell, see Configuring Next Experience with Theme Builder.	

Manage UI Builder pages and page variants

Learn what a page is in UI Builder. Understand the building blocks of a UI Builder page, such as column layouts and components.

UI Builder quick start

Create pages in UI Builder as part of a workspace or custom portal experience. UI Builder pages consist of layouts and components. You build a page using column layouts and components to guide a user through an experience. For example, you could build a page to manage travel requests for your employees. The page could have column layouts with components that contain lists of all travel requests submitted and approved. You can add buttons that let users add and submit travel requests. The way you build your page is limitless.

Note:
One developer should work on any one variant at a time.

Column layouts and components in UI Builder

Column layouts and components are the building blocks of your UI Builder page. Add column layouts and components to your page to build or customize your workspace or portal experience. For example, add a column layout and within a column place a button component that lets users submit requests.

You can add column layouts, columns, and components, as well as navigate between them, from the stage or the Content tree.

A column layout can contain one or more columns. Columns can contain components. Change the visual styling of column layouts, columns, and components to make it your own experience.

Types of page layout elements

Type	Description
Column layout	A flexible container with one to six columns. Add column layouts to provide structure and a framework for a page.
Column	Fill columns with components.
Component	Base elements of a page, such as buttons, lists, and forms.
Modal	Use a modal to create a page type in UI Builder that renders on top of the page and requires action.
Popover	A popover is a type of container that appears above a UI Builder page when triggered by an event. Use the popover component to display additional information or actions related to the page.

See Customize UI Builder pages using components for more information.

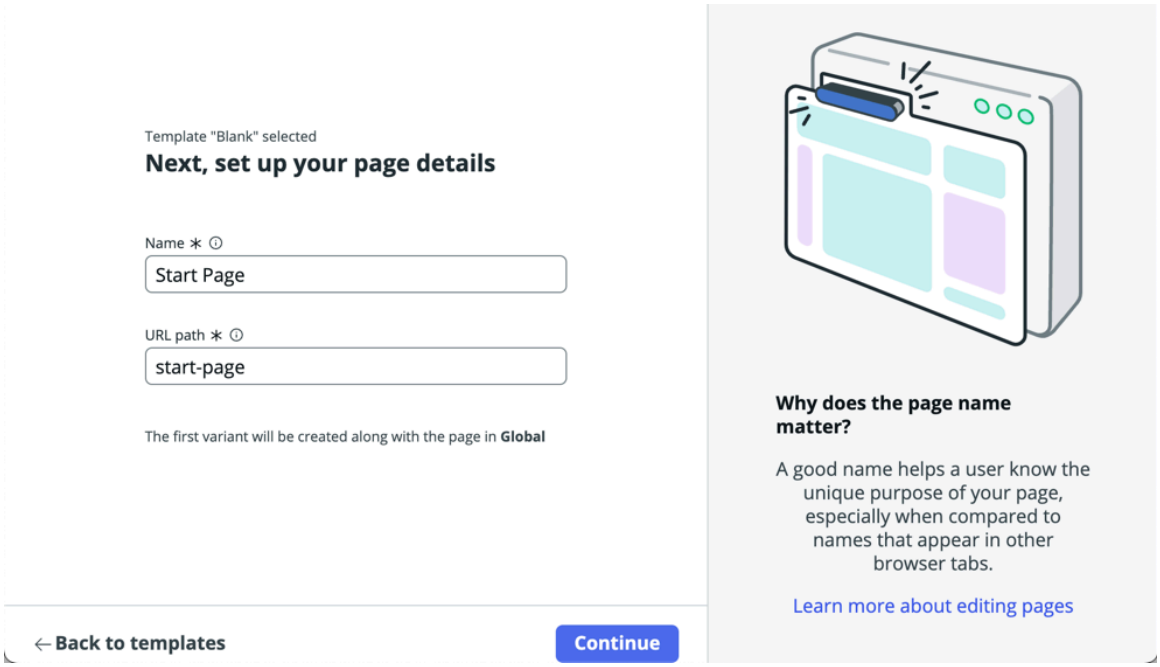
Create a UI Builder page

Create a UI Builder page to build a page experience from scratch or use a page template. Build a page one component at a time. If you use one of the pre-defined page templates, you start from a base structure and can customize it to meet your needs.

Name your UI Builder page. Set the path (or keep the default path that is automatically added based on your page name). A default path is added based on your page name. You can also create your own path, but the path must be unique. The **URL preview** shows the path of your page.

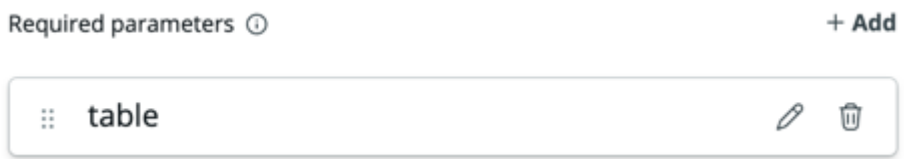
The application scope protects applications by identifying and restricting access to application files and data. The application scope defaults to the scope that the user is currently in within the ServiceNow AI Platform®. For more information about application scope, see Learn about security and roles.

Use a UI Builder page template to create a page based on a pre-defined page template, and then customize the page to your needs. You can reference or copy a page template. For more information, see Create a page from a template.



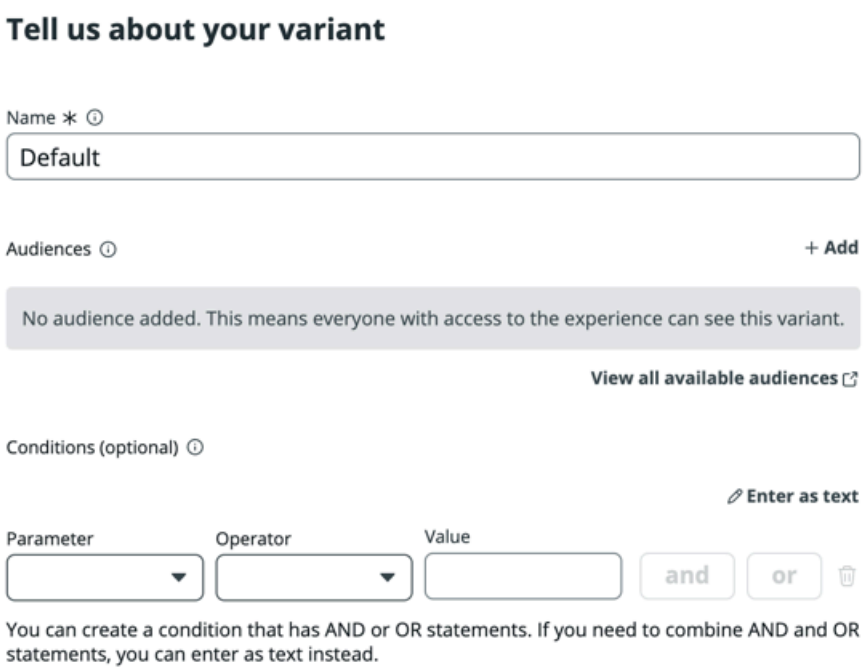
Create a UI Builder page: Advanced settings

Add required parameters to your UI Builder page. A required parameter is a piece of data that your page requires, such as a sys_id, table, or query. Required parameters are useful for components as they can bind to the value of the required parameter. For example, you can add a table parameter to the URL field and then bind data to that table. When the table is referenced, it exposes the table data to any components on the page. Required parameters are visible in the URL when you add them to your page. In the following example, a required parameter called table was added. Notice it was appended to the URL.



Add optional parameters to your UI Builder page. Optional parameters are optional pieces of data that you can add to the URL of your page. Unlike required parameters, optional parameters are always name and value pairs that work no matter what order they're provided.

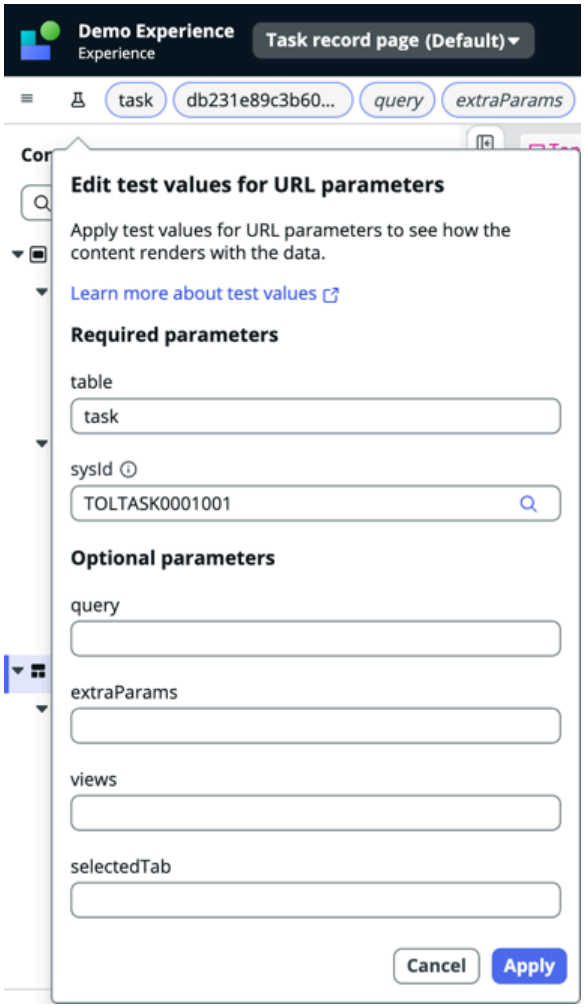
Set the audience and conditions settings for UI Builder page variants. When you create a page, UI Builder also creates a variant of the page for you by default. A page variant is a variation of your page at the same path that lets you target experiences for different audiences using user criteria. For example, a page for managers, and a variant of that page for the manager's direct reports. For more information about creating a variant, see Create a page variant.



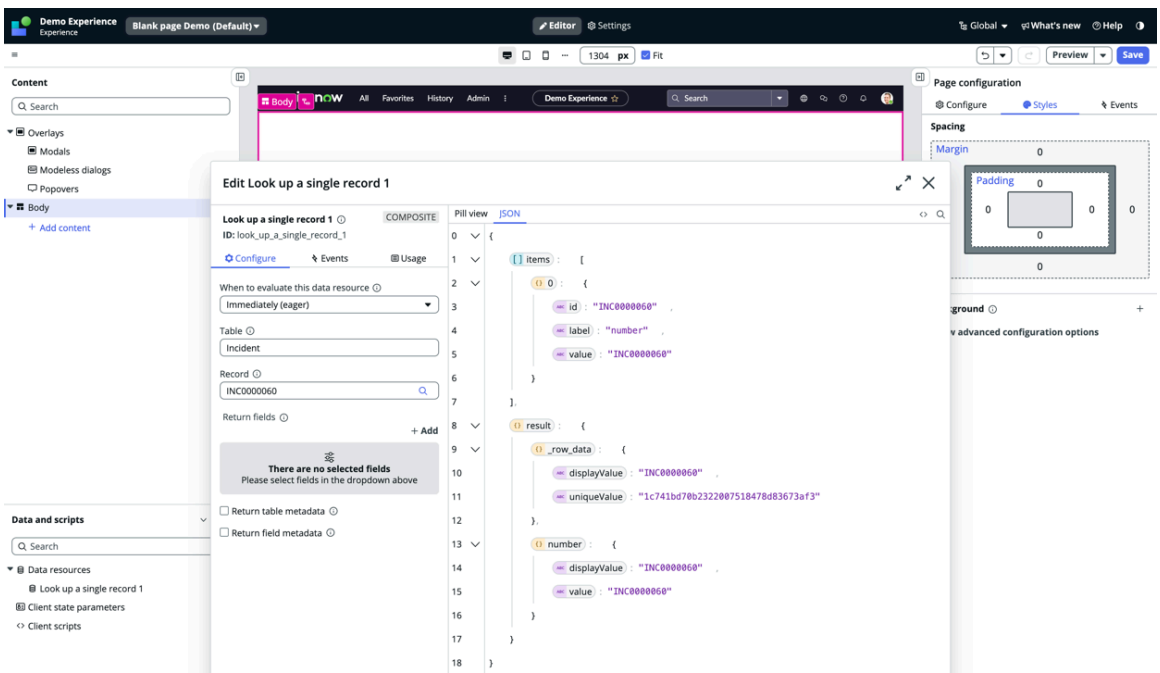
For more information about audiences, see Learn about audiences.

Test values in UI Builder

Add a test value to your UI Builder page as a way to bring test data into the page for testing purposes. For example, if you add a table as a required parameter, you could add a test value of incident and bind a data resource to it to bring in test data on the incident for that table.

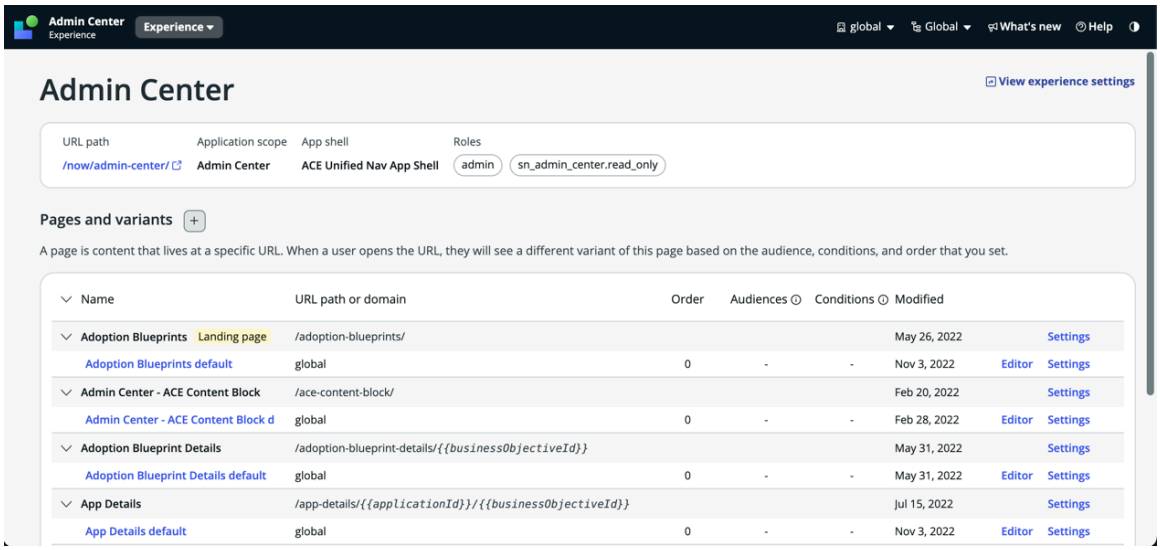


To get test values to show data, add a data resource, then configure the data resource to bind a record to the test value in the URL. For example, you could add `incident` as a test value. Then add a data resource named **Look up a single record**. In the **Table field**, dynamically bind the incident test value to a `context.props.table` table, as shown in the following image.

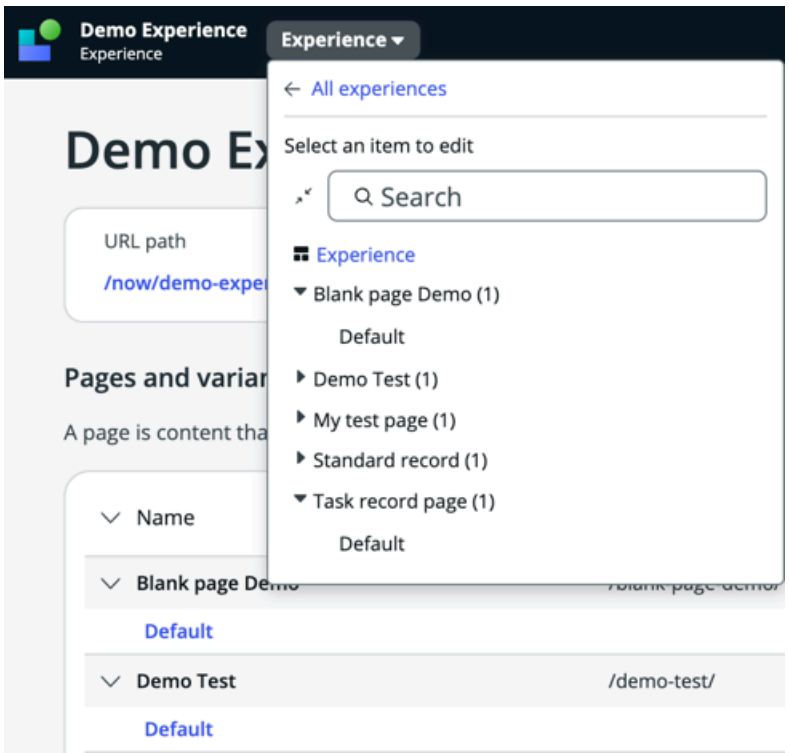


View an existing UI Builder page

You can work in any UI Builder page in your experience based on your role settings. Click on the name of the page you want to work in while in the experience view of your experience.



While in the page editor, open a different page variant by selecting the drop-down in the header.



Create a UI Builder page variant

A variant lets you target different audiences with different content, using user criteria. For example, you can create a homepage for agents, and a variant of the page for managers that exists at the same URL path.

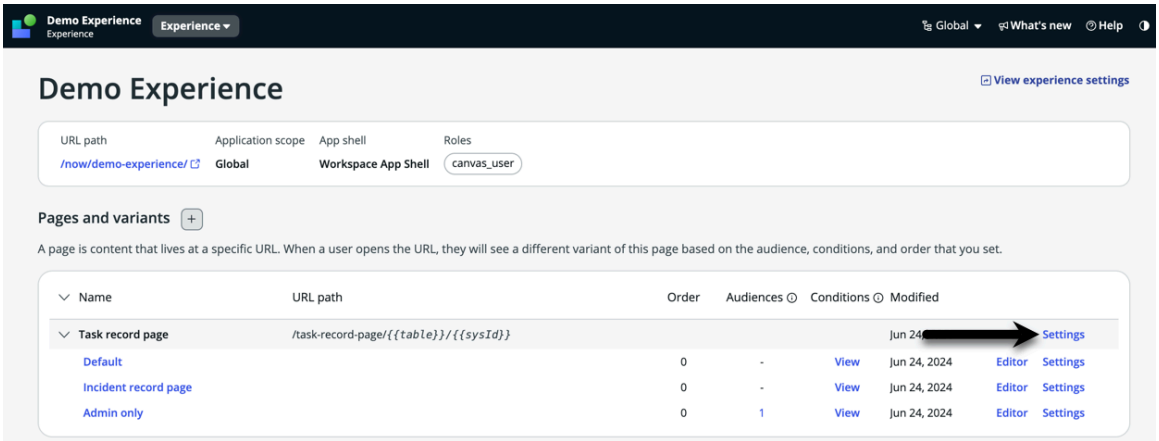
You set the audience for each UI Builder page variant. The audience determines who uses the page variant. For example, if you create a travel request page, create a variant of that page for managers to manage the employee travel requests. You set the audience for the manager page

for anyone in the manager role. Employees cannot view that variant. For more information about audiences, see Learn about audiences.

See Create a page variant for more information.

Edit UI Builder page settings

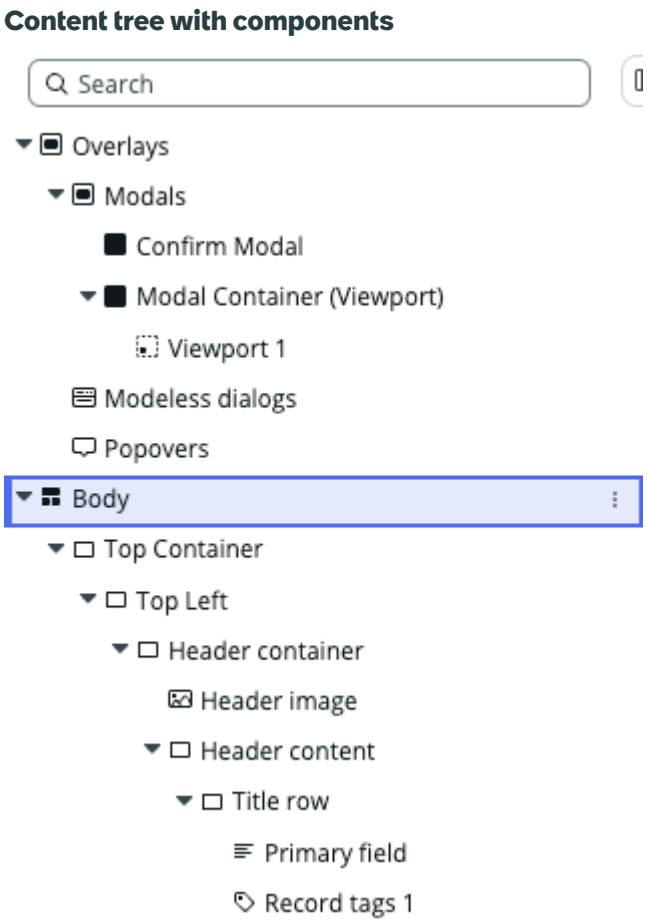
Change the settings of your UI Builder page at any time by selecting **Settings** in the UI Builder experience view. You can change the name, path, and parameters of your page after you create a page.



See Edit a page for more information.

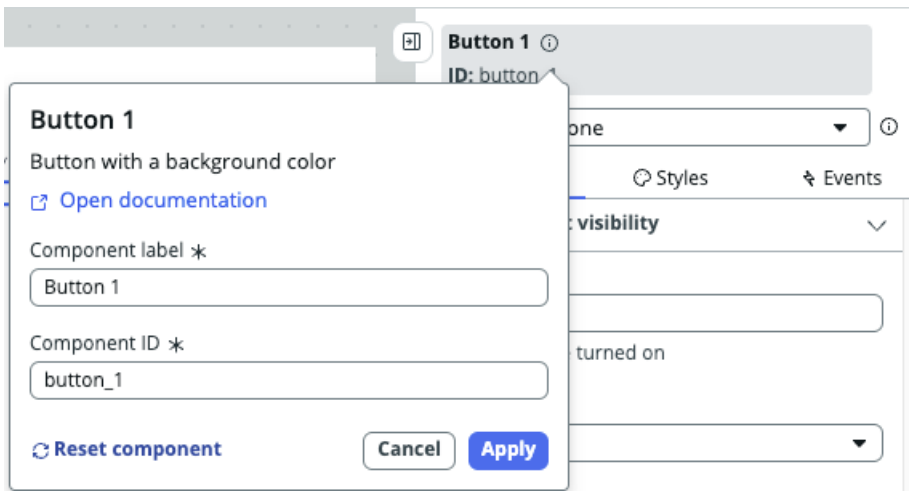
UI Builder Content tree

The content tree in the page management panel is important. It not only shows every column layout, column, and component on your UI Builder page, it lets you easily find your components and work with them. The content tree is especially important when you have multiple components on your page. You select the component in your content tree to highlight the component on the page, making it easier to build your page.



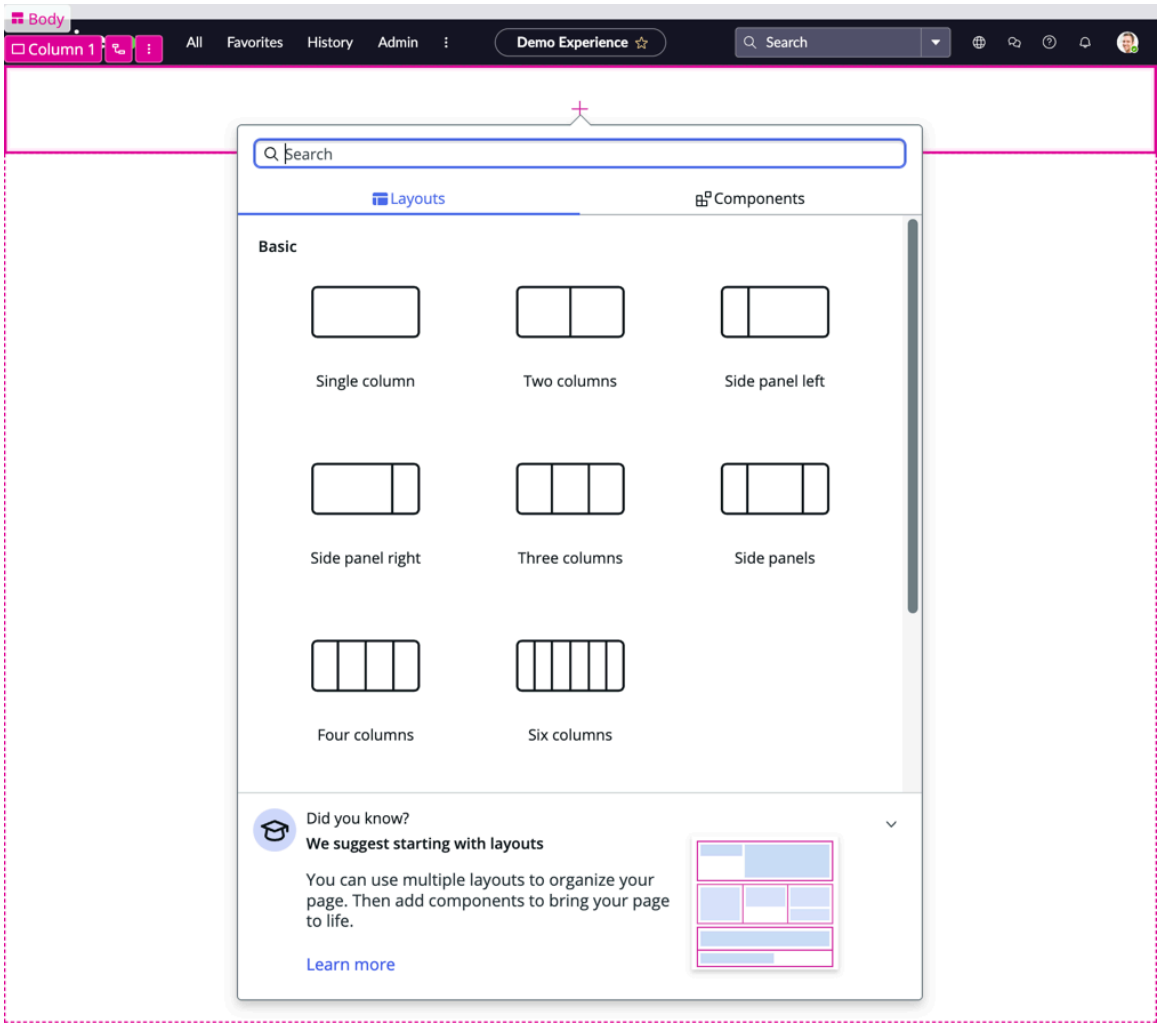
When you click a component in the content tree, it highlights the component on your stage so you can configure, add styling, events, data, and so on. You can drag items in the content tree to reorganize your page.

It's important to add a component label when you add a component to your page. The component label is used in the content tree to apply labels IDs to each component in the content tree. You can identify the components much easier in the content tree when they're labeled appropriately.



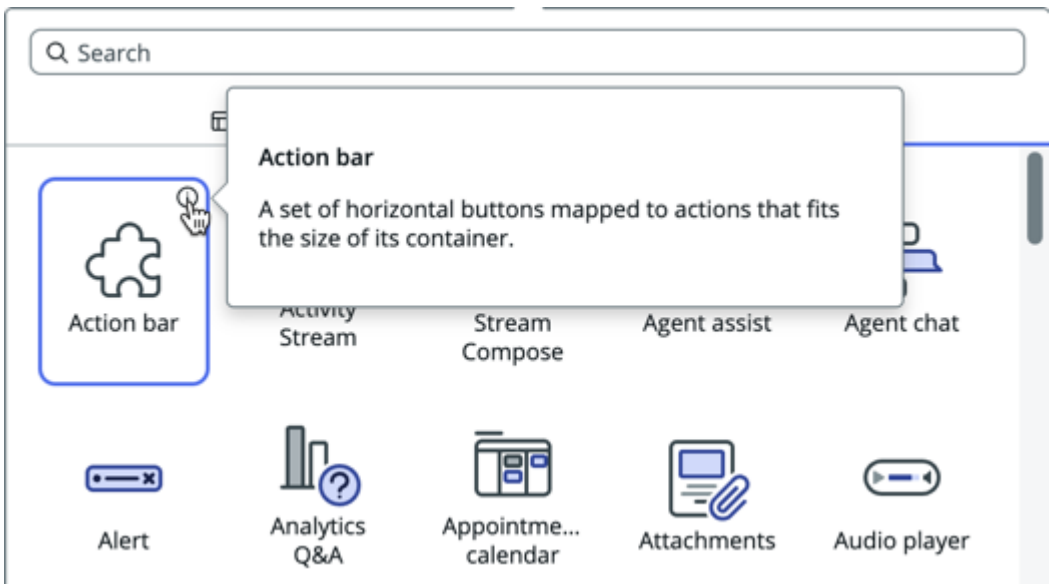
UI Builder Toolbox

Use the UI Builder toolbox to add layouts and components to your page. Access the UI Builder toolbox by selecting **+ Add content** or the **+** icon in the stage.



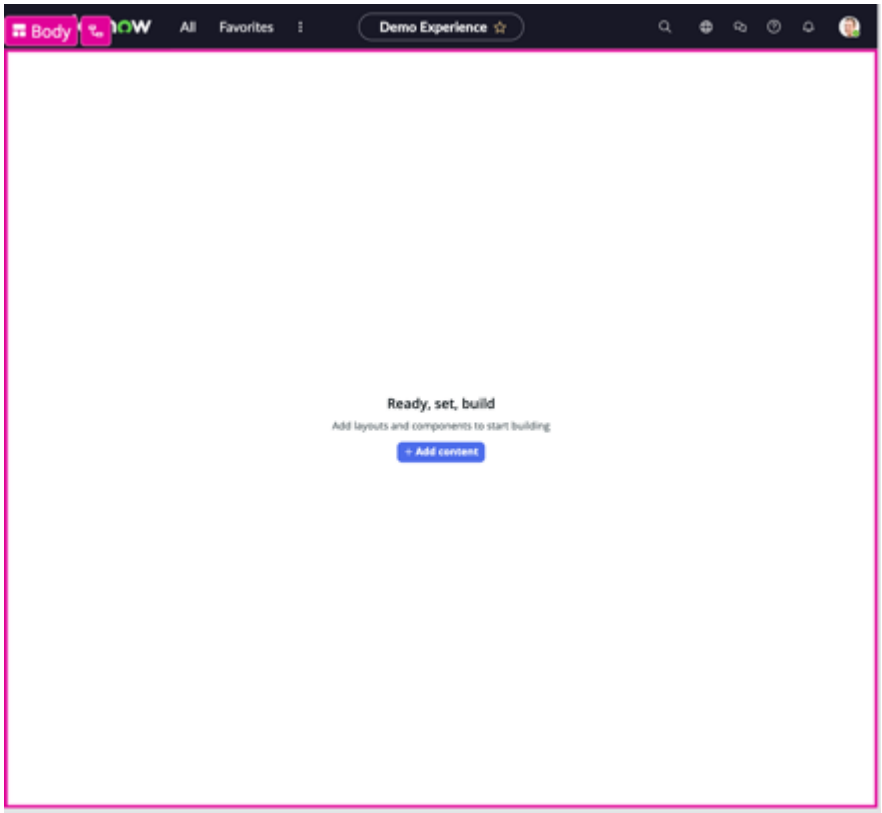
Search the toolbox to find layouts and components that you want to add to a UI Builder page.

View a brief description of a layout or component by selecting the information icon ⓘ to view the tooltip. The components tab displays different colors if a component can use a preset or is bundled with multiple components.

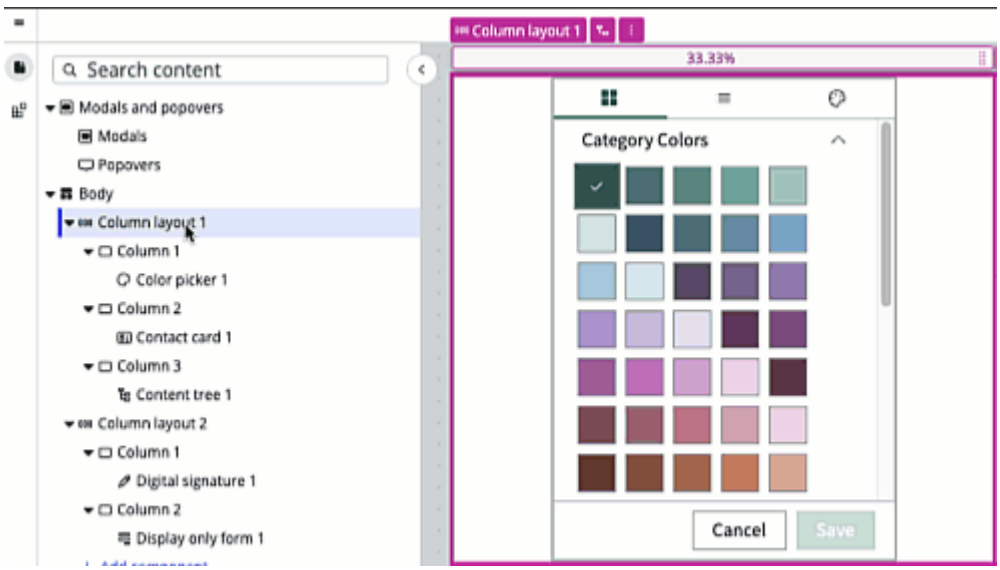


UI Builder Stage

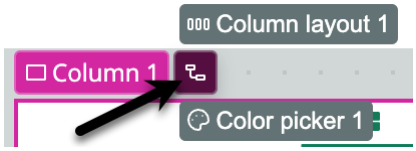
The stage is the largest area in the UI Builder editor and is used for building pages. Add your column layouts and components here by either selecting a + button, or dragging from the Component pane to the page.



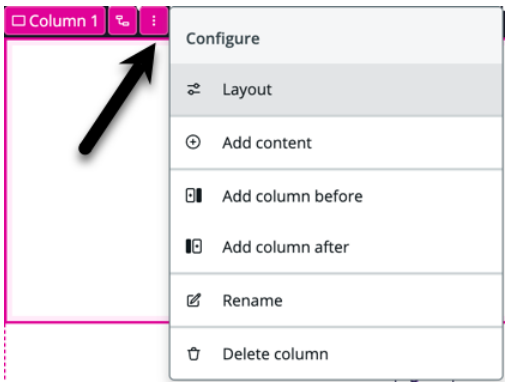
On the stage, the body, column layouts, and columns are outlined in magenta. Components are outlined in blue.



Use the Tree item icon to navigate to a different layer of content. For example, if a column is selected, easily navigate to the parent column layout or a component within the column.



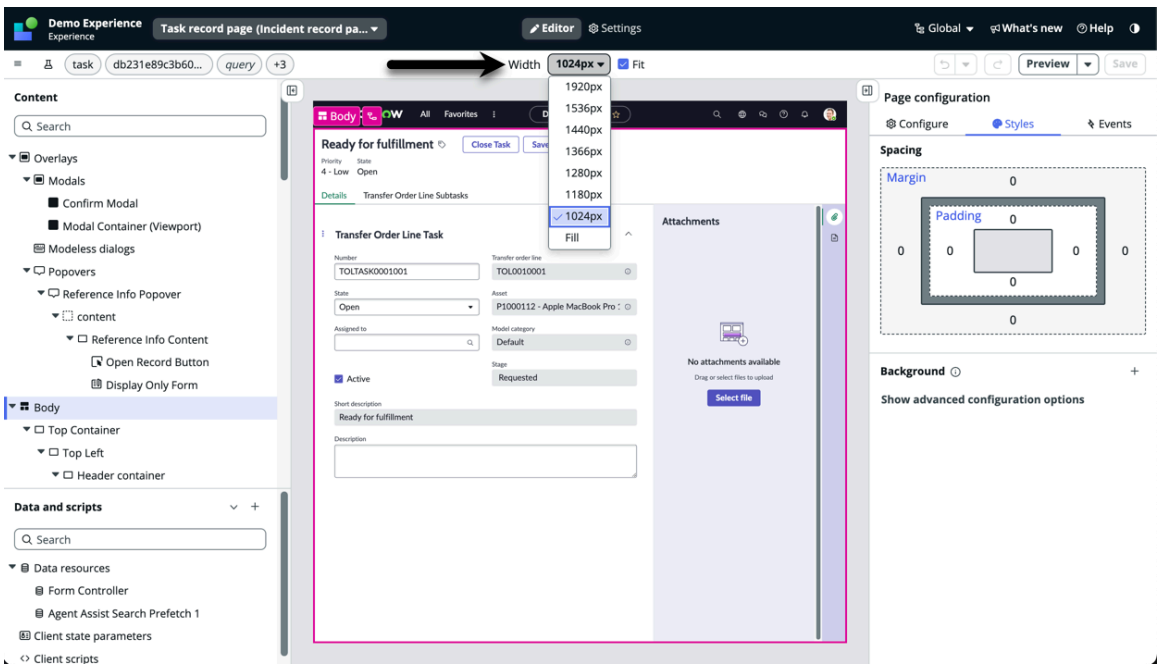
Select the Menu icon on a column layout or component for options such as duplicate and delete.



Changes made on the stage are reflected in the Styles panel at right. Changes made in the Styles panel immediately update the elements on the stage.

Scale the size of the stage in UI Builder

Adjust the size of the stage in the UI Builder editor by selecting the Width drop-down.

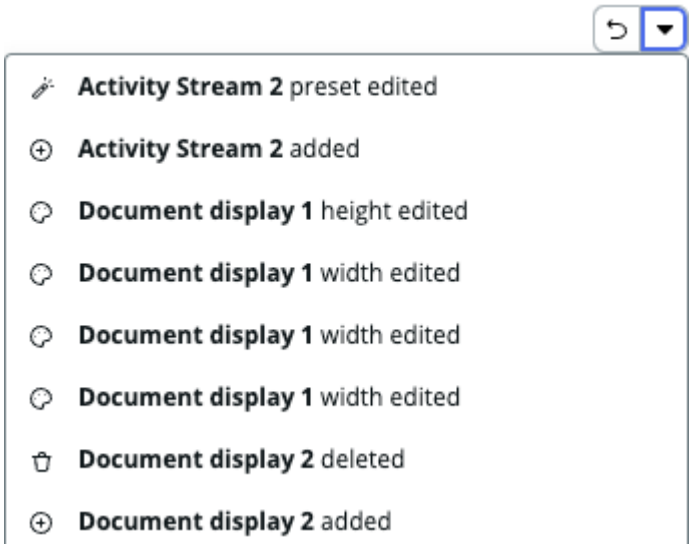


Undo and Redo in UI Builder

Undo and redo changes you've made to your UI Builder page. Reverse any action you make to a page layout, component, style or layout.

The undo and redo function can be found in the header toolbar. You can select the undo ( ) or redo ( ) button to reverse an action. You can also click the undo drop down to go back multiple steps. You cannot undo and redo changes made in the data shelf.

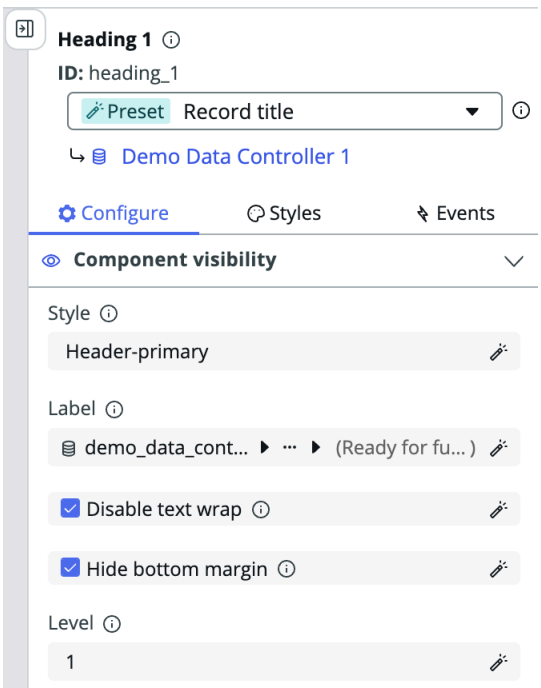
You can undo the previous 20 actions by selecting the undo drop-down.



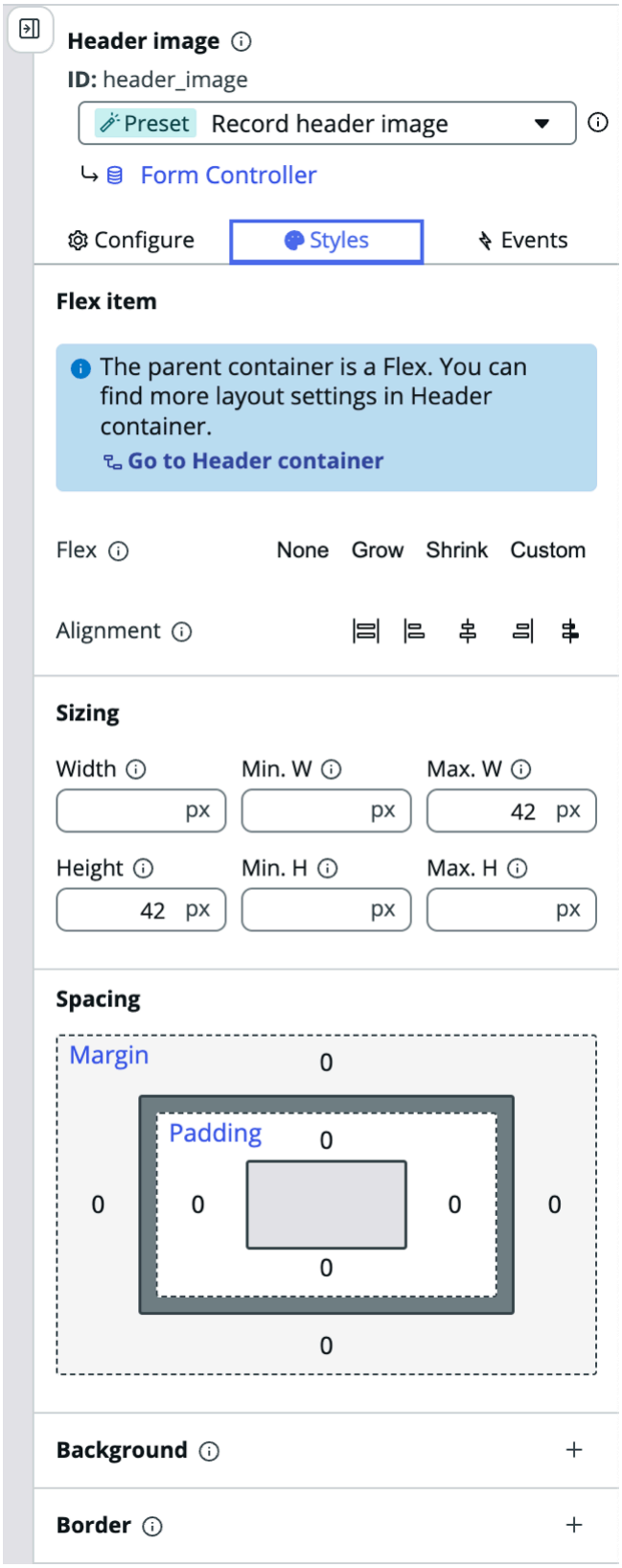
UI Builder Configuration panel

Use the configuration panel to work with components, including arranging, styling, and setting up event handlers.

- Select the **Config** tab to configure components. Each component has different configuration options that let you control the necessary parts of the component. For example, a button component is simple, and you can only configure its appearance, label, and a few properties. A list component is more complicated, and contains dozens of editable list parameters.

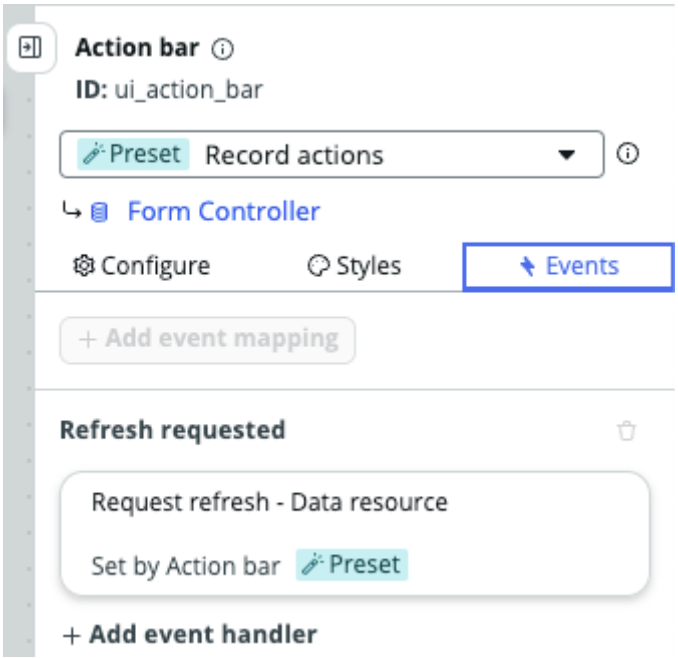


- Select the **Styles** tab in the configuration panel to add styling to a component. You can use standards-based Cascading Style Sheets (CSS) to change the visual style of a component. For example, add or change a background color, a border, or any other CSS style.



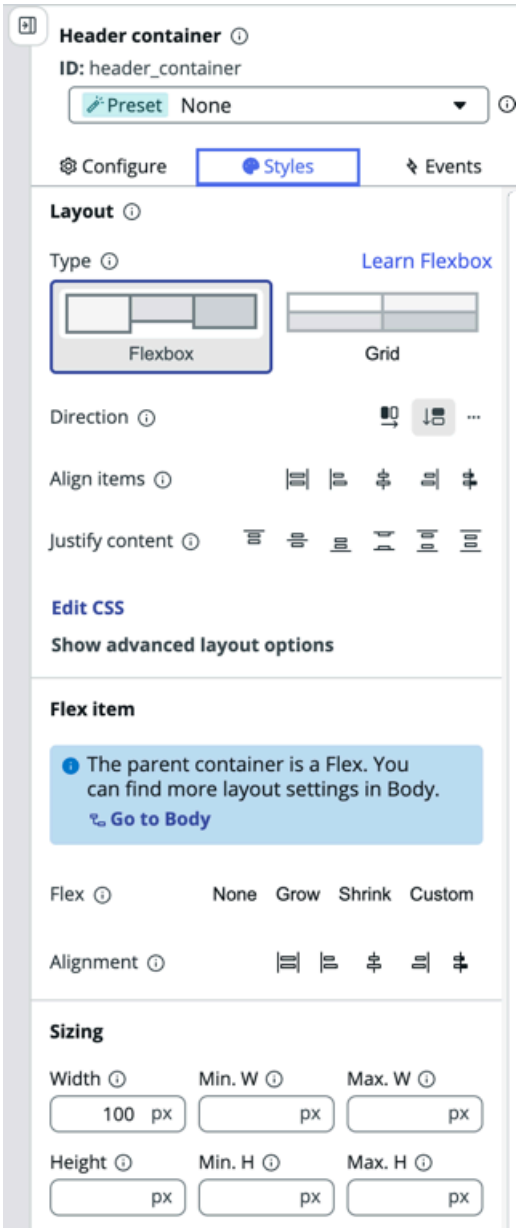
See Change the default appearance of components for more information.

- Select the **Events** tab to configure events that add actions to your components, pages, data resources, and declarative actions. When you add components to your UI Builder page, they are not configured to perform any action. For example, a button component is static and does not do anything until you bind an event action to it, such as deleting a record.



See Manage actions in UI Builder pages for more information.

- Column layouts govern how components are arranged on a page. When you add a column layout to a UI Builder page, you can configure how you want the layout designed. After you add a component to a column in the layout, you can configure the elements in more advanced ways. For example, you can justify the content, align items, and set the height, width, margins, background, borders, and padding of your column layout, columns, and components. CSS grid is the most powerful layout system. CSS Grid is built on top of a two-dimensional grid that gives you power over how you create your pages.



See Organize components in UI Builder pages for more information.

Open a UI Builder page to preview it

Preview a UI Builder page to see what the page looks like as a web page.

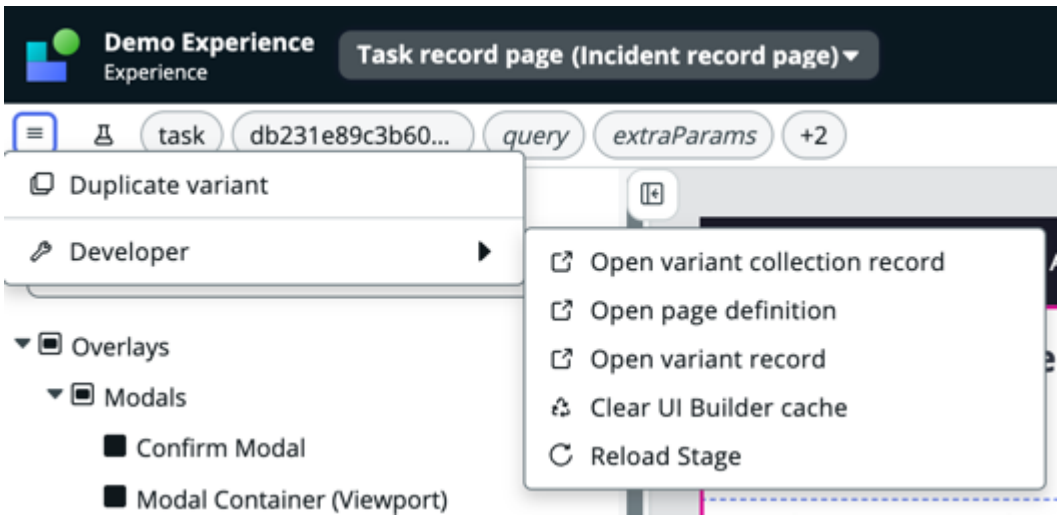
Select the Preview button to test the variant that you're currently working on, including modals, viewports, components, data resources, and dynamic data. Previewing a page variant is useful for seeing how it looks and functions while building an experience.

Another method of previewing a page is to request the URL path from the server. Use this method to test if a user sees the variant when they open the page. Select the drop-down arrow next to **Preview** and select **Open URL path** or select **Preview > Open URL path** Open URL path.

See Learn how to view and test your UI Builder experience for more information.

Developer editing in UI Builder

From the Menu, edit the UI Builder page as a developer on the platform. This option lets you change the platform-level details of your page and takes you out of UI Builder. You must have the proper role as a developer.



Create a page in UI Builder

Create a page in UI Builder for a portal, workspace, or custom application so that you can build a web experience for your users.

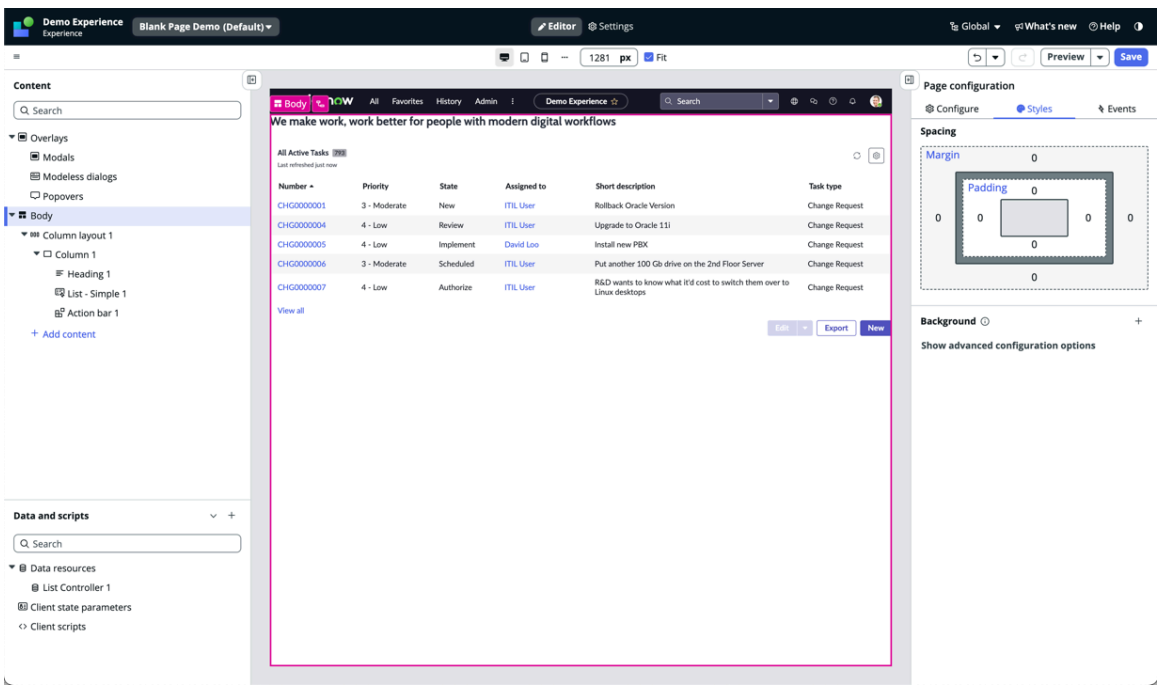
Before you begin

Role required: ui_builder_admin

About this task

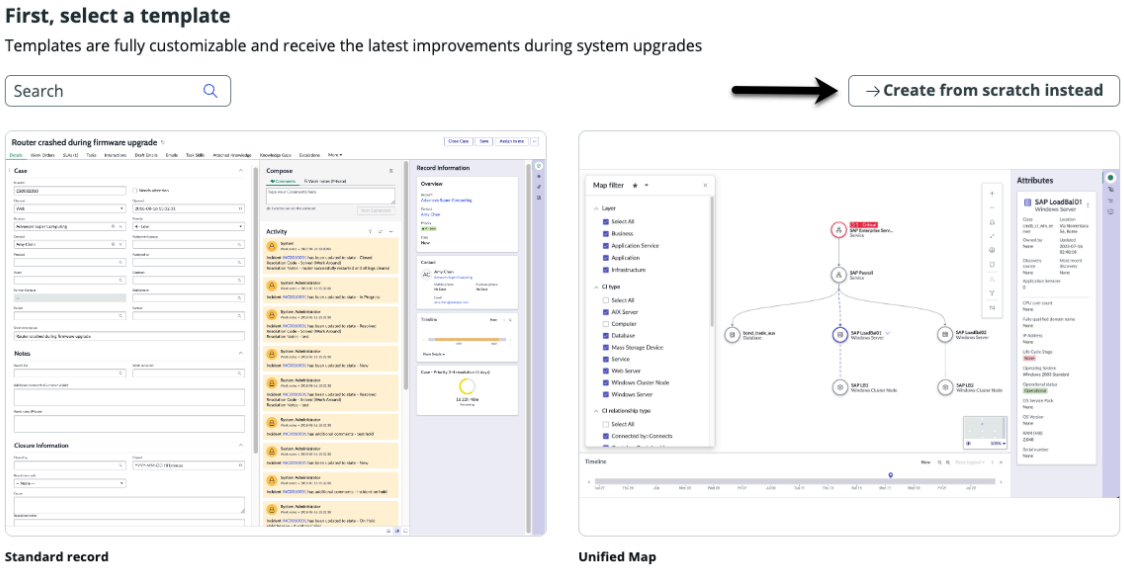
A page has a collection of components that make up a workspace, portal, or custom application user interface (UI).

UI Builder



Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Select the **Create new page** in the center of the screen.
4. Select **Create from scratch instead**.

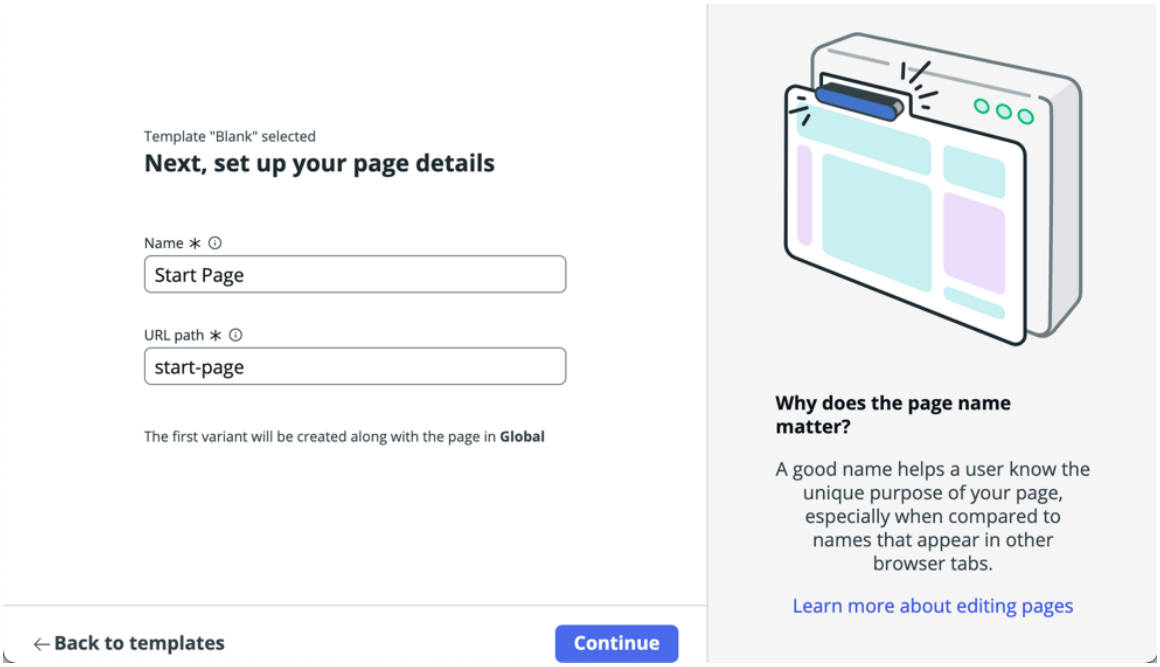


You can also create pages using page templates, see Create a page from a template for more information.

> **Note:**
> As of Xanadu Store Release 1, responsive authoring is available when creating a new UI Builder page from scratch. Responsive authoring is not available for existing pages or pages created with a template, however, these types of pages will continue to use the existing, default reflow model. For more information, see Responsive authoring.

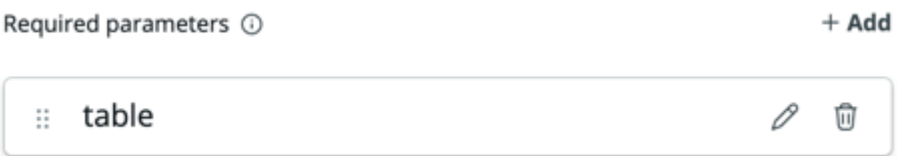
5. Enter a unique name for the page in the **Name** field.
6. Specify a path for your page in the **URL Path** field. UI Builder generates a default path based on the name that you provided in the previous step.

   A default path is added based on your page name. You can also create your own path. The path is required and must be unique. The path can include digits (0-9), letters (A-Z, a-z), and a few special characters (`" - "`, `" . "`, `" _ "`, `" ~ "`), with the words separated by a forward slash or hyphen. The **URL preview** shows the path of your page.



Note:
The application scope defaults to the scope that the user is currently in within the ServiceNow AI Platform®. For more information about the application scope, see Learn about security and roles.

7. Select **Continue**.
8. Add required parameters to your page URL.
   a. Click **+ Add**.
   b. Enter the required parameters for your page.
   A required parameter is a piece of data that your page requires, such as a sys_id, table, or query. Required parameters are useful for components, because they can bind to the value of the required parameter.

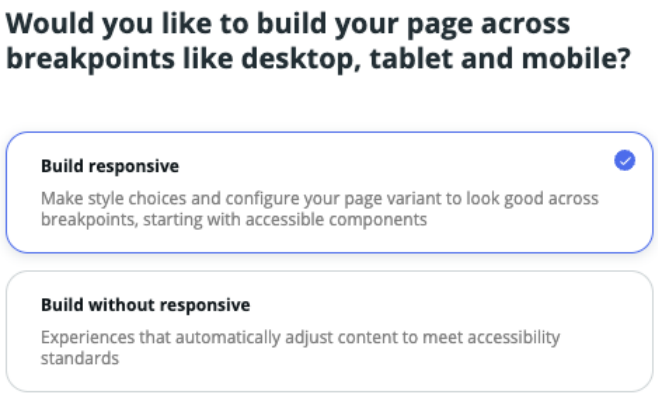


   For more information, see Manage UI Builder pages and page variants.
9. **Optional:** Add optional parameters to the URL of your page.
   a. Click **+ Add**.
   b. Enter the optional parameters for your page.
   Unlike required parameters, optional parameters are always name and value pairs that work no matter what order that they are provided in.

For more information, see Manage UI Builder pages and page variants.

10. Select **Looks good**.
11. Enter a name for the page variant.
    The form automatically contains **Default** but you can edit the page variant name.
12. **Optional:** Add one or more audiences for this page.
    If an audience you need is not listed, you can choose the Open audiences in platform link to create one.
13. **Optional:** Declare conditions for when to display the page by either using the provided dropdown menus or writing an encoded query string.

    (Optional) For more information on writing encoded queries, see Encoded query strings.
14. Select **Continue**.
15. Select the **Build responsive** option (default) for greater control of how the page appears at different screen form factors or select **Build without responsive** for automatic adjustment.



For more information, see Responsive authoring.

16. Select **Create** to create your blank page.
    The page opens in Editor view so you can start adding content such as layouts and components. For more information, see Customize UI Builder pages using components.

Create a page from a template

Use a template to create a page based on a pre-defined page template. A page template can help you create pages faster and keep pages consistent within an experience. Page templates may include components, data resources, and a layout.

This video shows you how to perform the following procedure. This video shows you how to create page from a template within a UI Builder experience.

Before you begin

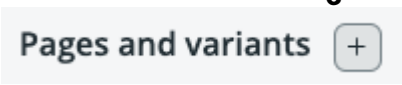
Role required: ui_builder_admin

About this task

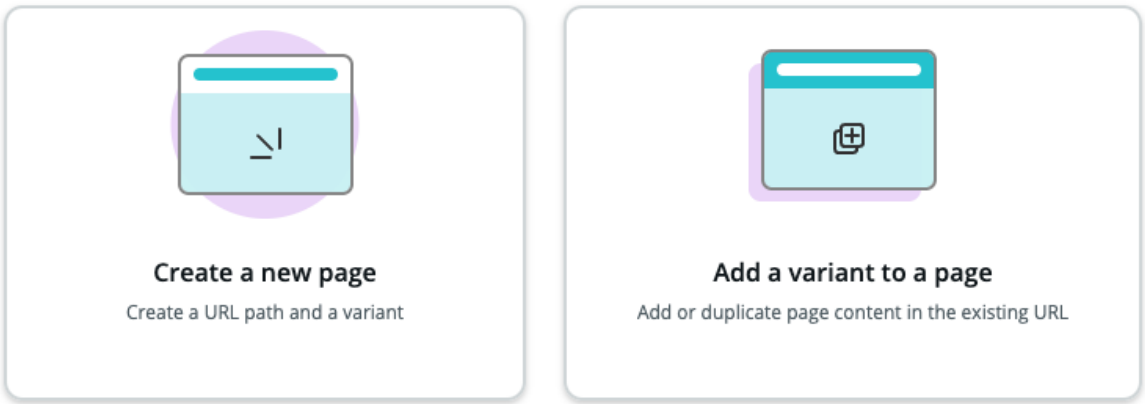
Select a page template when creating a page in your experience. After creating a page from a template, you can customize the page to your needs. Page templates include controllers that can be used with component presets. See Bind data to UI Builder pages using controllers (advanced feature) for more information.

Procedure

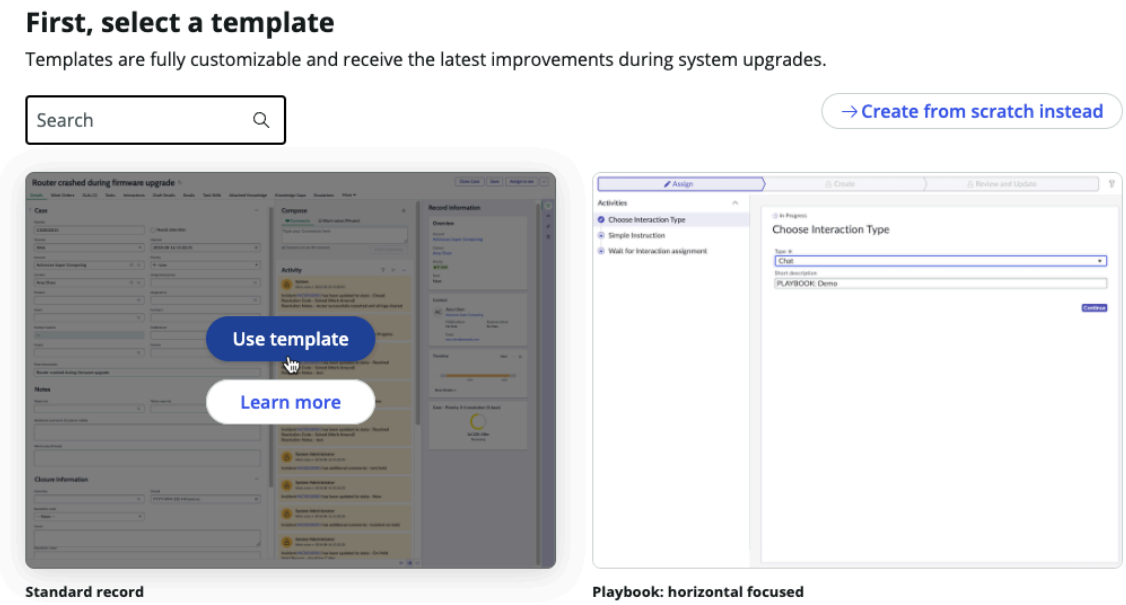
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Select the **+** icon in the **Pages and variants** section.



4. Select **Create a new page**.

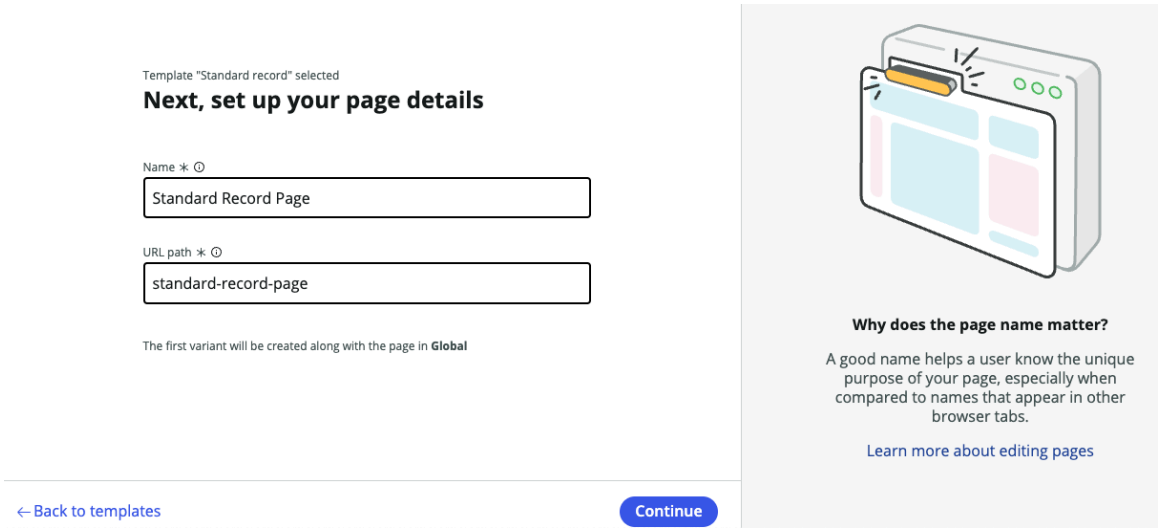


5. Select **Use template** when pointing to the template that you want to use.



6. Enter a unique name for the page in the **Name** field.
7. Specify a path for your page in the **Path** field. UI Builder generates a default path based on the name that you provided in the previous step.

A default path is added based on your page name. You can also create your own path. The path is required and must be unique. The path can include digits (0-9), letters (A-Z, a-z), and a few special characters (`" - "`, `" . "`, `" _ "`, `" ~ "`), with the words separated by a forward slash or hyphen. The **URL preview** shows the path of your page.



Note:
The application scope defaults to the scope that the user is in within the ServiceNow AI Platform®. For more information about the application scope, see Learn about security and roles.

8. Select **Continue**.
9. Review the page parameters included in the template.
   For more information about required and optional page parameters, see Adding Page Parameters in the ServiceNow Developer website.
10. Select **Looks good**.
11. Enter a name for the page variant.
    The first variant you create is named **Default** by default.
12. Add one or more audiences for this page.
    If an audience you need is not listed, you can choose the Open audiences in platform link to create one.
13. **Optional:** Declare conditions for when to display the page (this variant) by either using the provided dropdown menus or writing an encoded query string.

    (Optional) For more information on writing encoded queries, see Encoded query strings.
14. Select **Create** to create your page from template.
    The page you created displays in the **Page and variants** section of your experience. Select **Editor** to start adding components to your page. For more information, see Customize UI Builder pages using components.

Create a page from a legacy template

Use legacy page templates to reuse a page definition, such as record or list page, for pages in your workspace or portal.

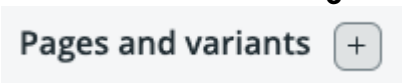
This video shows you how to perform the following procedure. This video shows you how to create a page from a legacy template in a UI Builder experience.

Before you begin

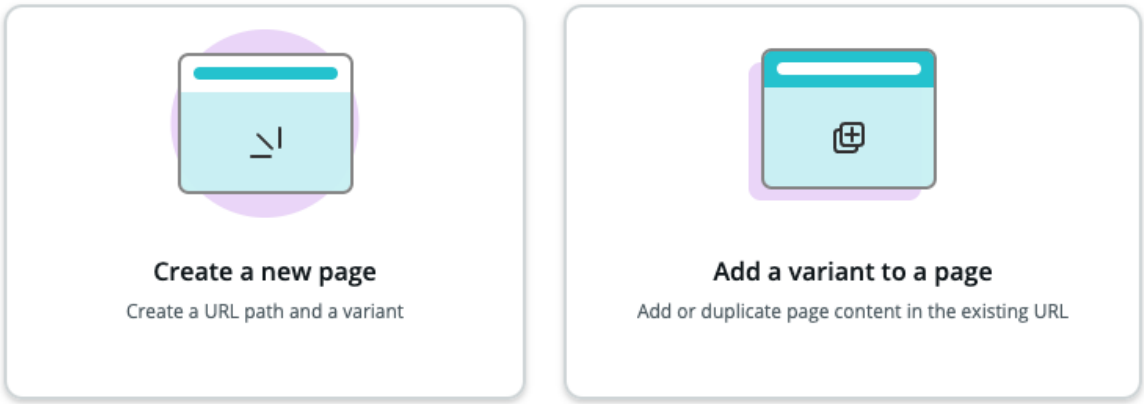
Role required: admin

Procedure

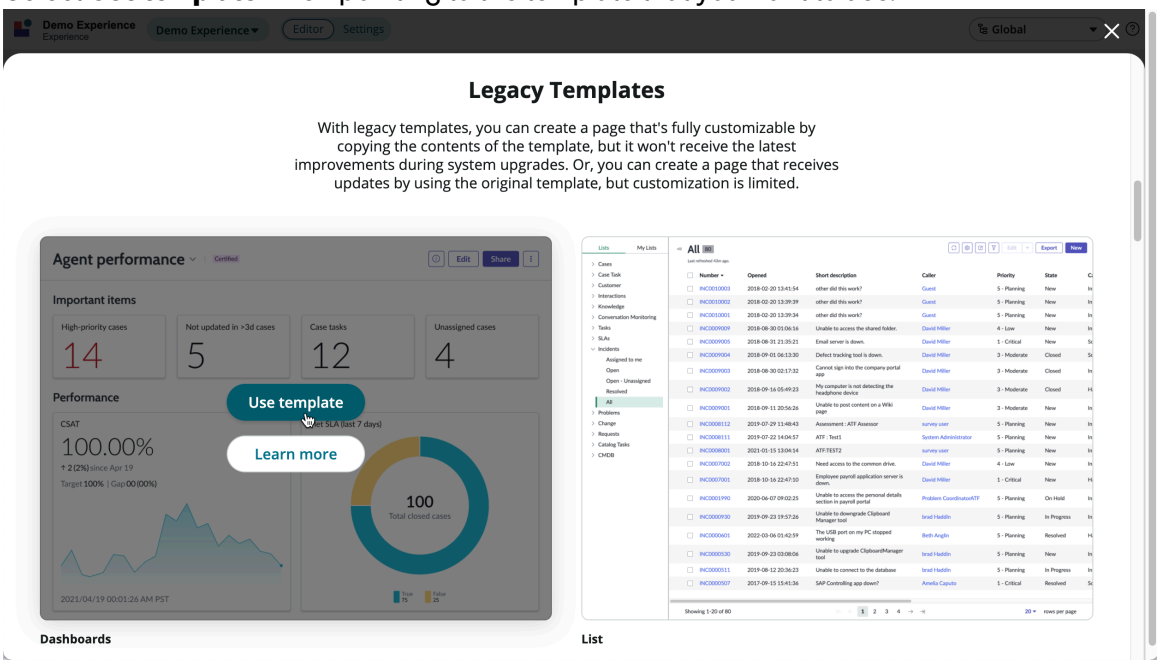
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Select the **+** icon in the **Pages and variants** section.



4. Select **Create a new page**.



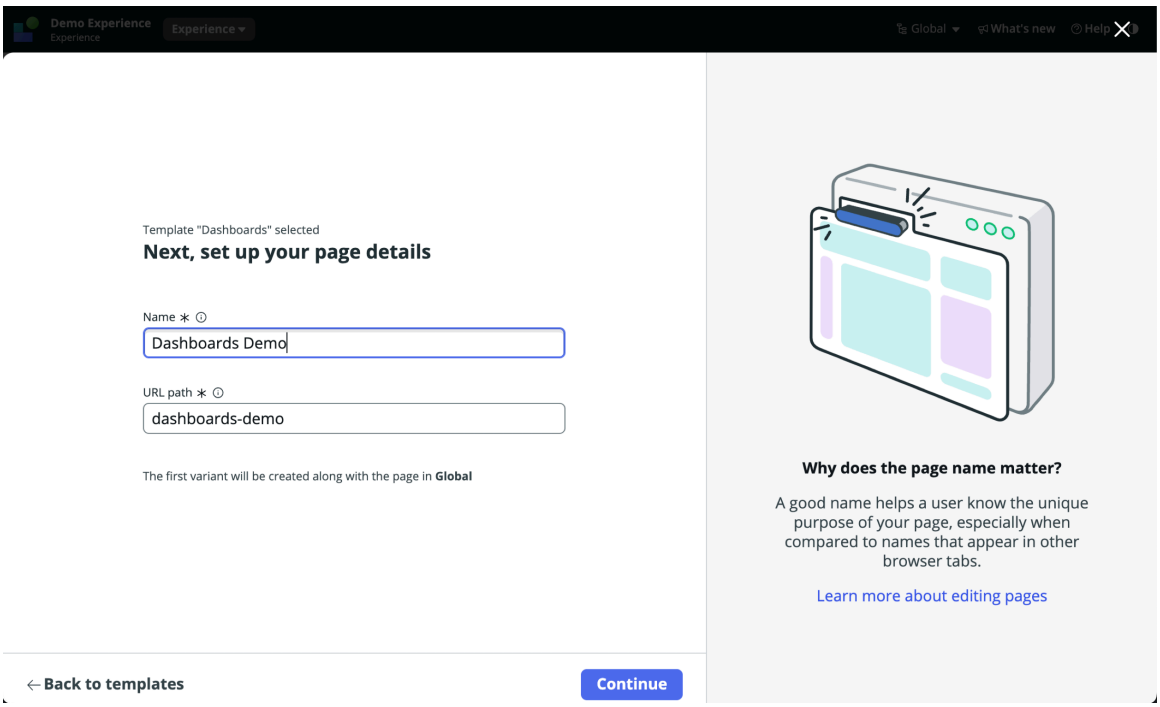
5. Select **Use template** when pointing to the template that you want to use.



6. Enter a unique name for the page in the **Name** field.
7. Specify a path for your page in the **Path** field. UI Builder generates a default path based on the name that you gave in the earlier step.

   A default path is added based on your page name. You can also create your own path. The path is required and must be unique. The path can include digits (0-9), letters (A-Z, a-z), and a

few special characters (`" - "`, `" . "`, `" _ "`, `" ~ "`), with the words separated by a forward slash or hyphen. The **URL preview** shows the path of your page.

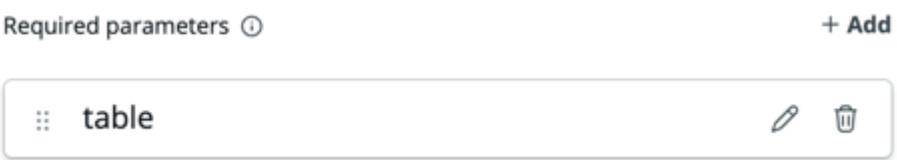


Note:
The application scope defaults to the scope that the user is in within the ServiceNow AI Platform®. For more information about the application scope, see Learn about security and roles.

8. Select how you want to use the base page template.
   Select **Use the original page (customization limited)** if you want to reference the template and its data. Select **Copy contents of the page (fully customized)** to copy the contents of the page template. When you reference a page template, your page automatically updates when you upgrade to a new release. If you copy the page template, your page will not update after upgrade.
9. Select **Continue**.
10. **Optional:** Add required parameters to your page URL.

    a. Click **+ Add**.

    b. Enter the required parameters for your page.
    A required parameter is a piece of data that your page requires, such as a sys_id, table, or query. Required parameters are useful for components, because they can bind to the value of the required parameter.



For more information about required and optional page parameters, see Adding Page Parameters in the ServiceNow Developer website.

11. **Optional:** Add optional pieces of data that you want to add to the URL of your page.

    a. Click **+ Add**.

    b. Enter the optional parameters for your page.
    Unlike required parameters, optional parameters are always name and value pairs that work no matter what order that they are provided in.

    For more information, see Manage UI Builder pages and page variants.

12. Select **Looks good**.
13. Enter a name for the page variant.
    The form automatically adds **Default** to the page name.
14. Add one or more audiences for this page.
    If an audience you need is not listed, you can choose the Open audiences in platform link to create one.
15. **Optional:** Declare conditions for when to display the page by either using the provided dropdown menus or writing an encoded query string.

    (Optional) For more information on writing encoded queries, see Encoded query strings.

16. Select **Create** to create your blank page.
    The page you created displays in the **Page and variants** section of your experience. Select **Editor** to start adding components to your page. For more information, see Customize UI Builder pages using components.

Edit a page

Edit a page to change the page name, path, and parameters.

Before you begin

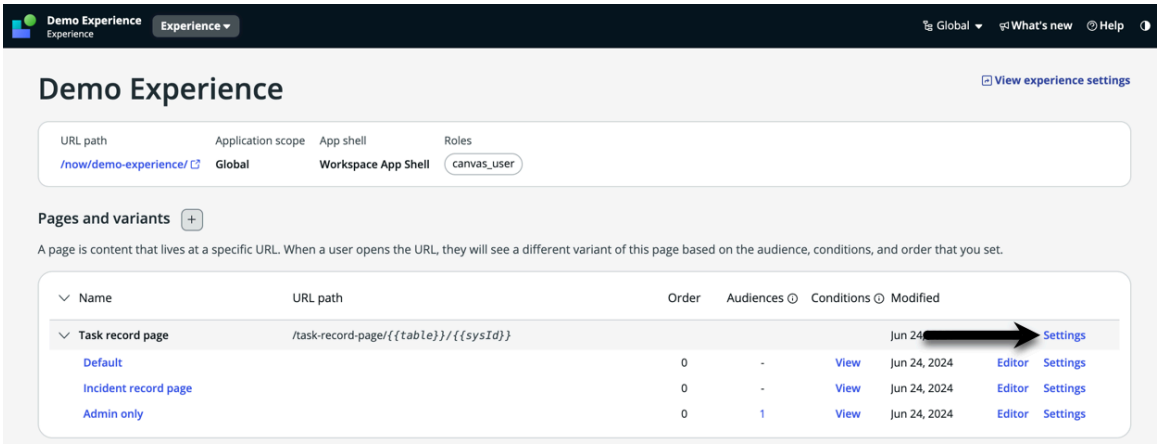
Role required: ui_builder_admin

About this task

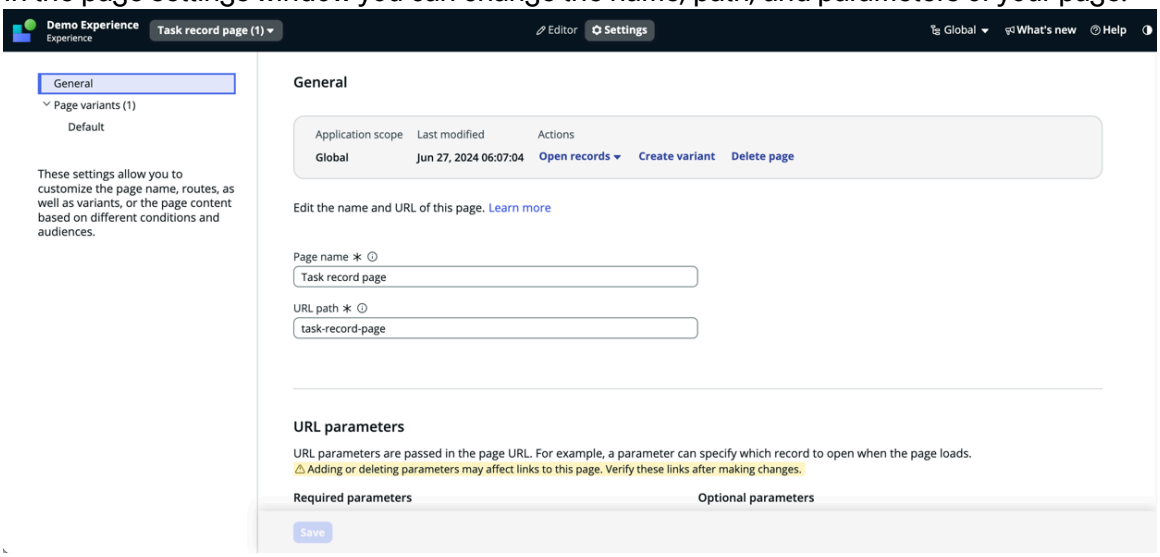
Edit page settings to change the name and path of your page. This is useful if you decide the name or path should be changed after you create the page. If your page is in a different application scope, you can choose to edit it in the original scope.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open the experience with the page that you want to edit.
3. Select **Settings** next to the page that you want to edit.



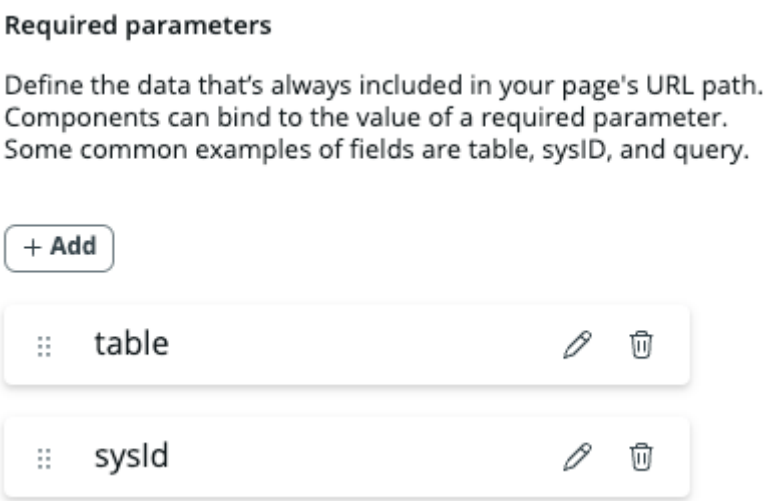
4. In the page settings window you can change the name, path, and parameters of your page.



5. Optional: Add required parameters to your page URL.

a. Click **+ Add**.

b. Enter the required parameters for your page.
A required parameter is a piece of data that your page requires, such as a sys_id, table, or query. Required parameters are useful for components, because they can bind to the value of the required parameter.



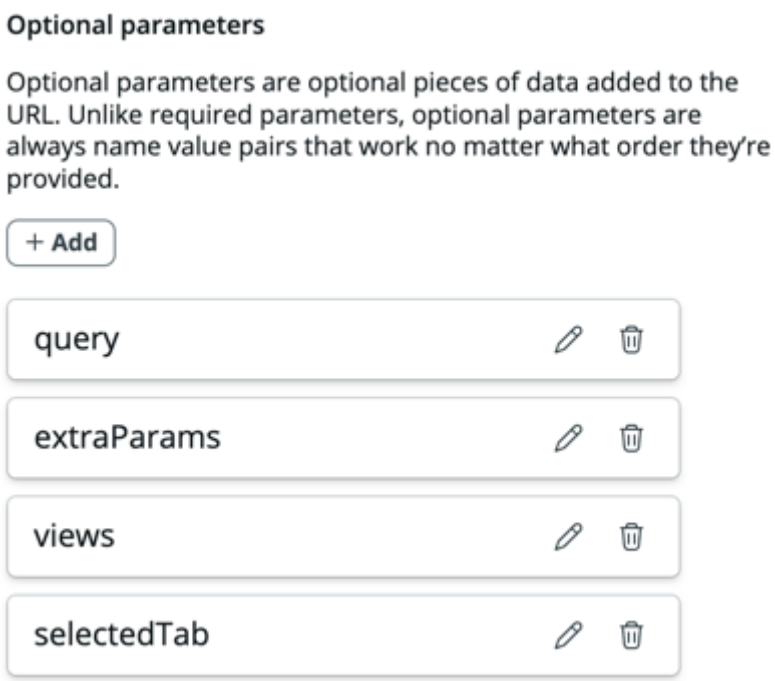
(Optional)

For more information, see Manage UI Builder pages and page variants.

6. **Optional:** Add optional pieces of data that you want to add to the URL of your page.

   a. Click **+ Add**.

   b. Enter the optional parameters for your page.
   Unlike required parameters, optional parameters are always name and value pairs that work no matter what order that they are provided in.



(Optional)

For more information, see Manage UI Builder pages and page variants.

7. Click **Save**.

Add an audience to your UI Builder page

Add one or more audiences to your page or page variant.

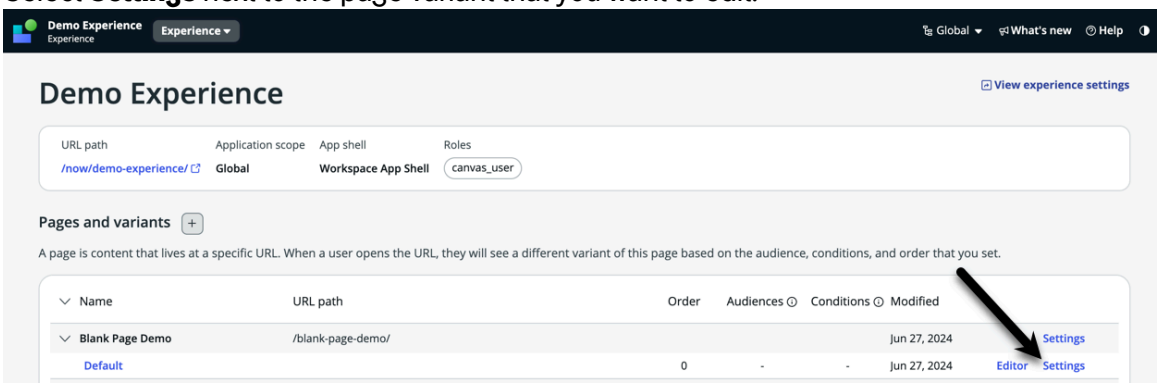
Before you begin

Role required: admin

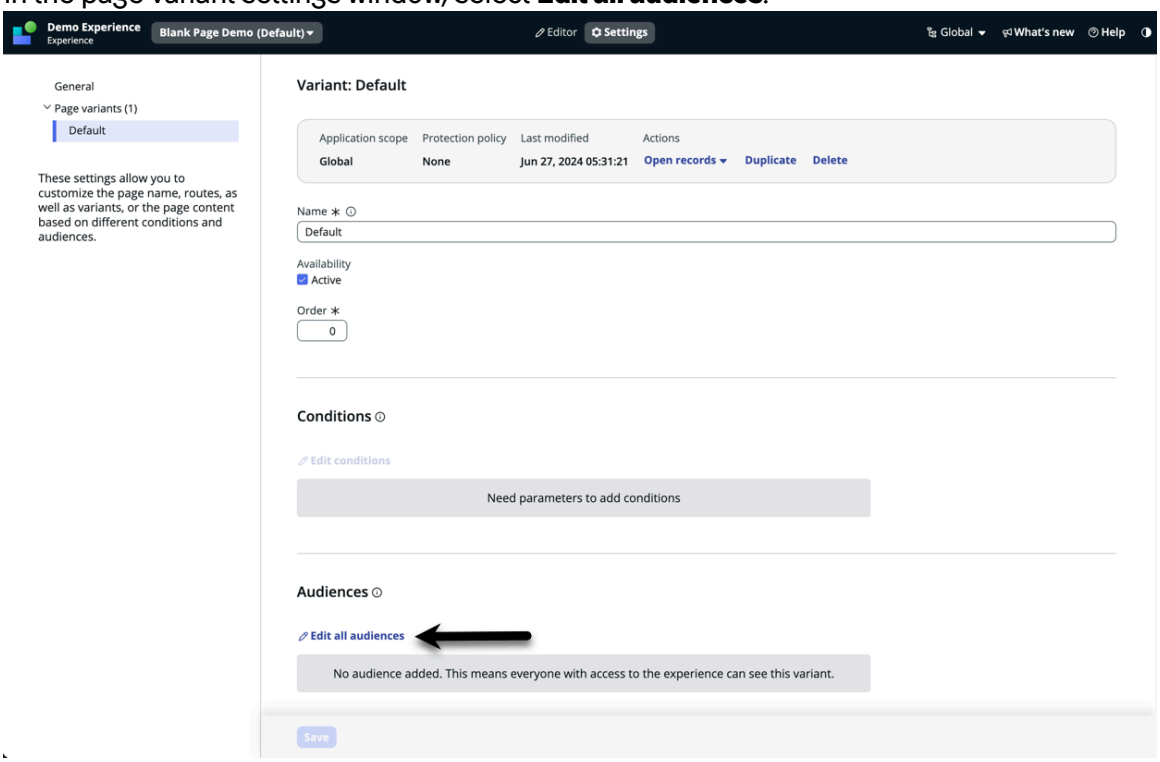
The *glide.ux.user_criteria_enabled* property needs to be set to **true** to configure access for users based on role, department, group, location, or company. See Enable the user criteria property, for more information.

Procedure

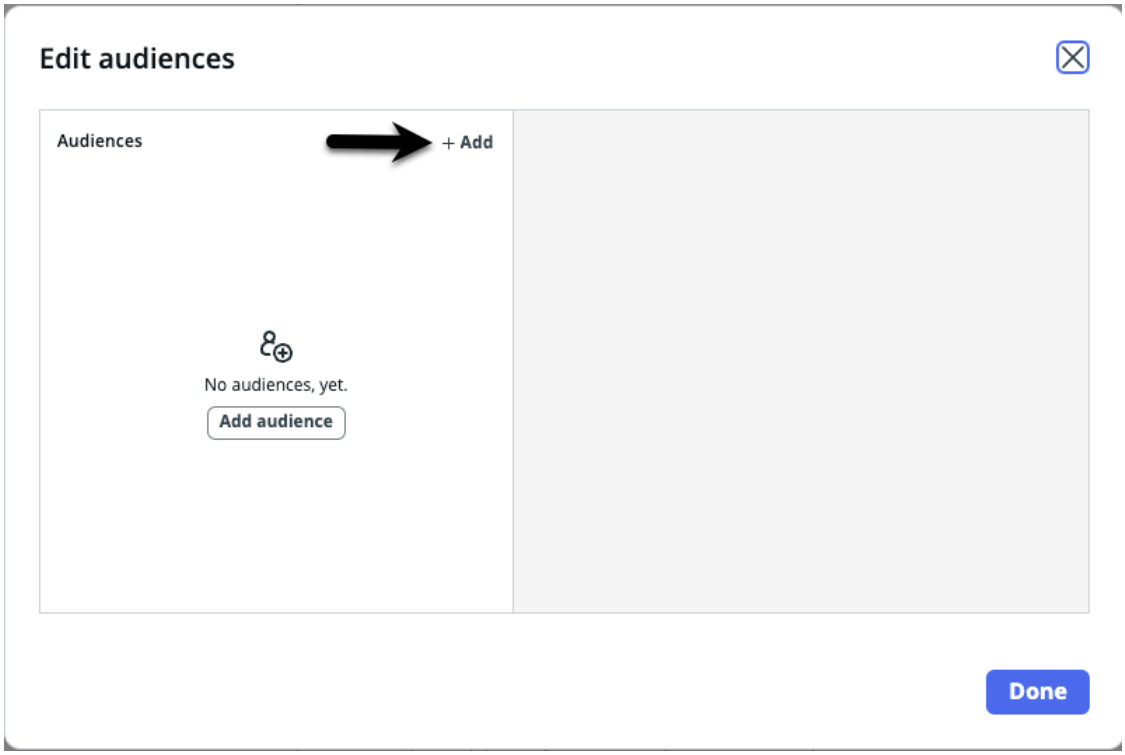
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open the experience with the page that you want to edit.
3. Select **Settings** next to the page variant that you want to edit.



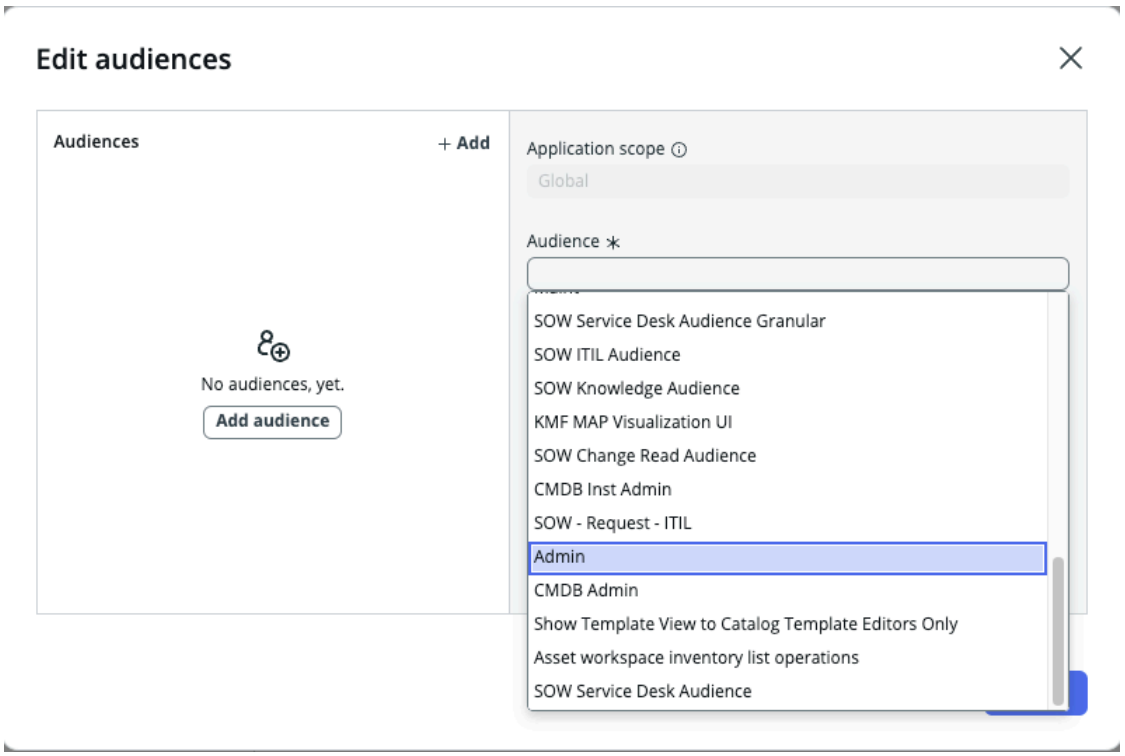
4. In the page variant settings window, select **Edit all audiences**.



5. Click **+ Add**.



6. Select an audience in the audience field.

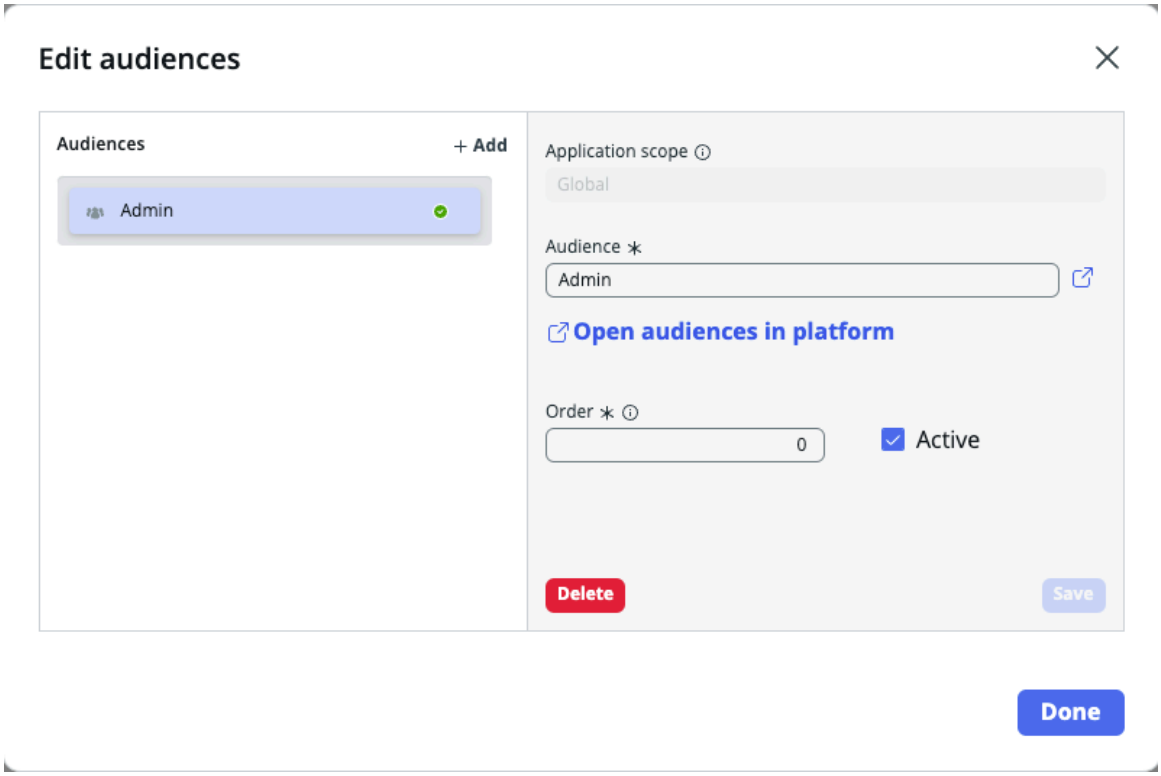


Note:
If an audience you need is not listed, you can choose the **Open audiences in platform** link to create one.

7. Set the order of the audiences.

Note:
To give higher priority to an audience, enter a lower number. If a user is part of multiple audiences, the audience with the highest priority is used.

8. Check **Active** if you want the audience to be able view the page.
9. Select **Save**.
   The audience displays in the **Audiences** list.



10. Select **Done**.

Enable the user criteria property

Enable the user criteria property to configure access for users based on role, department, group, location, or company in UI Builder.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **sys_properties.list**.
2. Select the *glide.ux.user_criteria_enabled* property.
3. In the **Value** field, specify **true**.
4. Click **Update**.

What to do next

Add audiences to your pages in UI Builder. For more information, see Add an audience to your UI Builder page.

Test values in a page

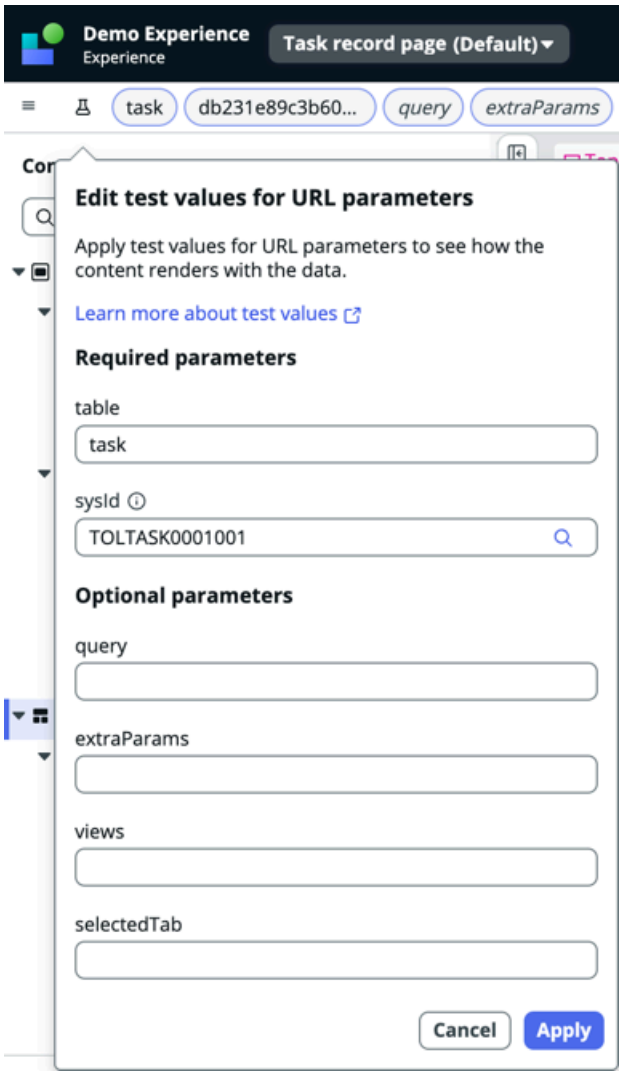
Add test values to your URL as a way to bring test data into a page.

Use test values to give values for required and optional URL parameters as a way to test your page with actual data.

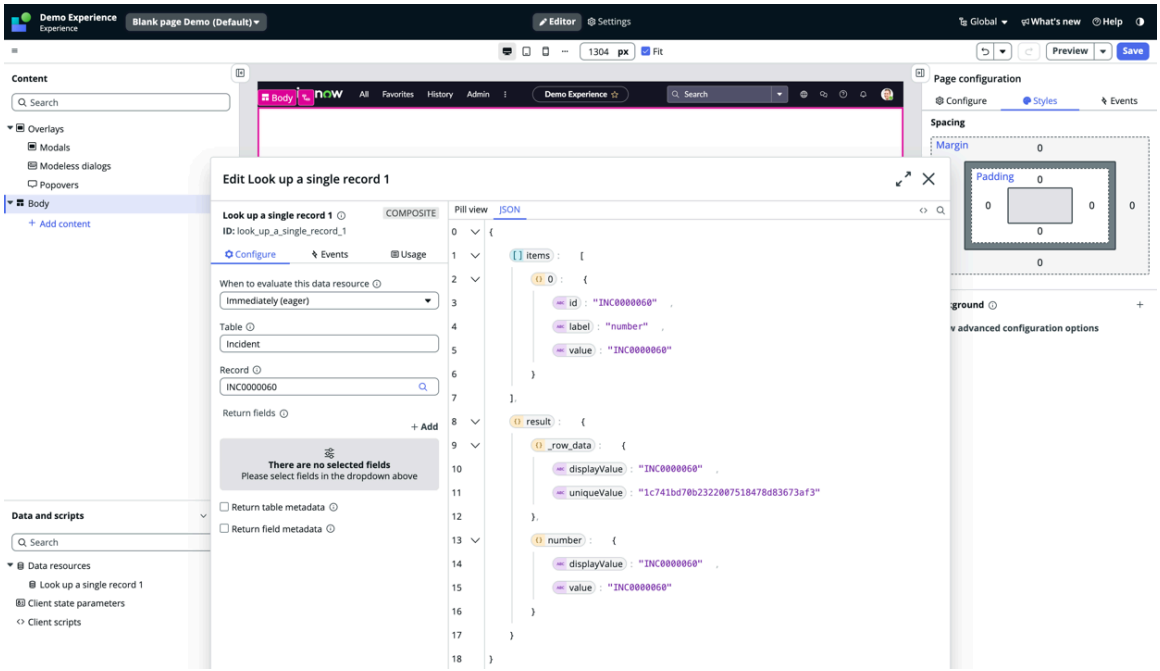
Test values are important because you can use them to simulate how your page behaves when the page gets its required or optional parameters from the URL. A UI Builder URL is not the same as the URL when you preview your page. So, UI Builder needs a way to simulate what happens to the page during different states of the preview URL.

For example, say that you're building a record page that displays a form for a single record. For the record page to load, the page must have a <table> and <sysId> in the URL so that it can get and display the proper record. In UI Builder, you must supply test values for the table and sys_id so that you can see how the page looks when you preview the page.

To get test values to show data, add a data resource, then configure the data resource to bind a record to the test value in the URL. For example, you could add `incident` as a test value.



Then add a data resource named **Look Up Record**. In the **Table field**, dynamically bind the incident test value to a `context.props.table` table, as shown in the following image.



Create a page variant

A page variant in UI Builder is a variation of a page that exists at the same path that targets different audiences using user criteria.

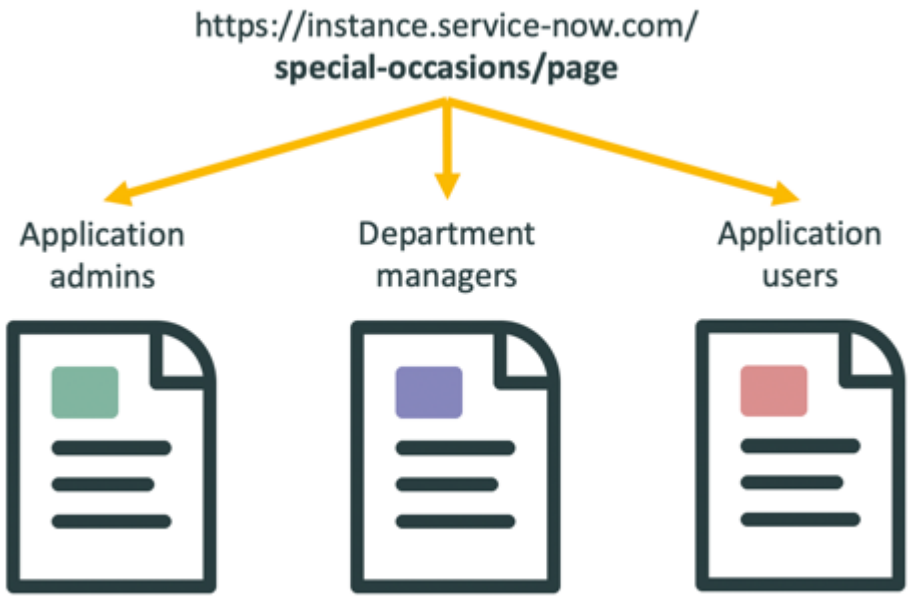
This video show you how to perform the following procedure. This video shows you how to create a page variant.

Before you begin

Role required: ui_builder_admin

About this task

Learn how to create a page variant in UI Builder. A page variant is based on your page with content targeted for a different audience. For example, you can create a homepage for agents, and a variant of that page for managers of those agents. The variant exists at the same URL.



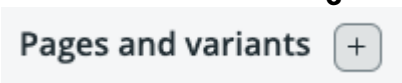
- When someone who is part of the application admins audience accesses the page URL, the user sees a version of the page that has application admin-specific details, such as application metrics and usage details. For the Special Occasions application the page would show how many users have birthdays and work anniversaries and how many do not. The page would also show how many issues have been reported with special occasions.

- When someone who is part of the department managers audience accesses the page URL, the user sees department-related information for the application, such as people in their department with upcoming birthdays and work anniversaries, and whether or not those people are okay with their special occasions being shared.

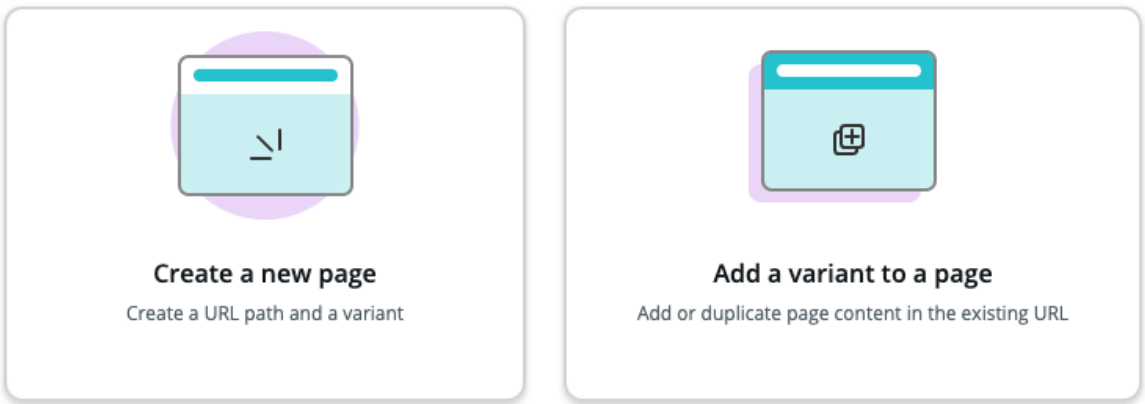
- When someone who is part of the application users audience accesses the page URL, the user sees a user-specific version of the page with their special occasions that allows them to configure whether or not they want others to see their special occasions.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open the experience with the page for which you want to create a variant.
3. Select the **+** icon in the **Pages and variants** section.



4. Select **Add a variant to a page**.



5. Click **+ Add variant**.

6. Click **Build from scratch instead** or select **Use template** when pointing to the template that you want to use.
7. Enter a name for the variant page.

   The name can be similar to your page name, or what ever you want it to be.

Note:
The application scope defaults to the scope that the user is in within the ServiceNow AI Platform®. For more information about application scope, see Learn about security and roles.

8. Specify one or more audiences for your variant page by selecting **Add an audience**. The audience is important because it lets you assign your page variant to a specific role or group. If an audience you need is not listed, create one by selecting **View all available audiences**, and then selecting **New**.
9. Declare conditions for when to display the page by either using the provided dropdown menus or writing an encoded query string.

   To learn more about declaring conditions, see Control the conditions for a page variant. For more information on writing encoded queries, see Encoded query strings.
10. If creating a page from scratch, select **Continue** and then select the **Build responsive** option (default).
11. Click **Create** to create your page variant.

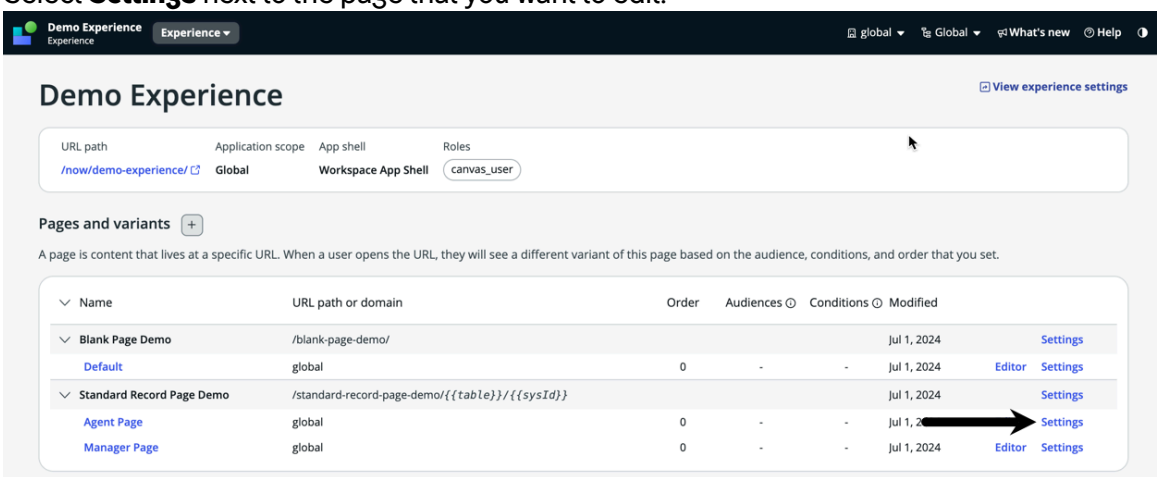
Edit page variant settings

Edit page variant settings to add additional criteria to determine when the page variant displays to users.

Before you begin
Role required: admin

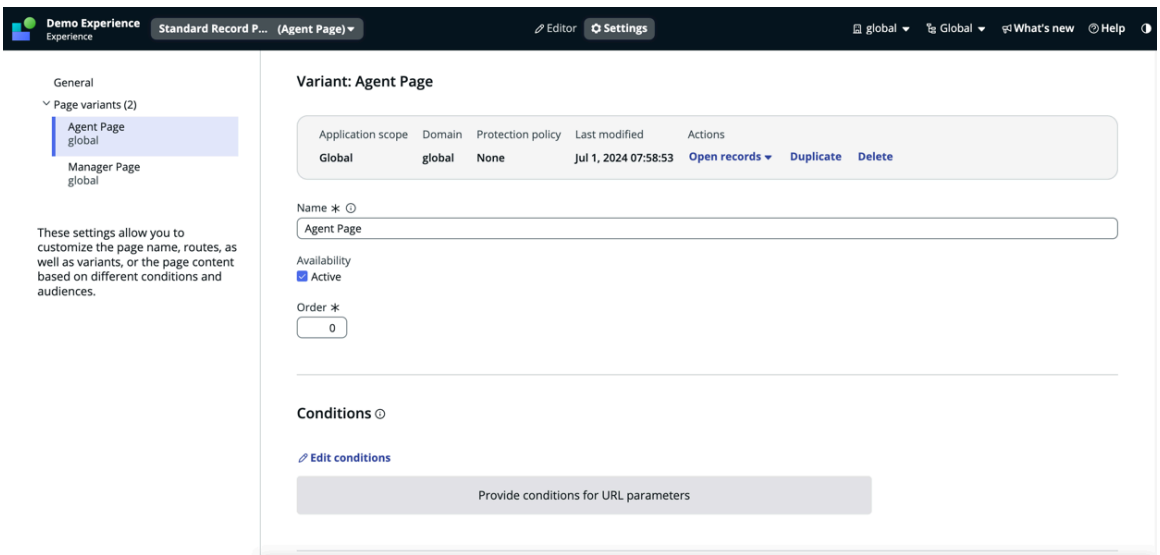
About this task
Customize page variant settings to set who sees the page variant. You can add and remove audiences as well as edit conditions to determine when a page variant is shown. Conditions can only evaluate the listed parameters. Conditions are based on setting an order and adding a query string that sets the criteria that must be met for the page variant to display. If you have multiple page variants that all have the same conditions, then the variants go by the order setting. The following task shows how to set the variant conditions and order.

Procedure
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open the experience with the page variant that you want to edit.
3. Select **Settings** next to the page that you want to edit.



4. On the form, fill in the fields.

Field	Description
Name	Enter a name for the page variant.
Availability	Select **Active** to make the page variant viewable to the selected audience.
Order	Add an order for the condition in the **Order** field. You may want certain conditions to have a higher priority than others. The lower the number, the higher the priority.
Conditions	Set the rules for when your pages are shown. For more information, see Control the conditions for a page variant.
Audiences	Add or remove audiences for the page variant. For more information, see Learn about audiences.



5. Click **Save**.

Control the conditions for a page variant

Set the conditions and manage the criteria that determine when a page variant is displayed. UI Builder page variants enable you to create different versions of a page to tailor content for a specific audience.

Before you begin

Role required: ui_builder_admin

About this task

If you have multiple page variants that all have the same conditions, the variants go by the order setting.

Procedure

1. Create a page variant.
   For more information, see Create a page variant.
2. Set the conditions for a new variant.

In this example, you configure a page variant to display when you open a record from the Task table.

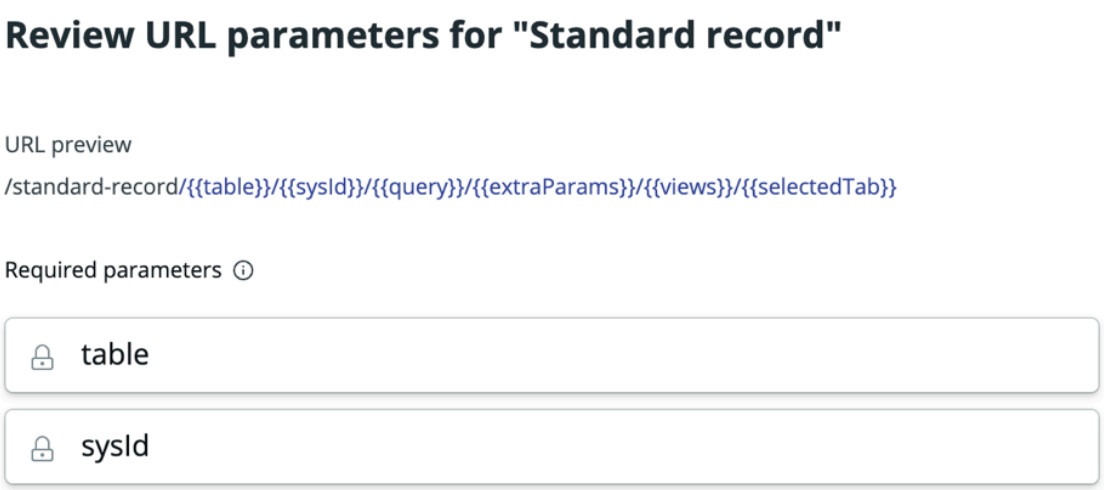
a. On the **First, select a template** screen, locate the **Standard record** template and select **Use template**.

> **Note:**
> Conditions can only evaluate parameters that have been added. In this example, the **Standard record** template includes the *table* and *sys_id* parameters.

b. Enter a name for your page and select **Continue**.

c. Review the URL parameters and select **Looks good**.

Notice that *table* and *sysId* are added as required parameters, which appear in the URL preview.



d. On the **Tell us about your variant** screen, enter a name for your variant.

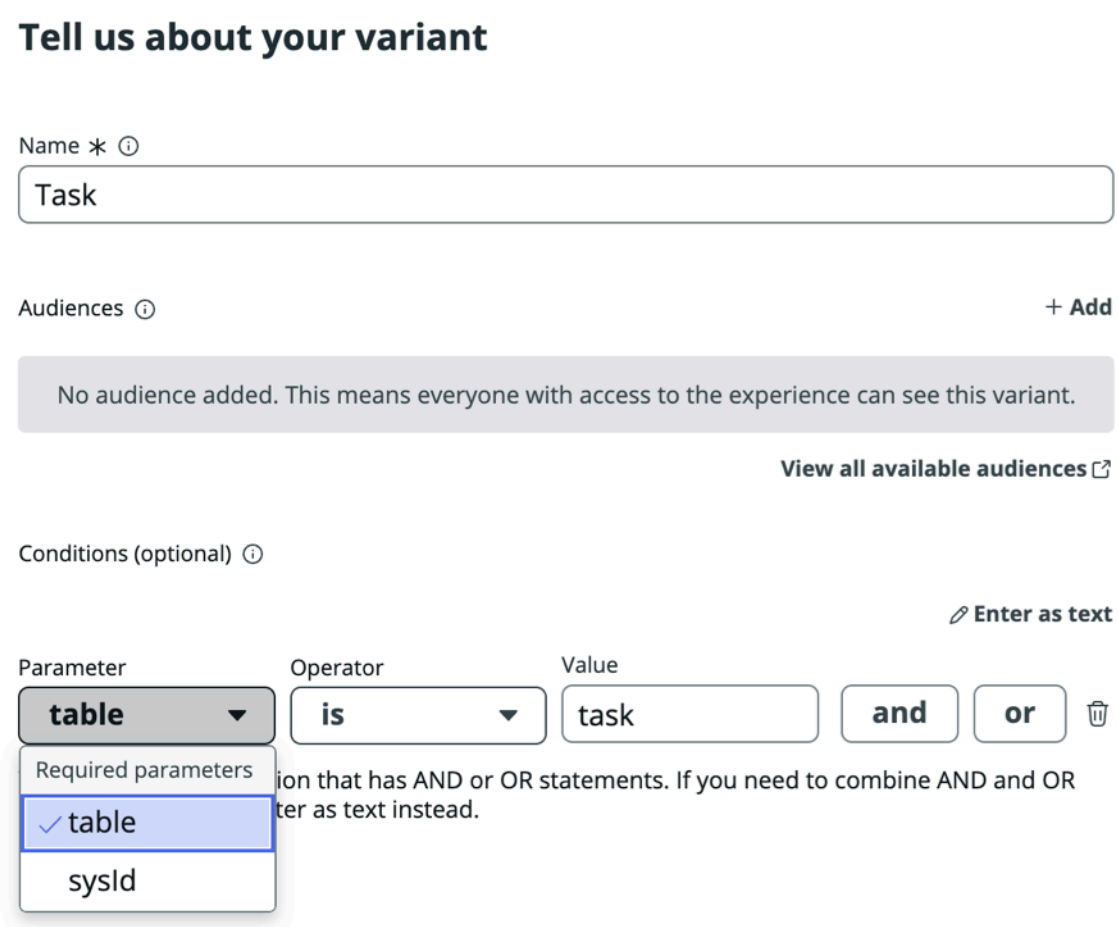
e. In the form section, fill out the fields.

Variant condition form section

Field	Description	Example
Parameter	The aspect or attribute that you want to check.	*table*, *sys_id* **Note:** Conditions can only evaluate the added parameters. However, conditions written for subpages inherit additional outputs from the parent page controllers, providing a wider range of parameters options.

Field	Description	Example
Operator	The rule that compares the parameter to the value. It tells you how the comparison is made.	**is**, **is not**, **starts with**
Value	The number, text, or option input that you want to compare with the parameter.	`"Incident"`, `"12345"`, `"Active"`, `"Change Request"`

Note:
The *`table`* and *`sysId`* parameters are available options for the **Parameter** field because they're required parameters as a part of the **Standard record** template.



f. Select **Create**.

3. Review the conditions created for your variant.

a. In the Experience view, find the variant for which you want to view the conditions.

b. Select the **View** button to display the conditions set for that variant.

4. Edit the existing conditions for your variant.

a. From the Experience view, find the variant for which you want to edit the conditions.

b. Select **Settings**.

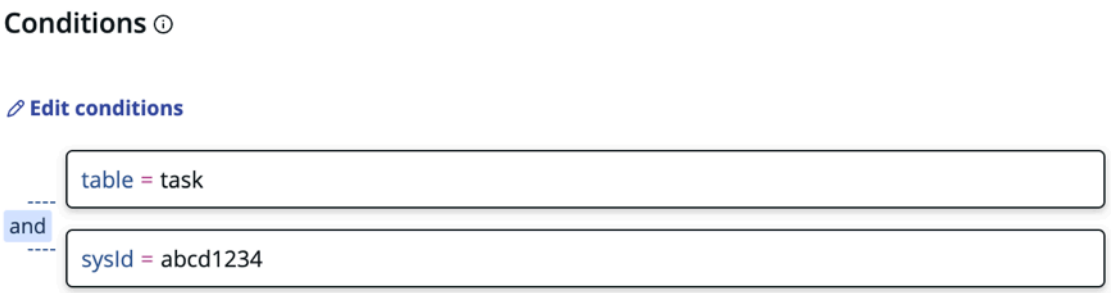
c. Under Conditions, select **Edit conditions**.

The condition you previously set appears. You can update these fields.

d. On the **Edit variant conditions** screen, next to the condition, select the **and**(and) or **or**(or) button to add another condition and specify the criteria.

For example, by adding the following **AND** condition (**[sysId] [is]** `[abcd1234]`), you configure the page variant to display when you access a record from the Task table with a sys_id of abcd1234.

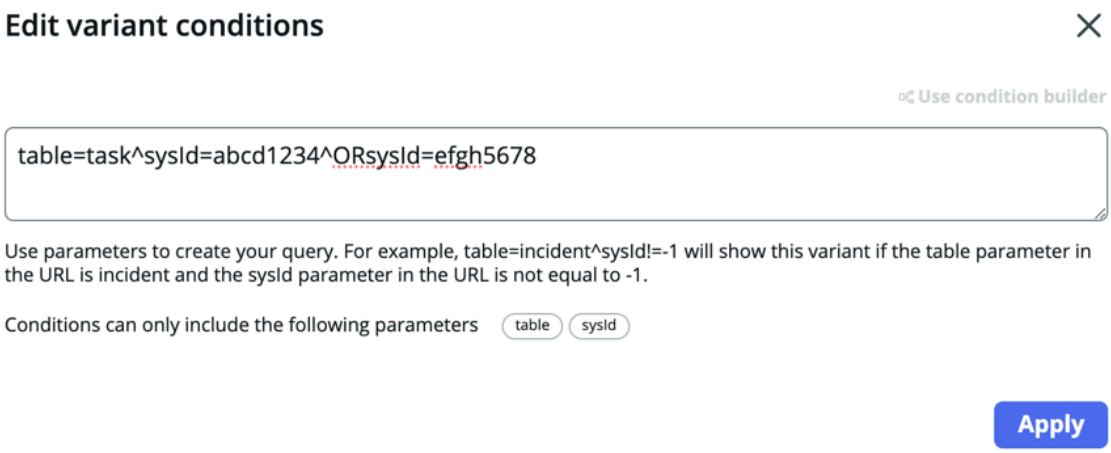
Note:
You can add **AND** or **OR** conditions. To mix both types, you must write an encoded query.



e. Optional: Select **Enter as text** to write an encoded query that specifies the criteria.

(Optional) The **Edit variant conditions** screen displays the encoded query field, where the conditions you previously set are shown in encoded query form: `table=task^sysId=abcd1234`.

For example, by adding the following 'OR' condition (`^ORsysId=efgh5678`), you set the page variant to display when you access a record from the Task table with a sys_id of either abcd1234 or efgh5678. For more information on writing encoded queries, see Encoded query strings.



Responsive authoring

Use responsive authoring to create UI Builder pages that adjust smoothly to different form factors (sizes), such as desktop, tablet, and mobile.

Responsive authoring in UI Builder

Responsive authoring enables you to create pages that look good and function properly at any form factor, making it easier for people to interact with the content. For example, a page containing three columns when viewed on a laptop, can adjust to a single column for smaller screens.

UI Builder currently offers three default form factors:

- Desktop (1281 pixels to infinity)
- Tablet (1280 pixels and smaller)
- Mobile (500 pixels to zero)

In addition to these form factors, you can create up to three additional custom breakpoints (widths). For more information, see Create a breakpoint for responsive authoring.

There are different techniques for editing pages so they're usable at different form factors. Use any of the following options:

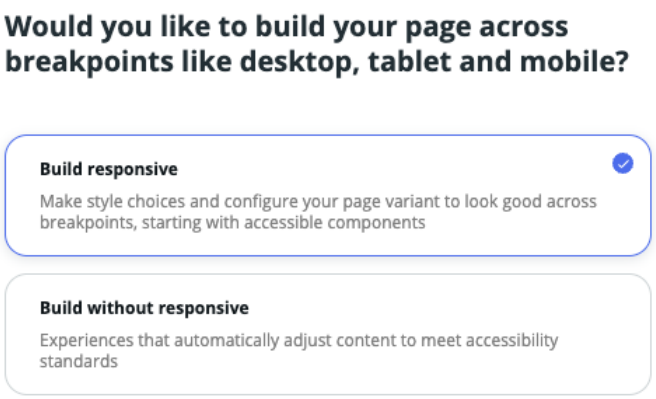
- Show or hide components
- Change component configuration
- Edit styles
- Rearrange the layout

Responsive authoring and reflow

As of Xanadu Store Release 1, responsive authoring is only available when creating UI Builder pages from scratch. Responsive authoring isn't available for existing pages or pages created with a template. These types of pages continue to use the existing, default reflow model to adjust pages for different screen widths.

Reflow transforms page layouts into a vertical, stacked view automatically without loss of content or functionality when users increase browser zoom to 400%. This adjustment helps users with low vision or who have trouble seeing web content in a browser due to monitor size, device type, poor lighting, or other situations.

All existing pages that were created with templates or from scratch use reflow automatically. Now, when creating pages from scratch in UI Builder, an extra step shows you that responsive authoring is selected by default. You can choose to use reflow instead by selecting **Build without responsive**, however, using the responsive authoring option gives you more control over how pages work and look at different form factors. For more information about creating pages in UI Builder, see Create a page in UI Builder.



For more information about reflow, see Reflow for Configurable Workspace.

For a list of UI Builder components that support reflow, see the **Support for reflow** section of the Next Experience Components release notes.

Application and cascading of changes

All changes made to a form factor are applied to all smaller form factors automatically. For example, any changes made to tablet apply to both tablet and mobile. However, if you then make changes to the mobile form factor, it overrides the cascaded settings from the tablet form factor. This cascading functionality enables you to make specific changes for each smaller form factor so the page looks and functions well.

Note:
Changes don't cascade up to larger sizes. For example, changes made to tablet aren't applied to the desktop.

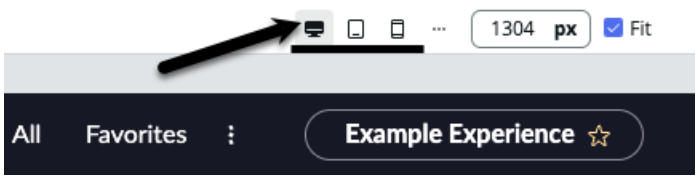
Responsive authoring and controllers

If you add a controller to a page, the controller properties are global and cannot be set per form factor. For more information about controllers, see Bind data to UI Builder pages using controllers (advanced feature).

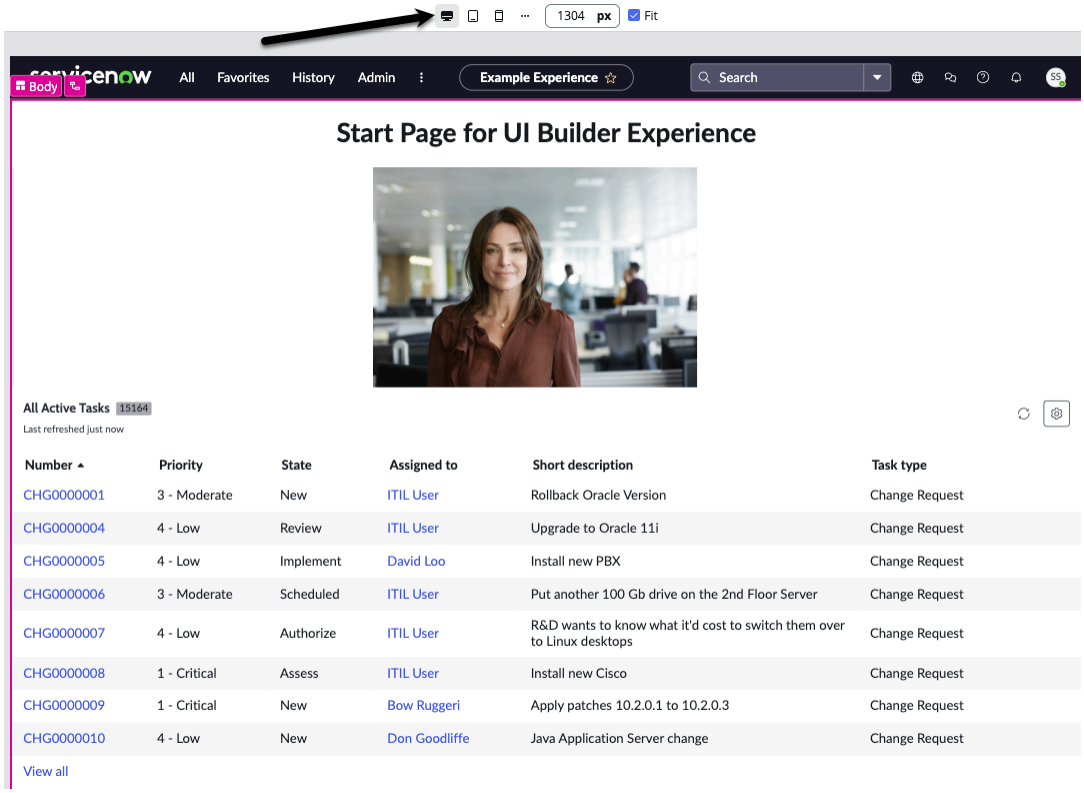
Using responsive authoring as you create pages

As you build pages, create tailored designs to control the look and feel for different form factors.

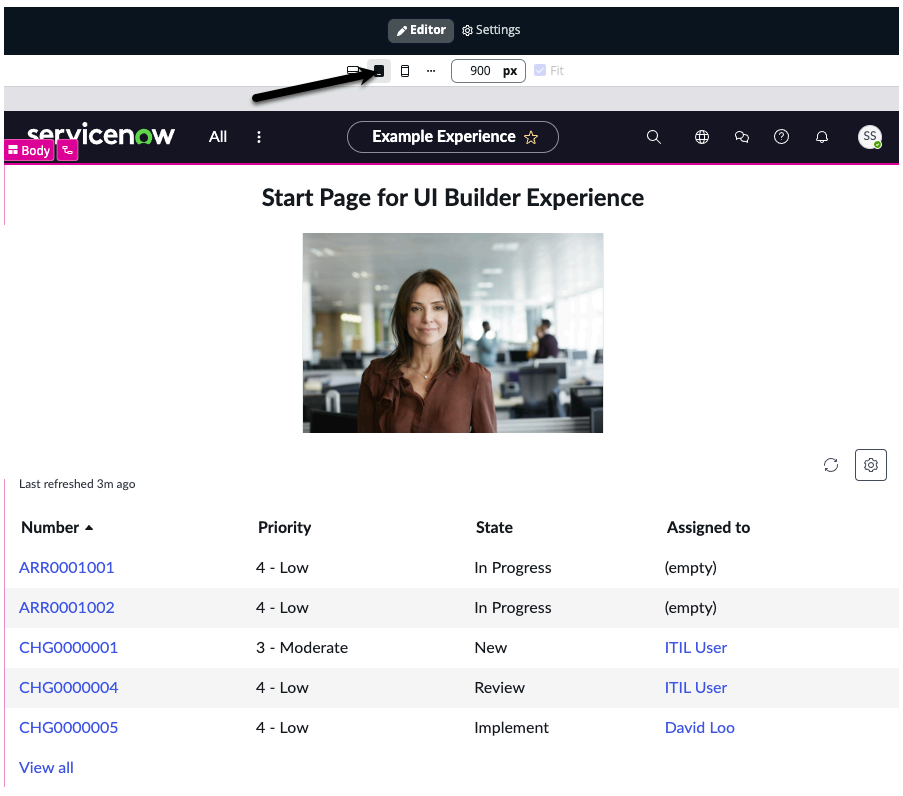
By default on the stage, you build pages for the desktop (1281 pixels to infinity). At any time, select another icon to see the page on the stage at a different form factor, such as tablet or mobile. As needed, edit the page so it displays appropriately at the new form factor.



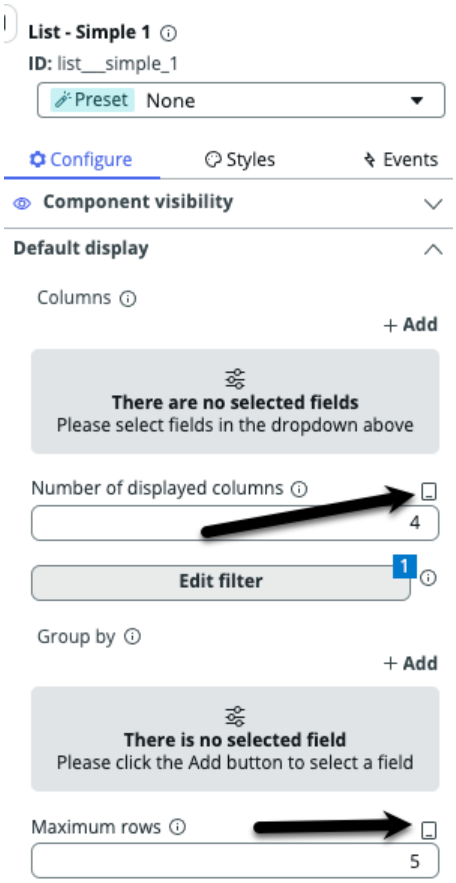
Here's what a sample page looks like at the desktop form factor.



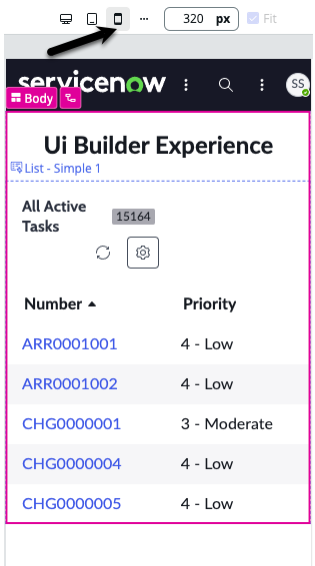
Here's the same page customized for the tablet form factor.



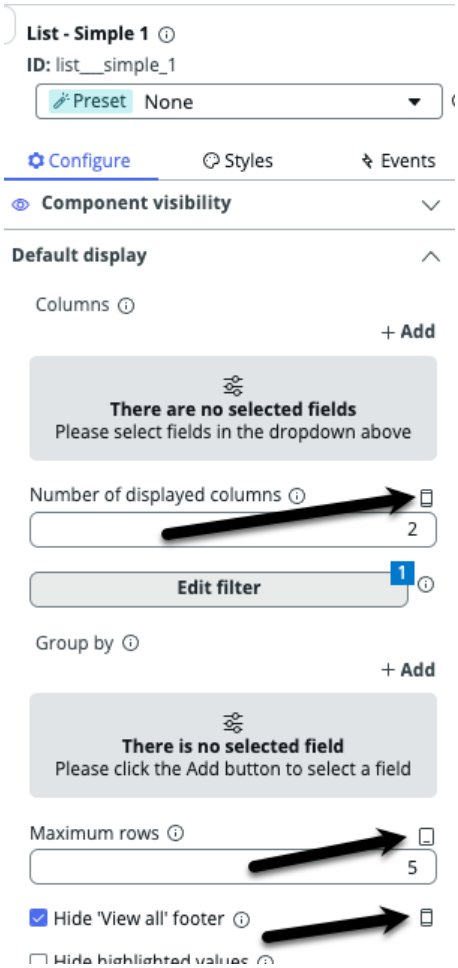
For tablet, the heading and image components have been reduced in size. The simple list component configuration was edited to show four columns instead of six and the maximum number of rows was set to 5. When editing a component for a form factor, the fields that have changed are marked with a cascading icon for that form factor (in this case, tablet). The icon enables you to determine the differences between the form factors for a given page easily.



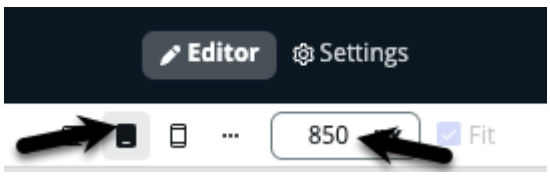
Here's the same page customized for the mobile form factor.



The number of words in the heading was reduced and the image component was hidden. The simple list component configuration was edited to show two columns instead of four and the option to hide the "view all" footer option was selected. The two fields customized specifically for mobile have a mobile cascading icon next to them. The **Maximum rows** field still retains the tablet cascading icon because that field is inheriting the tablet setting and wasn't customized for mobile.



In addition to the default form factors in the editor view, at any time you can manually enter a number in the pixel field. The stage adjusts to the nearest form factor based on range. For example, entering 850 pixels adjusts the stage to the tablet form factor because its range is defined as 1280 to 501 pixels.



Note:
Form factors and the ability to enter a pixel width are also available when previewing a page. For more information, see Learn how to view and test your UI Builder experience.

Adjusting component visibility in responsive authoring

When creating pages with responsive authoring for different form factors, learn how to show or hide components by completing steps to hide an image at mobile size.

Before you begin

Role required: ui_builder_admin

About this task

A method of increasing page usability is to hide components or show a different component across breakpoints. There are countless ways to use component visibility, but a few examples include:

- Use multiple button components horizontally across the top of a page for desktop and tablet size, but at mobile size, hide the buttons and show the dropdown component.
- Use a smaller button size for mobile size, but check that the buttons are large enough for users to press with a thumb. Consider changing the placement of buttons for mobile as well.
- Use images in desktop and tablet pages, but hide images, especially larger ones, in mobile.

In this procedure, show buttons across the top for the desktop and tablet form factors, but hide the buttons and use the dropdown component for mobile.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create > Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page from scratch with responsive authoring.
   For more information about how to create a page, see Create a page in UI Builder.
4. At the top, check that the desktop form factor icon is selected.

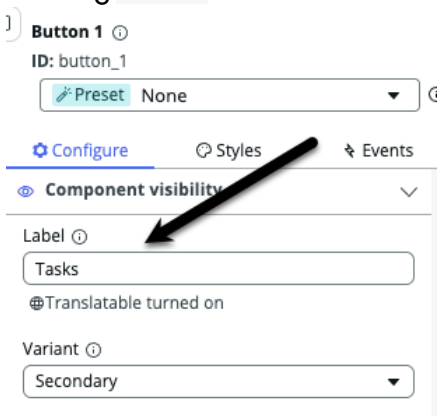
5. Add a column layout by selecting **+ Add content** under **Body** in the content tree.
6. On the **Layouts** tab, select **Single column**.
7. Add the first button component.

   a. Select the **+** icon in the center of the new column on the stage.

   b. On the **Components** tab, find and select the **Button** component.

   c. In the configuration panel, select **None - Configure the component manually**.

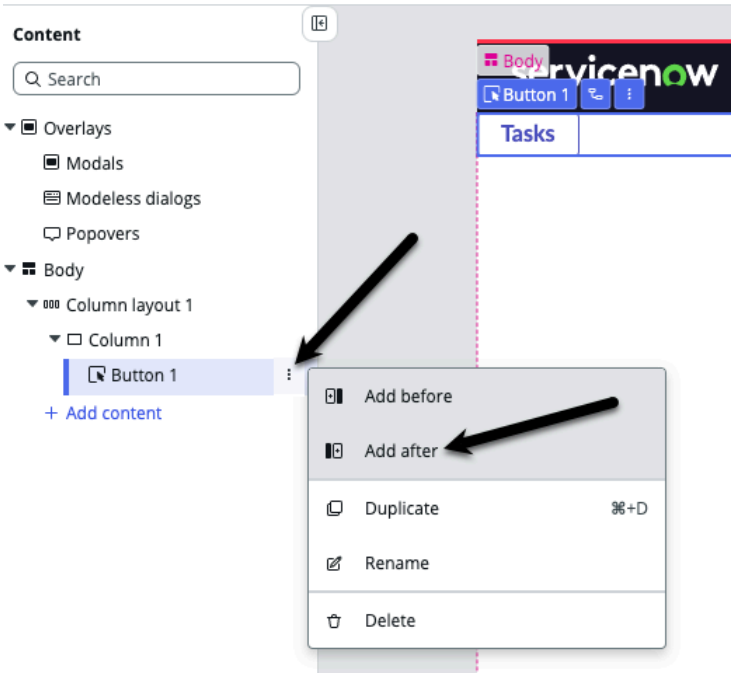
   d. In the configuration panel, on the **Configure** tab, replace the default text in the **Label** field by entering `Tasks`.



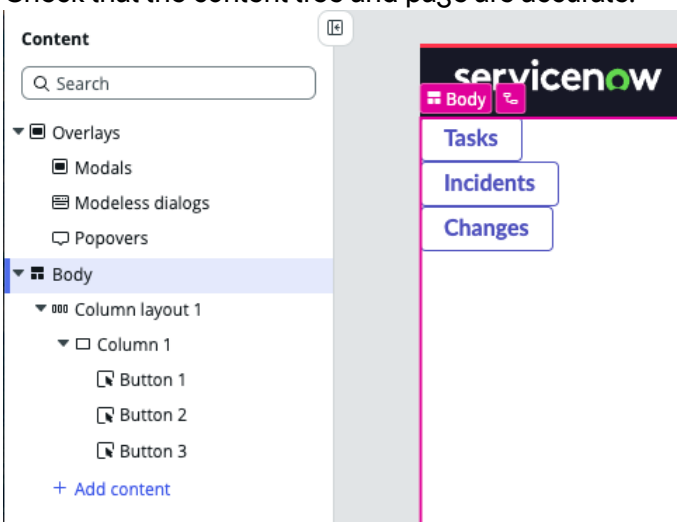
   e. Select **Save**.

8. Add two more buttons by repeating step 7 twice and in 7d name the buttons `Incidents` and `Changes`.

Note:
To add the buttons, point to **Button 1** in the content tree, select the menu icon, and then select **Add after**.



9. Check that the content tree and page are accurate.



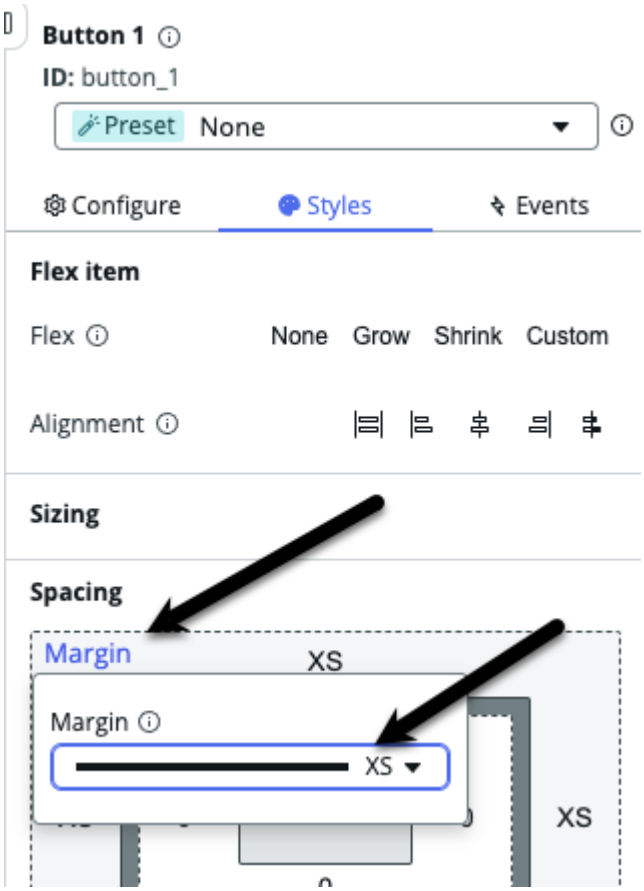
10. Add some space around each button.

    a. In the content tree, select **Button 1**.

    b. In the configuration panel, select the **Styles** tab.

    c. In **Spacing**, select **Margin**.

    d. In the menu, select **XS** (extra small).



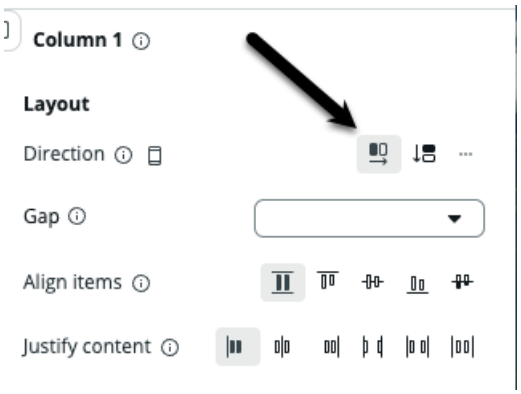
e. Select **Save**.

11. Add space around the other two buttons by repeating step 10 for each.

12. Change the buttons from the vertical direction to the horizontal direction.

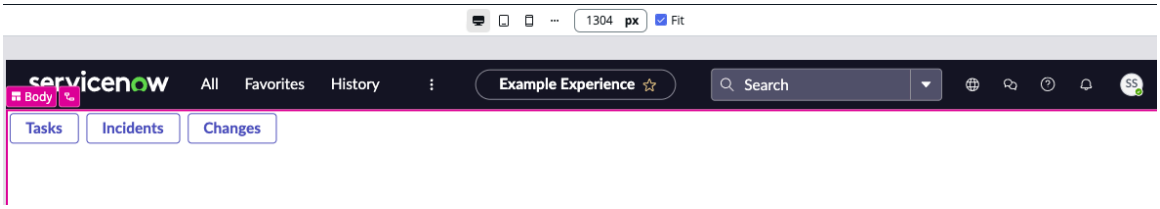
a. Select **Column 1** in the content tree.

b. On the configuration panel, in **Direction**, select the row icon.



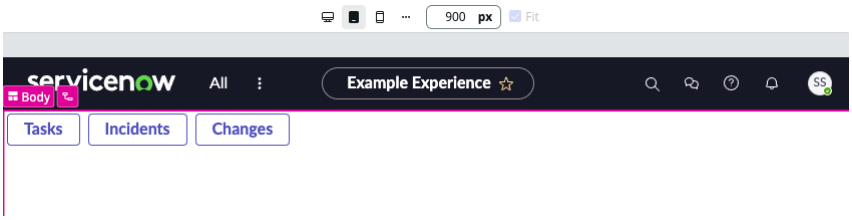
c. Select **Save**.

The page looks good at desktop width.



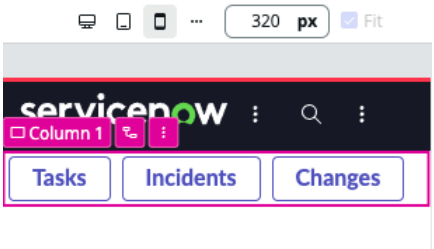
13. Select the tablet form factor icon.

The spacing and position of the buttons also looks good at tablet width.



14. Select the mobile responsive authoring icon.

The buttons use almost all the horizontal space at the smaller size.



15. Add the dropdown component.

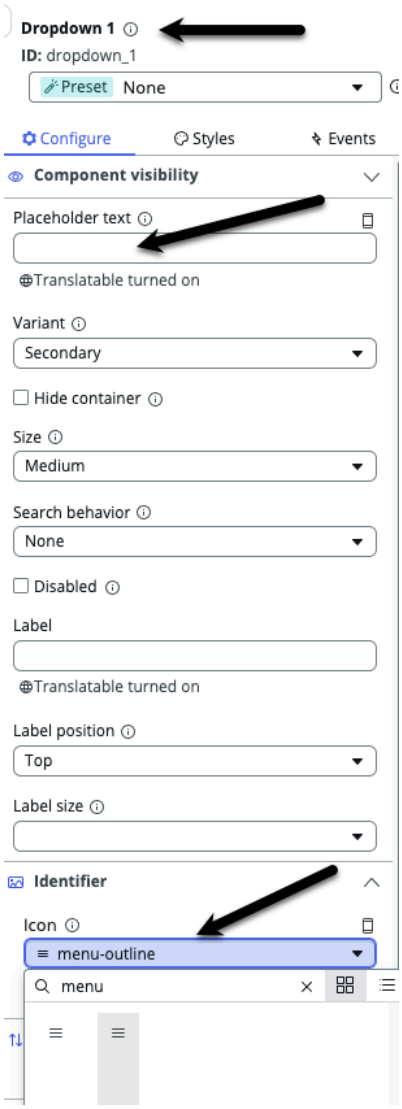
    a. Select the desktop form factor at the top of the stage.

    b. Point to **Button 3** in the content tree, select the menu icon, and then select **Add after**.

    c. On the **Components** tab, find and select the **Dropdown** component.

    d. In the configuration panel, select **None - Configure the component manually**.

    e. In the configuration panel, on the **Configure** tab, remove the default text in **Placeholder text** and select **menu-outline** in **Icon**.



f. Select **Save**.

16. Hide the dropdown component at the desktop and tablet form factors.

a. In the content tree, select **Dropdown 1**.

b. In the configuration panel, on the **Configure** tab, select **Component visibility** to show the options.

c. Select **Hide component**.

d. Select **Save**.

17. At the mobile form factor, hide the button components.

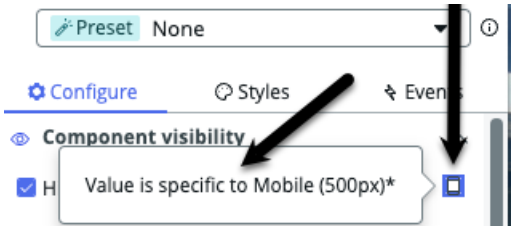
a. Select the mobile form factor icon at the top of the stage.

b. In the content tree, select **Button 1**.

c. In the configuration panel, on the **Configure** tab, select **Hide component**.

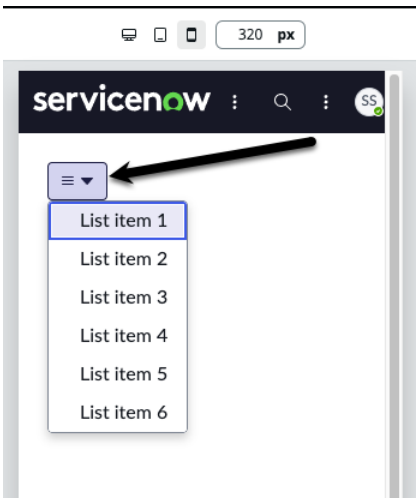
d. Repeat steps b-c for **Button 2** and **Button 3**.

e. Select **Save**.
Two locations show that the button components are hidden at mobile size. First, in the configuration panel, next to the **Hide component** option, select the mobile icon. A message is displayed to confirm that the value (hide component selected) is for the mobile size.



Second, in the content tree, the three button components are displayed with the hidden icon.

18. At the mobile form factor, show the dropdown component.
    a. In the content tree, select **Dropdown 1**.
    b. In the configuration panel, on the **Configure** tab, select **Hide component** to clear the field.
    c. Select **Save**.
19. Select **Preview**.
    The mobile form factor is displayed with the menu icon. If you select the arrow, the default options are displayed (List item 1, List item 2, and so on). You would configure the dropdown component to display **Tasks**, **Incidents**, and **Changes** just like the horizontal buttons available at the tablet and desktop form factors.



20. Select the desktop and tablet form factor icons at the top.
    The desktop and tablet form factors only show the horizontal buttons, not the menu icon.
21. Close the preview overlay by selecting the **X**.

Configuring components for responsive authoring

When creating pages with responsive authoring for different form factors, learn how adjusting some component configuration options can make pages look and work better at smaller sizes.

Before you begin

Role required: ui_builder_admin

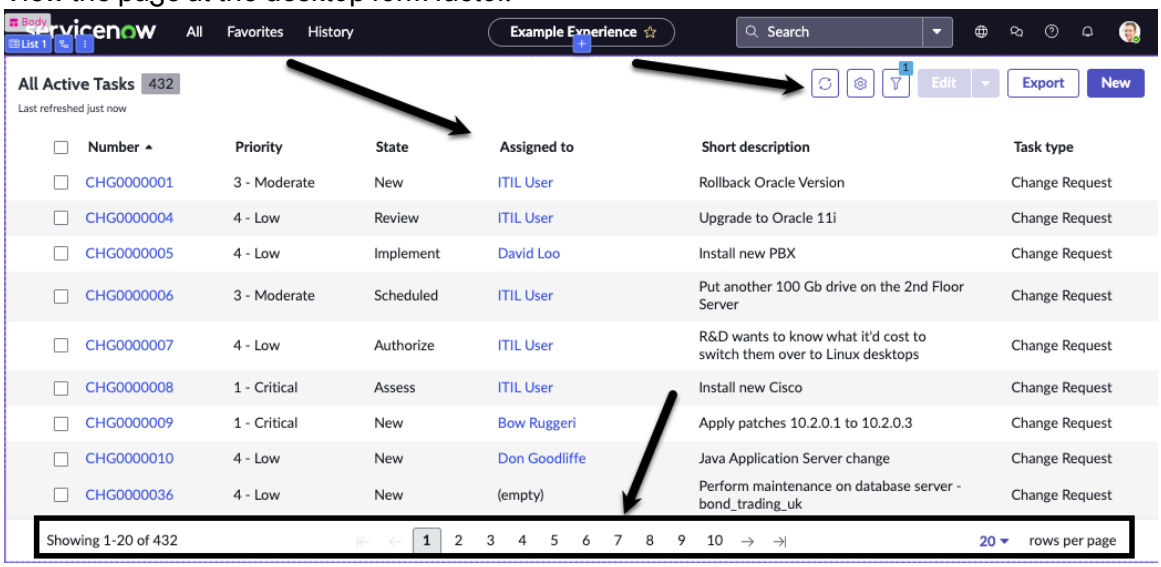
About this task

A method of increasing page usability is to tailor the look and feel of components using different configurations across breakpoints. There are countless ways to use component configuration. In this procedure, edit several configuration options for the List component to make it more suitable for smaller form factors.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create > Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page from scratch with responsive authoring.
   For more information about how to create a page, see Create a page in UI Builder.
4. At the top, check that the desktop form factor icon is selected.

5. On the stage, select **+ Add content**.
6. On the **Layouts** tab, select **Single column**.
7. Add the list component by selecting the **+** icon in the center of the new column.
8. On the **Components** tab, find and select the **List** component.
9. In the configuration panel, select **None - Configure the component manually**.
10. View the page at the desktop form factor.



Note the elements displayed, including the icons at the top, the number of columns, and the pagination information at the bottom.

11. Select the mobile form factor icon.

    At the mobile form factor, the list component is crowded and contains both vertical and horizontal scroll bars.

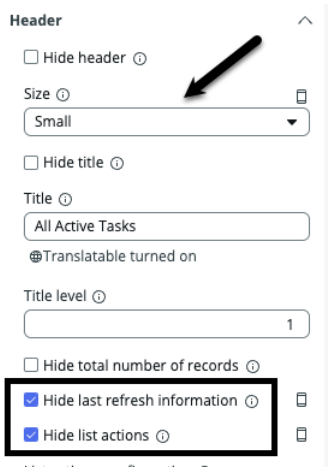
Make some simple configuration changes to improve the usability and appearance for the mobile form factor.

12. Select **List 1** in the content tree.
13. In the configuration panel, on the **Configure** tab, find the **Number of displayed columns** field and enter 1.
    Only the **Number** column is displayed. The horizontal scroll bar was removed.
14. In the configuration panel, on the **Configure** tab, scroll down to make the following changes in the **Header** section:

    a. Change the **Size** to **Small**.

    b. Select the **Hide last refresh information** option.

    c. Select the **Hide list actions** option.



As you made each change, the stage updated automatically. The header is a smaller size, the refresh time was removed, and the list actions icon was removed.

d. Select **Save**.

15. On the stage, scroll to the bottom of the mobile view.

16. In the configuration panel, on the **Configure** tab, scroll down and make the following changes in the **Pagination** section (if necessary, select **Pagination** to expand the field):

a. Select the **Hide range** option.

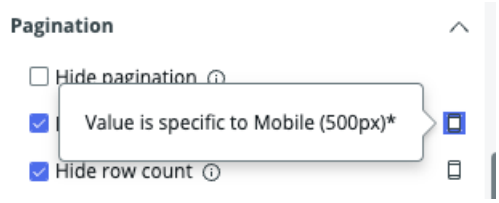
b. Select the **Hide row count** option.

c. Select the **Hide rows per page selector**.

d. Select **Save**.
The range, row count, and rows per page selector were removed. (The row count is still available at the top of the mobile screen).

In the configuration panel, the options and fields you edited are marked with the mobile cascade icon.

17. Select the mobile cascade icon next to any option or field to confirm that the value is for the mobile form factor.



18. Select the tablet and the desktop form factor icons at the top to confirm that no changes were made to the larger form factors.

Adjust styles for responsive authoring

When creating pages with responsive authoring for different form factors, learn how to change style options to increase the usability of the page at smaller sizes.

Before you begin

Role required: ui_builder_admin

About this task

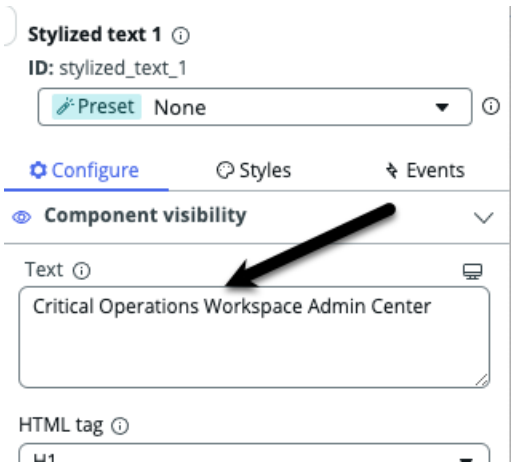
A method of increasing page usability is to tailor components using different styles across breakpoints. There are numerous options for editing styles. In this procedure, edit the margin around the stylized text component and reduce the text size so it uses less space on smaller form factors.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create > Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page from scratch with responsive authoring.
   For more information about how to create a page, see Create a page in UI Builder.
4. At the top, check that the desktop responsive authoring icon is selected.

5. On the stage, select **+ Add content**.
6. On the **Layouts** tab, select **Single column**.
7. Add the stylized text component.
   a. Select the **+** icon in the center of the column on the stage.
   b. On the **Components** tab, find and select the **Stylized text** component.
   c. In the configuration panel, select **None - Configure the component manually**.
   d. In the configuration panel, on the **Configure** tab, double-click (or use the keyboard shortcut) in the **Text** field to select the default text.

e. Replace the default text by entering `Critical Operations Workspace Admin Center`.



f. Select **Save**.

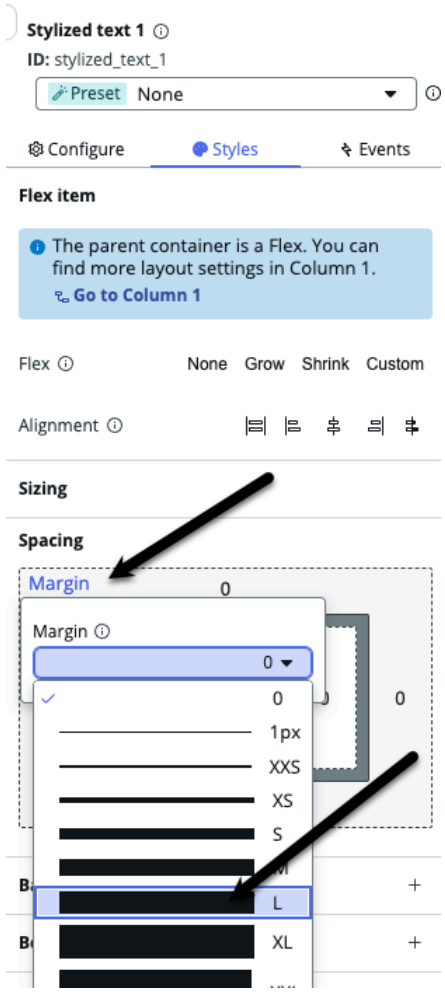
8. Add some space around the stylized text component and center it.

a. In the content tree, select **Stylized text 1**.

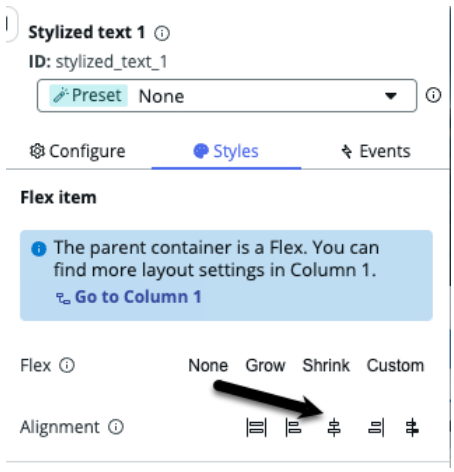
b. On the configuration panel, select the **Styles** tab.

c. In **Spacing**, select **Margin**.

d. Select **Large** in the list.



e. In **Alignment**, select the center icon.



f. Select **Save**.

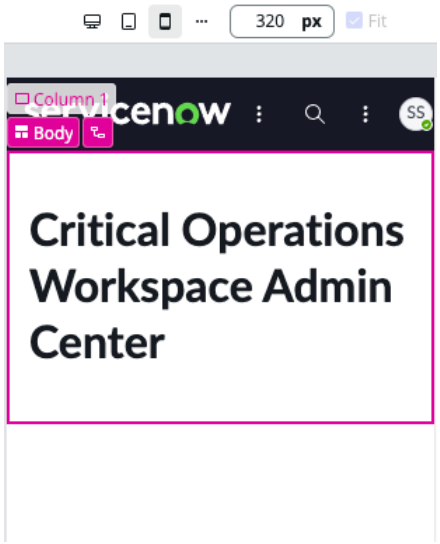
The stage updates automatically, showing the additional white space around the stylized text component and centering it.

9. Select the tablet form factor icon.

The spacing and heading size of the stylized text component looks good at tablet size.

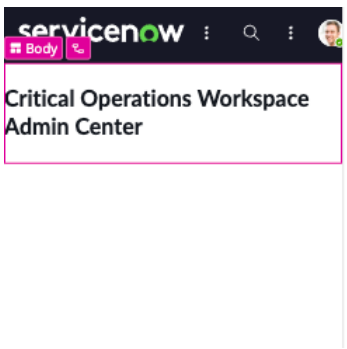
10. Select the mobile form factor icon.

The stylized text component uses a great deal of vertical space at the smaller size.



11. Remove the extra margin space and make the heading smaller just for the mobile form factor.
    a. In the content tree, select **Stylized text 1**.
    b. On the configuration panel, select the **Styles** tab.
    c. In **Spacing**, select **Margin**.
    d. Select **0** in the list.
    e. Select the **Configure** tab.
    f. In **HTML tab**, select **H3**.
    g. Select **Save**.

    The stage updates automatically showing that the stylized text component uses less space.



12. Select the tablet and the desktop form factor icons at the top to confirm that no changes were made to the larger form factors.

Adjust layout for responsive authoring

When creating pages with responsive authoring for different form factors, learn how to adjust the layout to improve the look and feel of the page at smaller sizes.

Before you begin

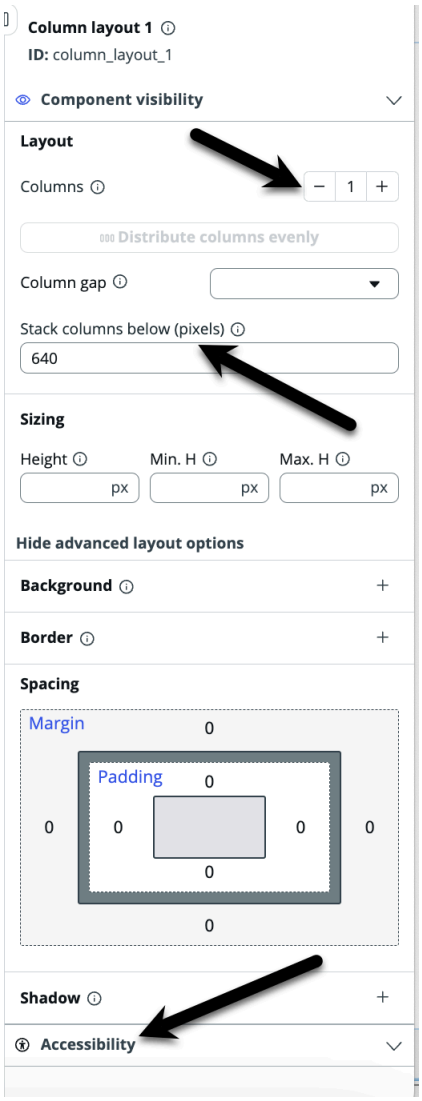
Role required: ui_builder_admin

About this task

A method of increasing page usability is to tailor components by adjusting the layout across breakpoints. There are numerous options for adjusting the layout. In this procedure, adjust the layout of the tabs component for the mobile form factor.

Most column layout options on the property pane can be edited for different form factors. However, the following options have global values across form factors:

- Number of columns
- **Stack columns below (pixels)** option
- All options under **Accessibility**



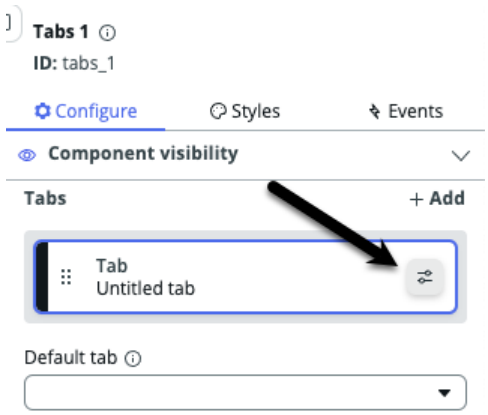
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create > Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page from scratch with responsive authoring.

For more information about how to create a page, see Create a page in UI Builder.

4. At the top, check that the desktop form factor icon is selected.

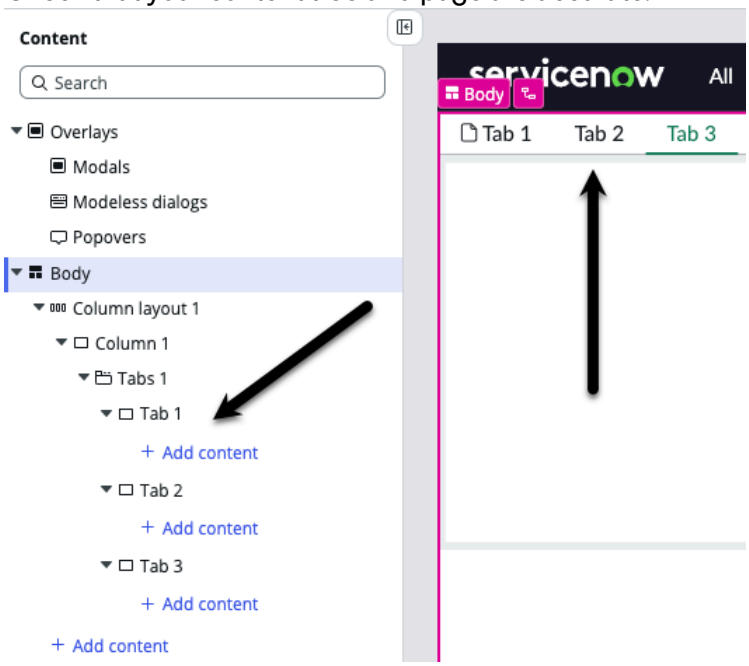
5. On the stage, select **+ Add content**.
6. On the **Layouts** tab, select **Single column**.
7. Add the tabs component.
   a. Select the **+** icon in the center of the column on the stage.
   b. On the **Components** tab, find and select the **Tabs** component.
   c. Select **Save**.
8. In the configuration panel, on the **Configure** tab, edit the name of the first tab by selecting the edit icon.



9. In **Tab label**, enter `Tab 1`.
10. In **Tab ID**, enter `tab_1`
11. Select **Save**.
12. Add a second tab.
    a. On the **Configure** tab, select **+ Add** next to **Tabs**.
    b. Select **Start from an empty container** and then select **Next**.
    c. In **Tab label**, enter `Tab 2`.
    d. Check that **Tab ID** has been automatically filled in with **tab_2**.

e. Select **Create**.

13. Add a third tab by repeating step 12 and in 12c name the tab `Tab 3`.
14. Check that your content tree and page are accurate.

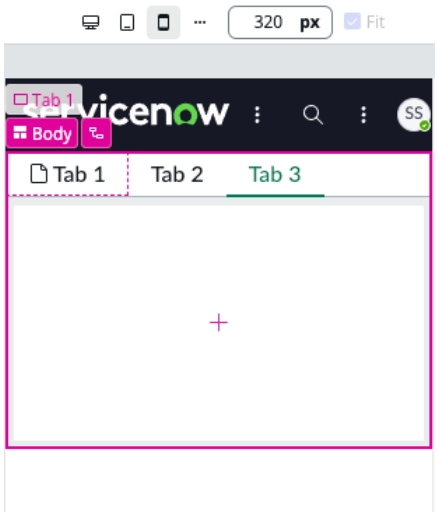


15. Select the tablet form factor icon.

The spacing and position of the tabs looks good at tablet width.

16. Select the mobile form factor icon.

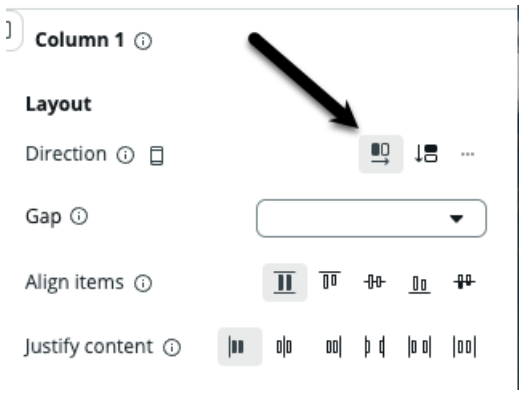
The tabs take more space at the smaller size.



17. Change the layout of the tabs just for the mobile size.

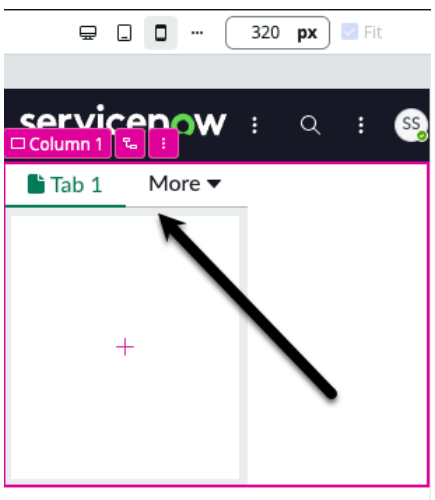
a. In the content tree, select **Column 1** under **Column layout 1**.

b. On the configuration panel, in **Direction**, select the row icon.

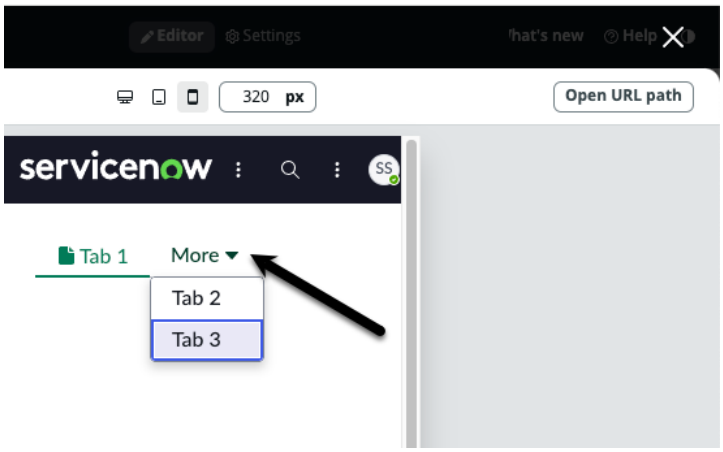


c. Select **Save**.

The stage updates automatically showing that tab 1 is visible and that there's a new dropdown menu.



18. Select **Preview**.
19. Select **More**.
    The other two tabs are available.



20. Select the desktop and tablet form factor icons to see that the More drop-down doesn't appear.
21. Select the **X** to close the **Preview** overlay.

Create a breakpoint for responsive authoring

Learn how to create custom breakpoints for responsive authoring to control the look and feel of a page at different form factors.

Before you begin

Role required: ui_builder_admin

About this task

UI Builder currently offers three default form factors:

- Desktop (1281 pixels and larger)
- Tablet (1280 pixels and smaller)
- Mobile (500 pixels and smaller)

Add up to three additional breakpoints. For example, there's a default form factor of 500 pixels and smaller for mobile devices, but you may need a breakpoint of 360 pixels.

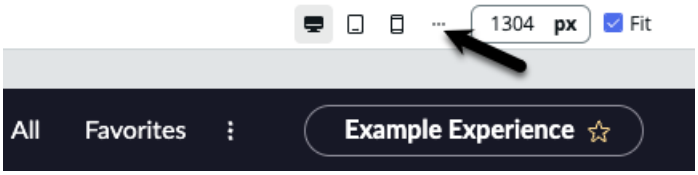
Note:
Custom breakpoints represent a range. If you create a breakpoint at 800 pixels, all changes apply to devices with a width at and lower than the new size. For example, a page opened on a screen 750 pixels wide uses the 800 pixel settings (properties and styles) unless there's another custom breakpoint at 750 pixels.

Procedure

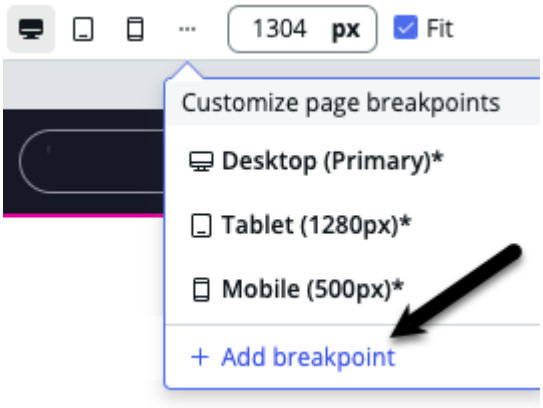
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create > Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Open an existing page created from scratch with responsive authoring or create a page from scratch with responsive authoring.

For more information about how to create a page, see Create a page in UI Builder.

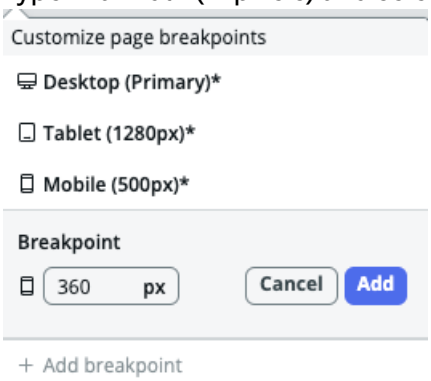
4. Select the menu icon next to the three form factor icons.



5. Select **+ Add breakpoint**.



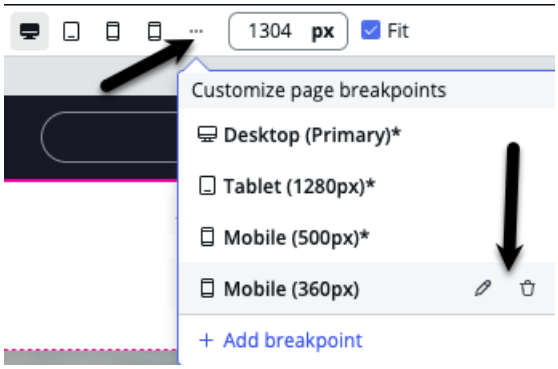
6. Type in a width (in pixels) and select **Add**.



The new breakpoint is saved and appears as a form factor icon.

As you build the page, select any of the form factor icons to view and edit the page at that size.

7. To edit or delete a custom breakpoint, select the menu icon, point to the breakpoint name, and select the edit or delete icon.



Note:
The default form factors (desktop, tablet, and mobile) can't be edited or removed.

Learn how to view and test your UI Builder experience

Preview your experience in UI Builder to see how it looks and functions while building an experience. Previewing helps ensure that your experience works as expected, that the data resources are available, and that the layouts are set up correctly.

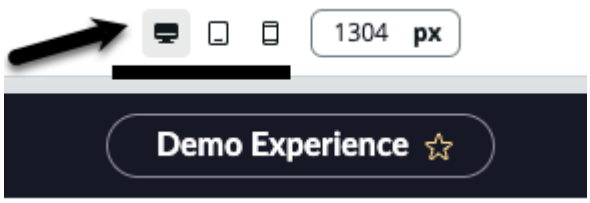
Test your page

You can test your page as the developer or see it as it would appear to an end user. Previewing doesn't require saving first.

Note:
Previewing inactive variants and page collections (sub pages) isn't currently supported.

You can test various aspects of your page.

- Test the variant that you're currently working on, including modals, viewports, and dynamic data by selecting the **Preview** button, which opens an overlay.
- Test with different data by changing the test values and selecting **Apply**. For more information, see Test values in a page.
- See what the variant looks like at different screen widths by selecting a form factor icon. For example, view the variant at tablet or mobile size. For more information about creating pages for different form factors, see Responsive authoring.



Note:
If you edit the test values or the form factor when previewing, the new test values or form factor are retained when you view the page variant in the editor. For example, if you changed the form factor to mobile while previewing, the page variant is displayed in the editor at the mobile form factor. You can always change to another size by selecting a form factor icon.

When you have finished testing, select the **X** in the upper right to close the overlay.

Test your page as an end user

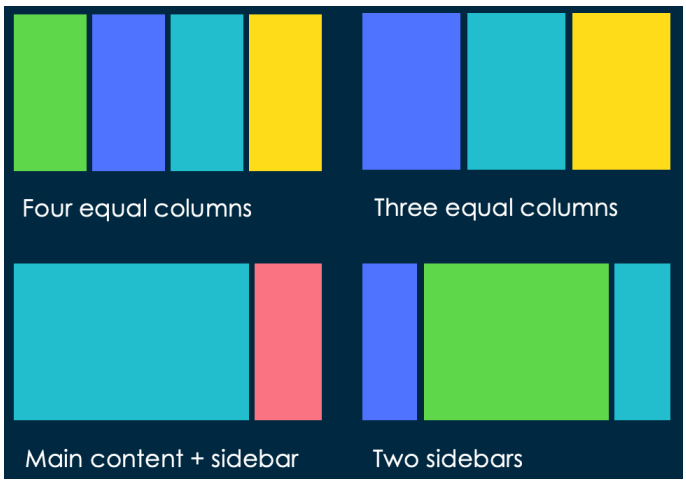
Test which variant users see by opening the URL path. Select the arrow next to the **Preview** button and in the drop-down list select **Open URL path**. UI Builder checks the audience, conditions, and condition order, and then opens the appropriate variant in a new browser tab. When you have finished testing, close the tab.

Organize components in UI Builder pages

Column layouts organize components on a UI Builder page. Learn how column layouts are used in UI Builder.

Column layouts enable you to create useful and visually appealing pages in an experience. A column layout is a type of flexible container that can hold components. Column layouts can have columns of equal or different widths. You can configure the column layout as a whole and the individual columns within the layout. For example, set a different border width and color for a column layout and for the individual columns within the layout. Column layouts can contain empty columns on a UI Builder page to create white space. Use multiple column layouts on a page. For example, a two-column layout could contain a header and image, then a three-column layout below it could hold more components like a list and form.

Think about the layout of your page before you begin building it. Analyze what you want to achieve and how complex the page layout must be. These questions determine what type of column layout or layouts you should use.



Column layouts in UI Builder use standards-based Cascading Style Sheets (CSS) rules and web layout technologies such as Flexbox, CSS grid, and absolute positioning. So whatever you can do in CSS you can do here. Visit Mozilla to learn more about CSS.

Note:
If you have existing UI Builder pages created in an earlier release (Utah or Tokyo), you can upgrade to Vancouver or Washington DC and add column layouts to the pages.

Column layouts

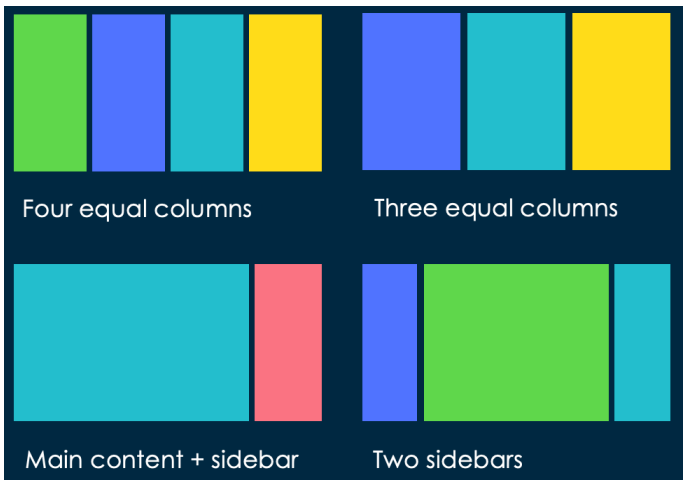
Column layouts are used to design and organize UI Builder pages.

This video shows you how to perform the following procedure. This video shows you how to add a column layout to a UI Builder page.

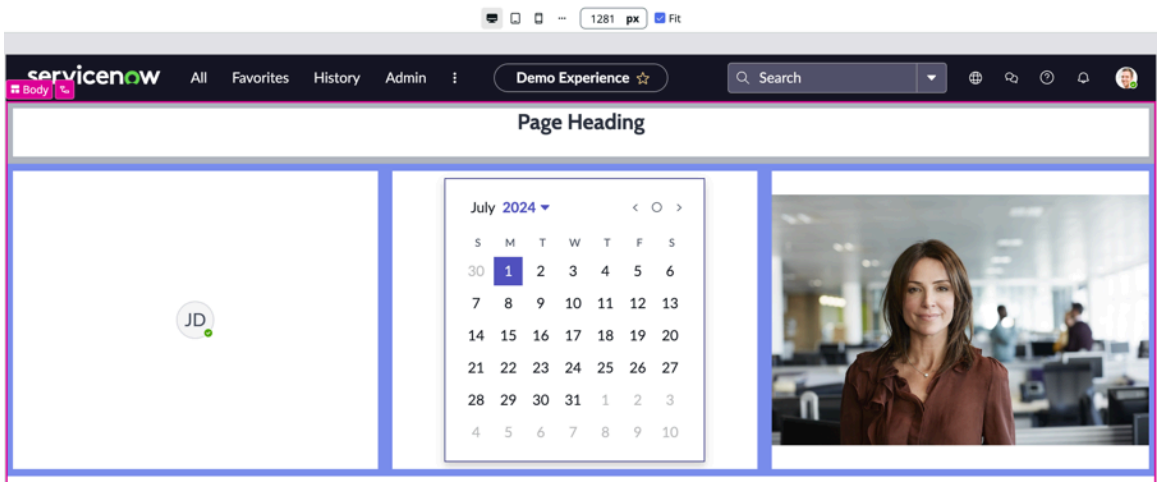
Column layouts enable you to create useful and visually appealing pages in an experience. A column layout is a type of flexible container that can hold components. Column layouts are

simple layouts that are preconfigured, enabling you to design an efficient and attractive page quickly. It's useful to define and set the structure of the page with column layouts before adding components.

Column layouts can have columns of equal or different widths.



You can configure the column layout as a whole and the individual columns within the layout. For example, set a different border width and color for a column layout and for the individual columns within the layout.



Note:
Currently, hiding/showing columns in a column layout based on conditions isn't supported.

Add a column layout

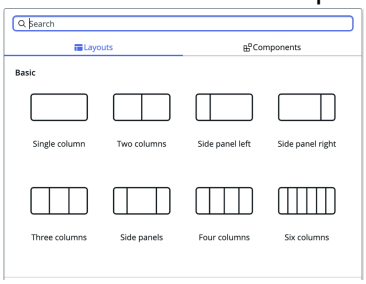
Add a column layout in UI Builder to build structure and organize components on an experience page.

Before you begin
Role required: ui_builder_admin

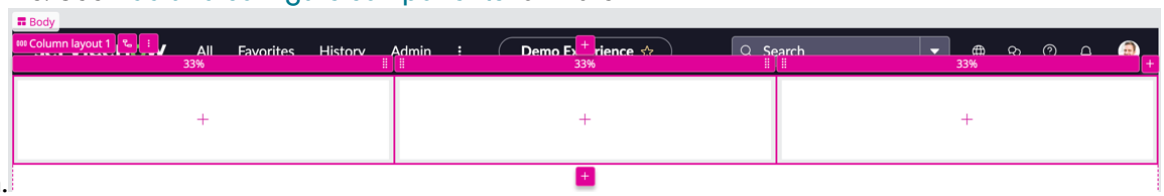
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.

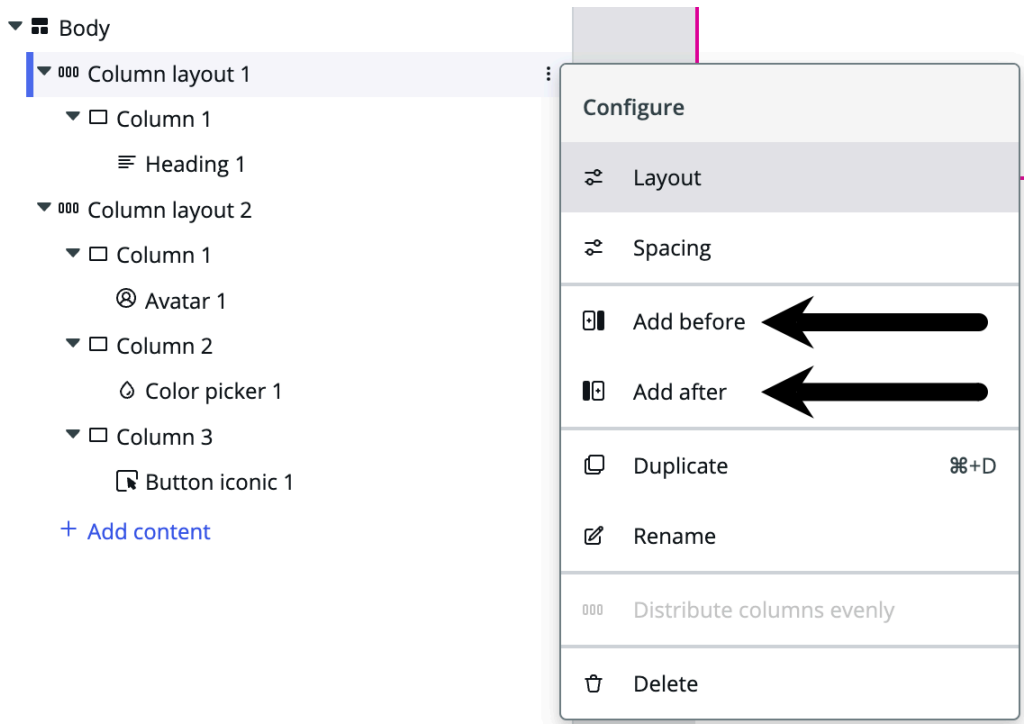
3. Create a page in UI Builder or open a page.
4. Select **+ Add content** in the content tree.
5. Select one of the **Basic** options on the **Layouts** tab.



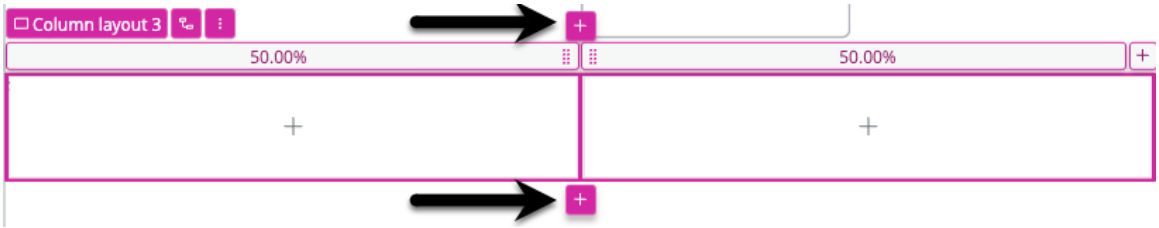
The column layout appears on the stage so you can add components to the columns. See Add and configure components for more information.



6. Add additional column layouts above or below.
   - In the content tree, select and hold (or right-click) on a column layout name or select the Menu icon for a column layout. Then, select **Add before** or **Add after** from the list and select a layout.



- On the stage, select a column layout, and then select the plus sign at the top or bottom.



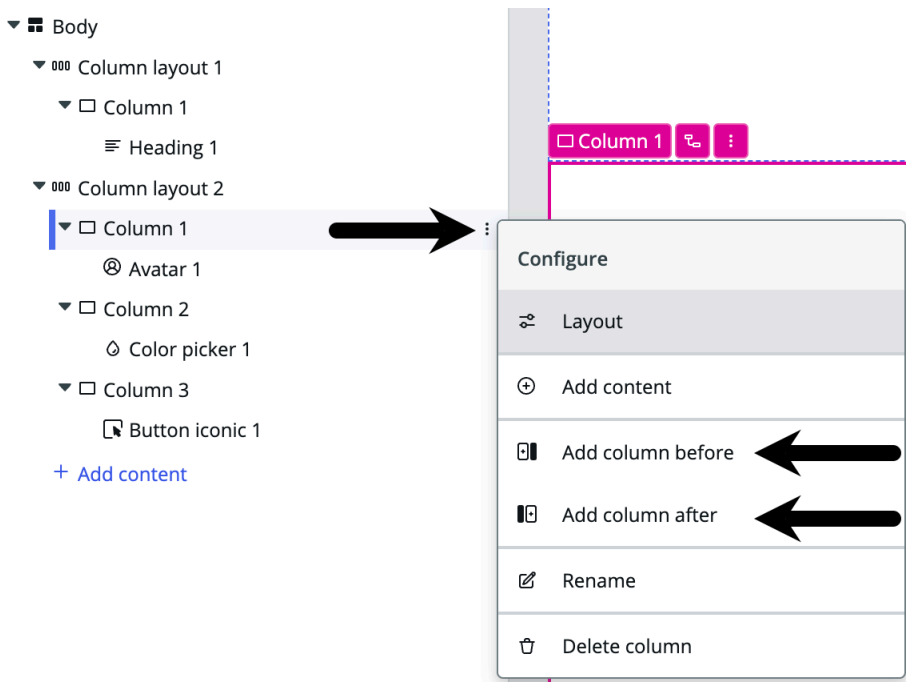
Configure column layouts

Configure column layouts in UI Builder to add the appropriate number of columns and the look you want on an experience page.

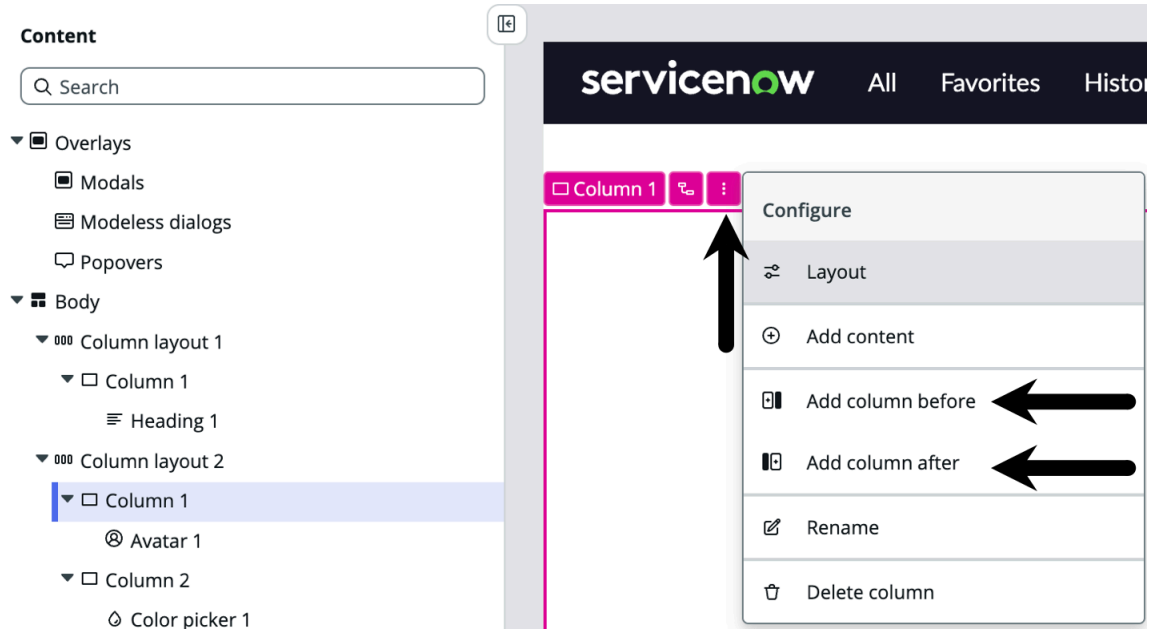
Before you begin
Role required: ui_builder_admin

Procedure

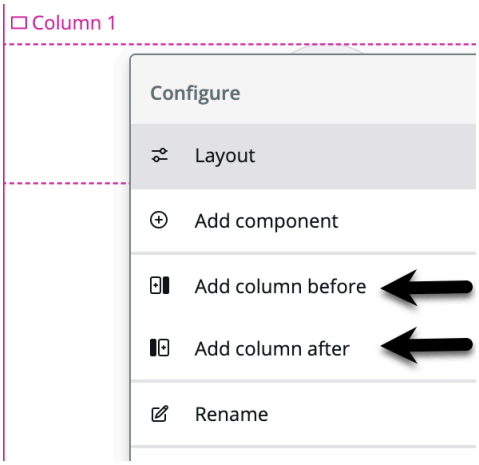
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page in UI Builder or open a page.
4. If the page doesn't already contain a column layout, Add a column layout.
5. Add (up to a maximum of six) columns using the content tree, the stage, or the configuration panel.
   - In the content tree, select and hold (or right-click) on a column name or select the Menu icon for a column and then select **Add column before** or **Add column after** from the list.



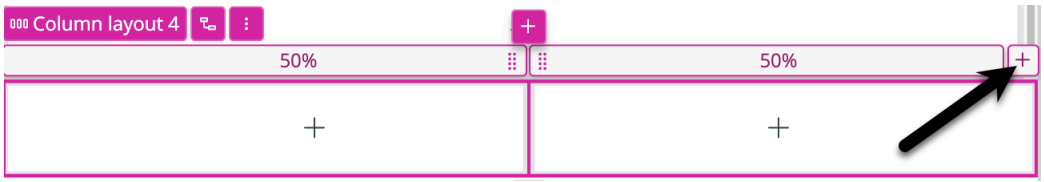
   - On the stage, select a single column, select the Menu icon next to the column name, and then select **Add column before** or **Add column after** from the list.



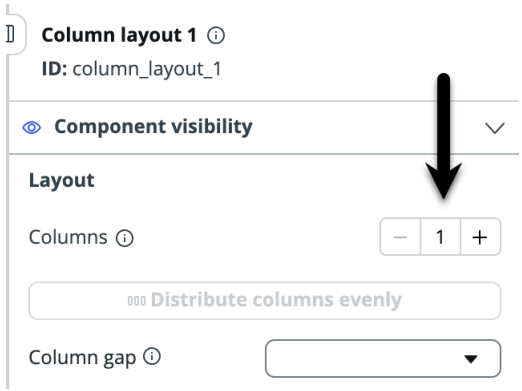
- On the stage, select and hold (or right-click) on a column and then select **Add column before** or **Add column after** from the list.



- In the content tree, select a column layout and then on the stage select the + plus icon to add columns to the right side of the column layout.

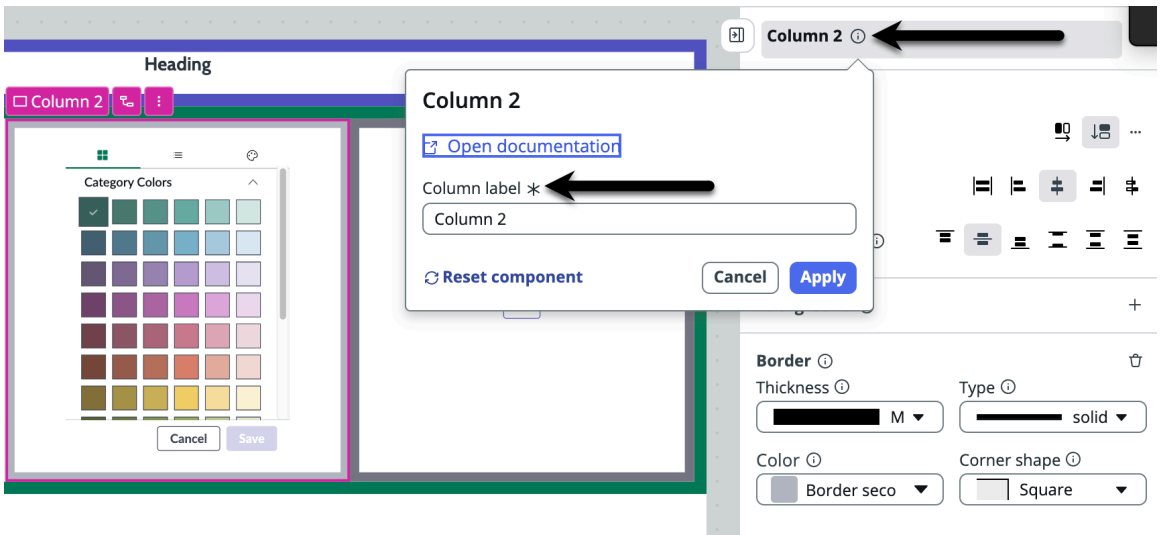


- In the content tree, select a column layout and then in the configuration panel use the + plus icon next to **Columns** to add columns to the right side of the column layout.



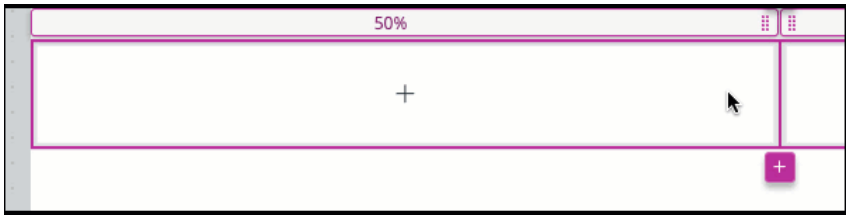
Note:
Columns can be nested inside one another, but this level of complexity isn't often required on pages.

6. Rename a column by editing the column label from the content tree, the stage, or the configuration panel.
   - In the content tree, select and hold (or right-click) on a column name or select the Menu icon for a column, select **Rename**.
   - On the stage, select the Menu icon next to a column name, select **Rename**.
   - In the configuration panel, select the information icon next to the column name.



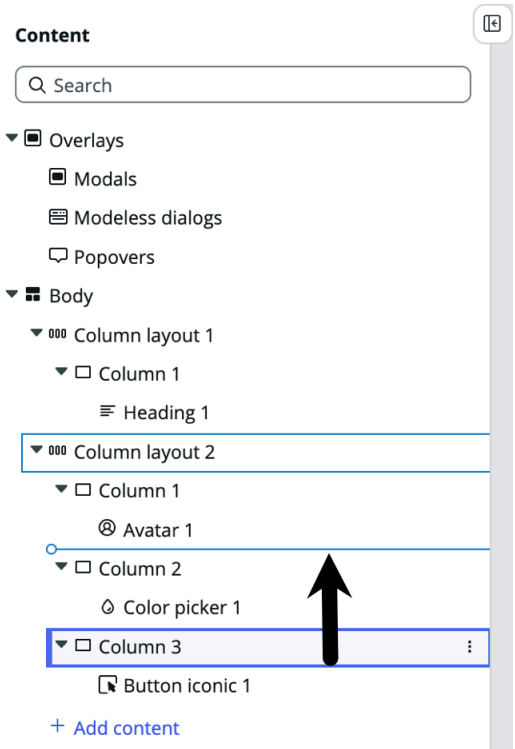
Renaming columns can help you distinguish between different columns on the stage and in the content tree. If you have a complex page with many column layouts and columns, it is useful to rename columns.

7. To change the width of a column in a layout, select the column layout (in the content tree or on the stage), then drag a column divider on the stage left or right to make the column narrower or wider.

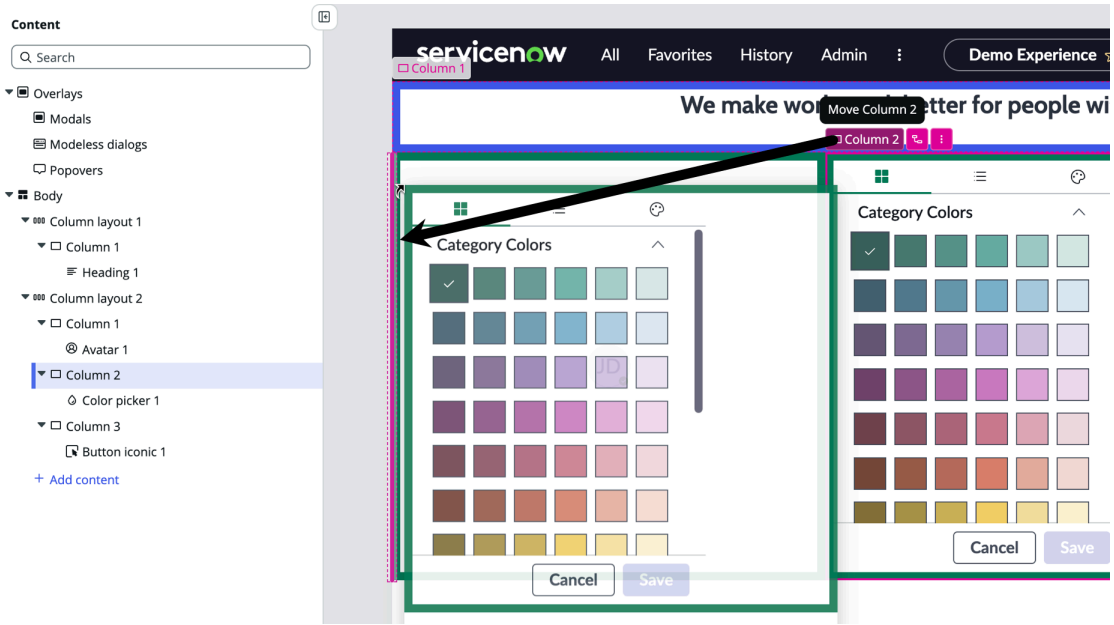


The width percentages at the top of the columns change automatically as you drag the column divider. The percentages add up to 100%. When resizing a column, be aware of how much space the component or components within the column need. For example, list and form components usually don't display well in narrow columns.

8. Distribute columns in a layout evenly across the horizontal using the content tree, the stage, or the configuration panel.
   - In the content tree, select and hold (or right-click) on a column layout name or select the Menu icon for a column layout, and then select **Distribute columns evenly** from the list.
   - On the stage, select a column layout, select the Menu icon next to the column layout name, and then select **Distribute columns evenly** from the list.
   - Select a column layout (in the content tree or on the stage) and in the configuration panel select **Distribute columns evenly**.
9. Reorder columns (including their contents) within a column layout using the content tree or the stage.
   - In the content tree, select the column and drag it to a different position (a blue horizontal line shows where the column can be dropped).



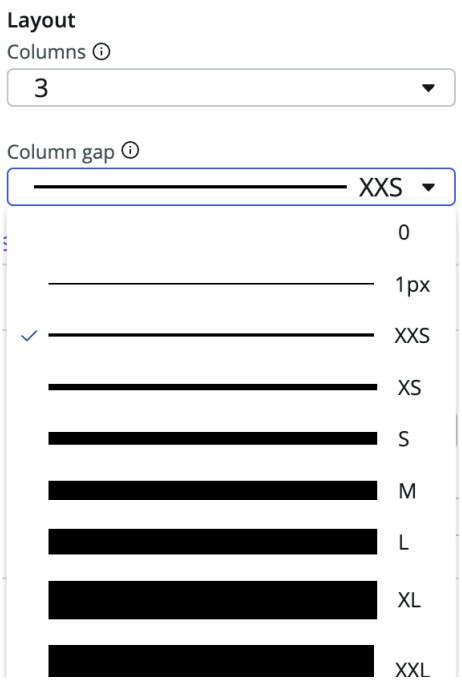
- On the stage, select a column header, drag the column to a different position, and drop the column into the drop zone defined by a vertical magenta line.



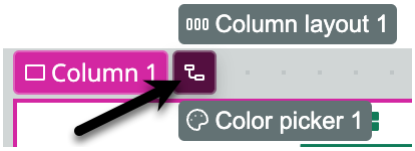
If you reorder the column in the content tree the stage updates, and if you reorder on the stage the content tree updates.

10. Specify the amount of space between columns from the content tree, the stage, or the configuration panel.

- In the content tree, select and hold (or right-click) on a column layout name or select the Menu icon for a column layout, select **Layout** from the list, and select a size option in **Column gap**.
- On the stage, select a column layout, select the Menu icon next to the column layout name, select **Layout** from the list, and select a size option in **Column gap**.
- Select a column layout (in the content tree or on the stage) and in the configuration panel select a size option in **Column gap**.



11. Select the Tree icon to navigate to a different layer of content, for example, from a column to the parent column layout or a child component.



12. Delete columns from the content tree or the stage.
    - In the content tree, select and hold (or right-click) on a column name or select the Menu icon for a column, and then select **Delete column** from the list.
    - On the stage, select a single column, select the Menu icon next to the column name, and then select **Delete column** from the list.
    - In the content tree, select a column layout and then in the configuration panel use the - minus icon next to **Columns** to remove columns from the right side of the column layout.

    For information about editing column spacing, see Set the gap between components in columns.

Set the gap between components in columns

In UI Builder, if a column contains multiple components, set the gap between the components.

Before you begin

Role required: ui_builder_admin

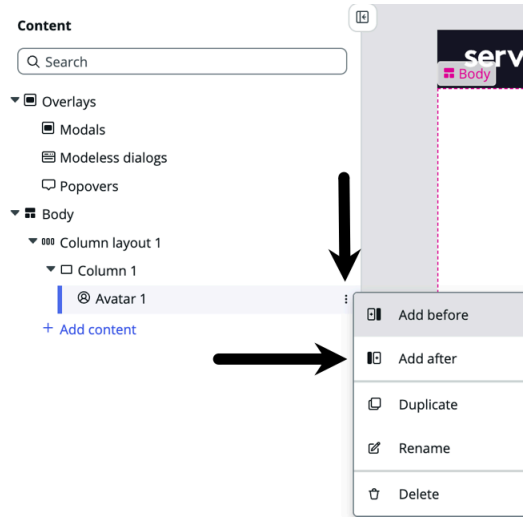
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page in UI Builder or open a page.
4. If the page doesn't already contain a column layout, Add a column layout.
5. Within a single column, add two components

    a. In the content tree, identify a column and select **+ Add content** under the column name.

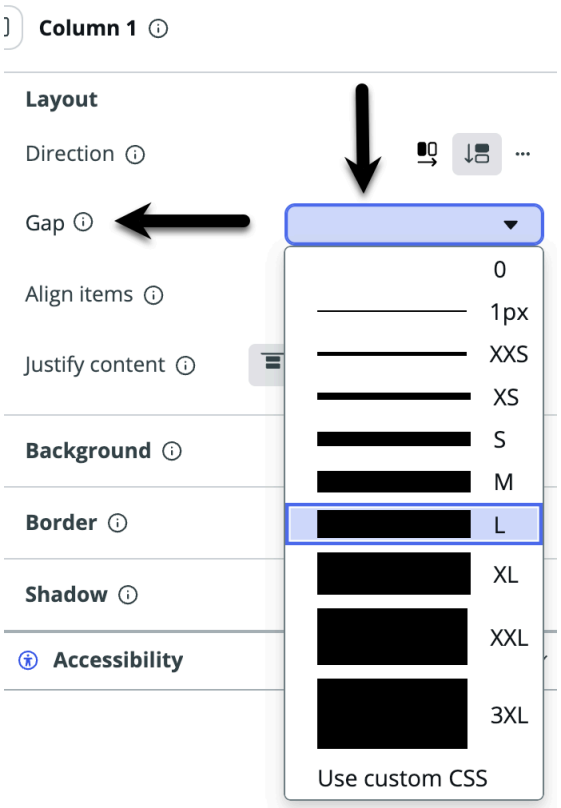
    b. In the toolbox window, select a component, for example, **Avatar**.

    c. In the content tree, move your mouse over the name of the component you just added, select the open menu (three vertical dots) icon, and select **Add after**.



d. In the toolbox window, select another component, for example, **Button**.

6. Select **Save**.
7. In the content tree, select the column to which you added the two components.
8. In the configuration panel, select a size option in **Gap**.



9. To specify an exact gap size in pixels, move your mouse over the **Gap** field, select the **Use custom value** (pencil) icon, and type in a number, for example, 300px.

10. To change between stacking the components and placing them next to each other in the column, select the **Row** or **Column** icon in **Direction**.

To reverse the order and placement of the components in the column, select the **Advanced** (three horizontal dots) icon in **Direction**, and then select the **Reverse** option.

Set advanced column layout options

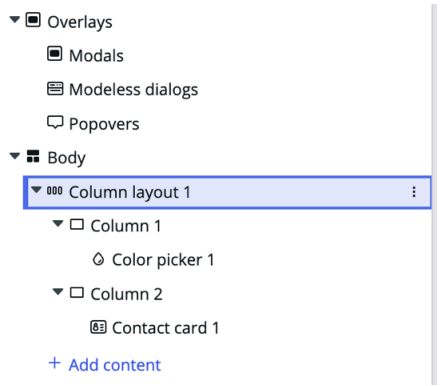
Set advanced column layout options in UI Builder including stack width and column layout height.

Before you begin

Role required: ui_builder_admin

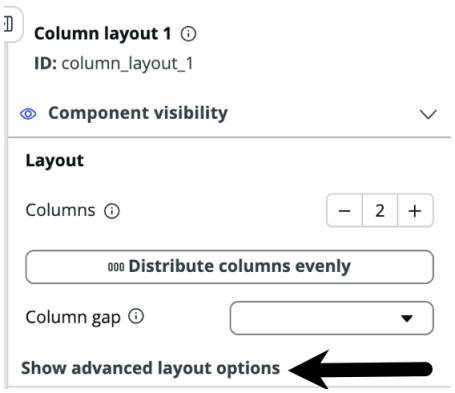
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page in UI Builder or open a page.
4. If the page doesn't already contain a column layout, Add a column layout.
5. Select a column layout containing two or more columns in the content tree.



6. To set a stack width, go to **Configuration panel > Layout** and select **Show advanced layout options**.

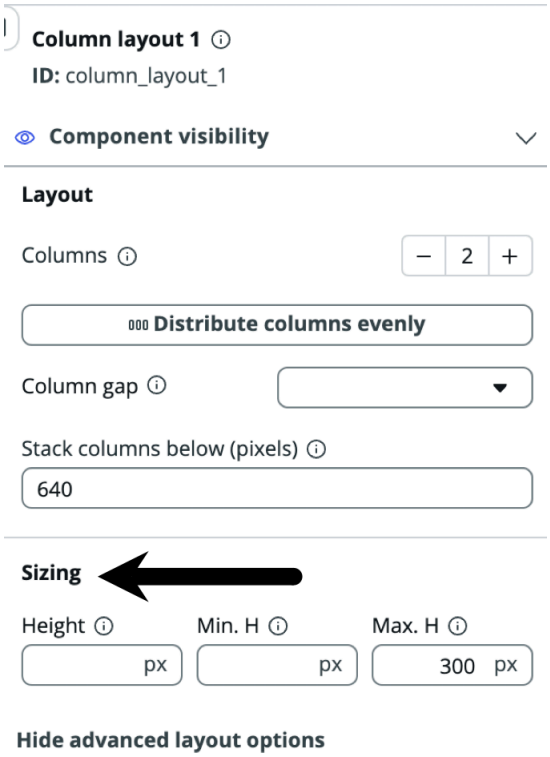
> **Note:**
> Setting a stack width enables you to control how a page with column layouts looks in a narrow browser window, tablet, or mobile device.



7. Type a width in **Stack columns below**.
8. Select **Save**.
9. To test the updated stack width, select the arrow next to **Preview** and select **Open URL path**.
10. Reduce the browser window width to less than the stack width that you specified to check that the columns on the right move below the first column.
11. Close the browser tab that opened with the URL path.
12. To control column layout height, select a column layout with two or more columns in the content tree, go to **Configuration panel > Layout**, and select **Show advanced layout options**.

> **Note:**
> Setting height is useful if your column layout contains a component that can be tall, like the List component.

13. Under **Sizing**, type numbers into one or more of the options (**Height**, **Min. H**, and **Max. H**), and set the sizing unit (for example, **px**, **%**, or **em**).



14. Select **Save**.
15. To test the updated sizing, select the arrow next to **Preview** and select **Open URL path**.
16. Reduce and expand the browser window height to test the specified values.
17. Close the browser tab that opened with the URL path.

Upgrading layouts in UI Builder

Upgrade layouts created in Quebec and Rome to the new layout system.

The UI Builder layout system was enhanced and updated in the San Diego release. This new layout system features a simplified content tree and a new styling panel for layout configuration and component styling. All newly created page variants use the new layout system. You have the option to upgrade to the new layout system for existing and custom pages.

The new layout system includes the following:

- A simplified content tree
- Improved styling panel for layout configuration and component styling
- Low-code UI for configuring CSS Flexbox and Grid layouts.
- Enhanced UI for component styling
- Updated layouts do not require slots

Things to look for after upgrading layouts

Variants and pages are upgraded individually.

A page/variant is made up of a layout, components, and subcomponents, that are wrapped in a container within a layout with data resources, event bindings, and composition. Only the layout of your page is upgraded when upgrading to the new layout system, so there is no impact to your data, components, or component styling.

All slots in the old layout system are migrated during the upgrade process to the new layout. The upgrade process changes how the component containers are structured on your page.

While component styling is not impacted, there may not be a one-to-one style migration of how components were positioned. Complex pages can have visual misalignments that occur through the upgrade process as styles are merged from slots to containers. Some issues must be resolved manually.

Upgrade your layout to the new layout system

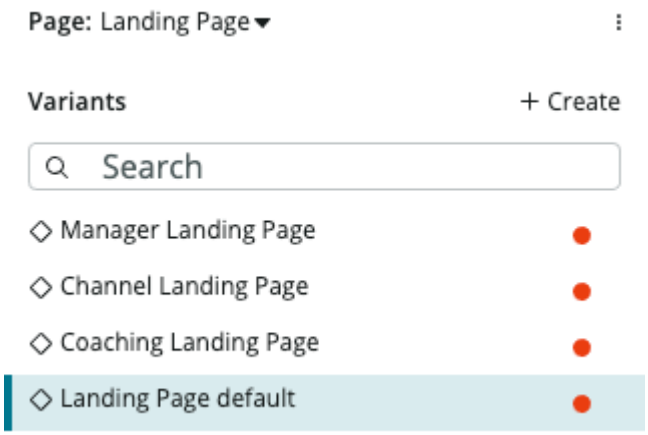
Upgrade your UI Builder pages to the new layout system.

Before you begin

Role required: ui_builder_admin

Procedure

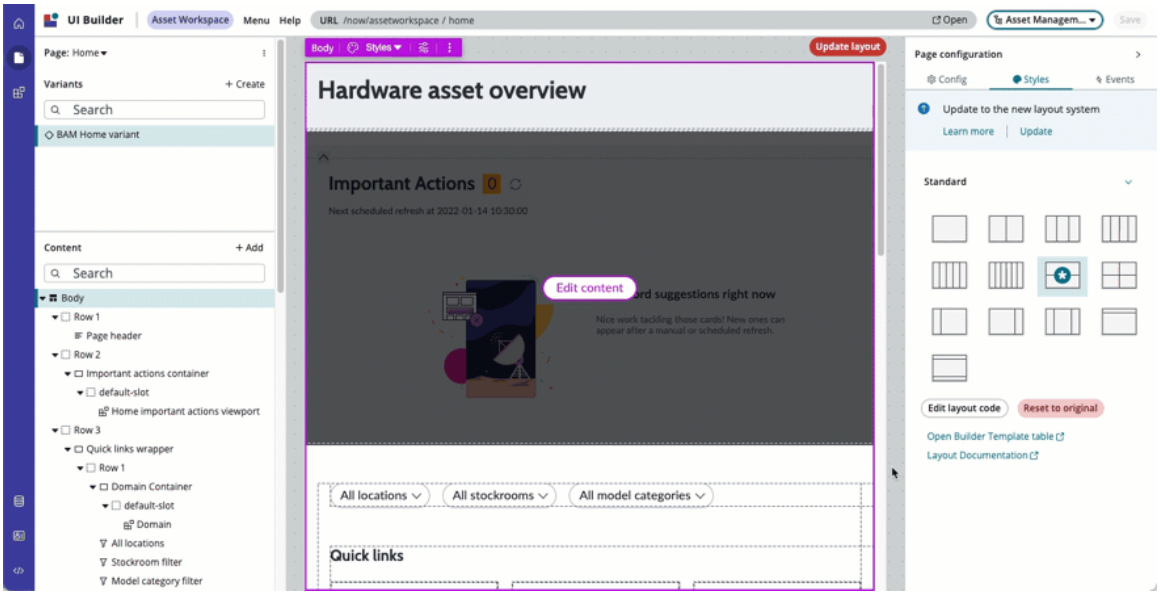
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience with the page you want to upgrade.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Select the page that you want to upgrade.



   You can find page variants that can be upgraded to the new layout system tagged with a red dot.

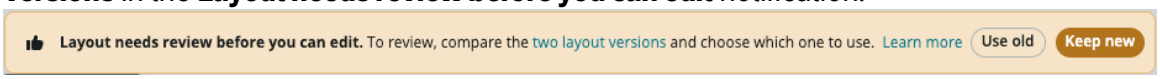
   Note:
   Legacy pages created using the pre-Quebec row/column layout system are not upgradable and must be rebuilt with the new layout system.

4. Select the **Update layout** button.



UI Builder begins updating your page layout to the new layout system. A notification appears when the process is complete.

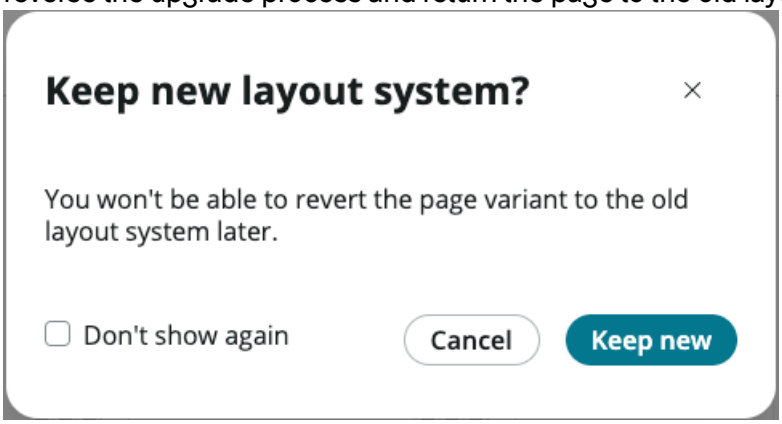
5. You can compare your old layout to the upgraded layout by selecting **compare the two layout versions** in the **Layout needs review before you can edit** notification.



Two browser tabs open, one displaying your current layout and one displaying the upgraded layout.

6. Select **Keep new** if you would like to update the page to the new layout system, or select **Use old** if you would like to keep the page layout as it was.
A modal appears asking you to confirm your selection.

7. Select **Keep new** to complete the upgrade to the new layout system, or select **Use old** to reverse the upgrade process and return the page to the old layout system.



8. The page reloads with the changes applied.

9. View and test your page by selecting the **Preview** button (

).

Using Flexbox layouts in UI Builder

Create a Flexbox layout in UI Builder to build powerful pages so that you can customize with cascading style sheets (CSS) and can improve your performance.

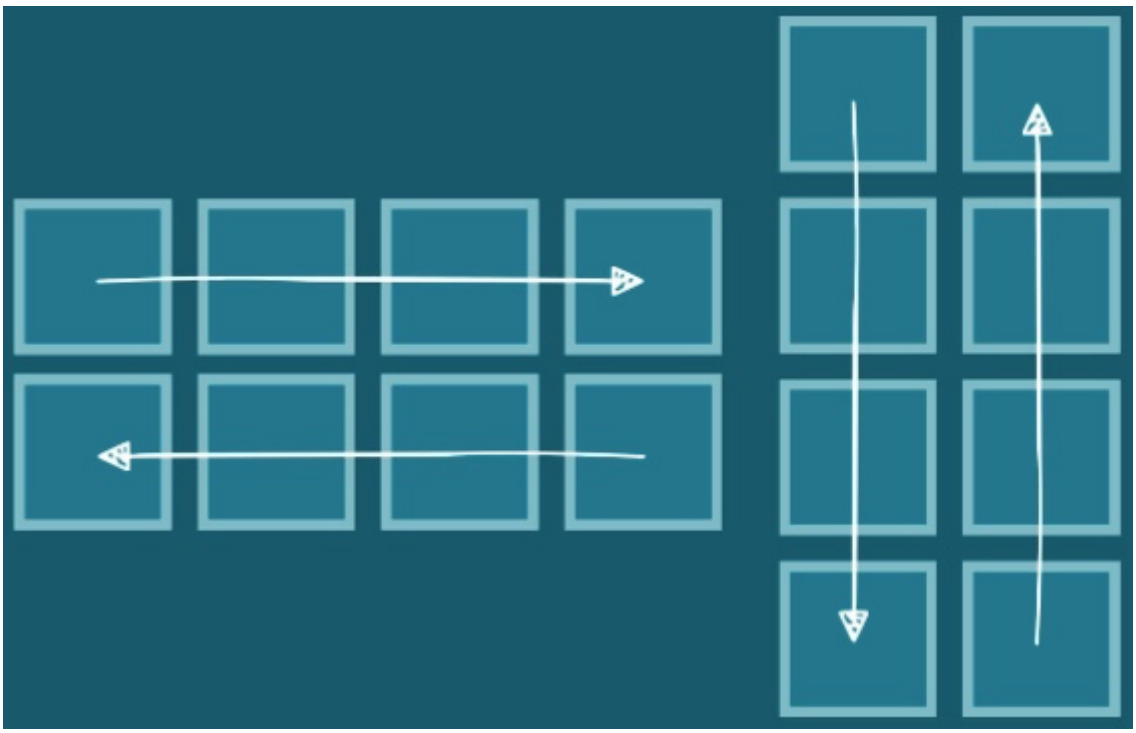
You can easily build custom pages with Flexbox layouts in UI Builder. Customizing the layout of your page lets you take full advantage of Flexbox so that you can achieve your overall page design. For more information, see Organize components in UI Builder pages.

Flexbox is a one-dimensional layout system in CSS. Flexbox is inherently flexible, which is useful for when you don't know the size of your content. If you plan to build complex pages, you can customize the layout of your page in the configuration panel styles tab or with CSS. Visit Mozilla to learn more about Flexbox.

You can change the direction of your content in the layout in the following ways:

- Row: Rows are organized left-to-right or right-to-left, depending on the direction of your browser's default language. Left/right is the case for an English browser.
- Row-reverse: Rows are organized in the reverse direction of your browser's default language, such as right-to-left or left-to-right.
- Column: Up/down or down/up
- Column-reverse: Down/up

Flexbox row and column directions



- Justify content: Defines the alignment along the main axis. Choices are as follows:
  - Flex-start: Items are at the start of the flex direction, similar to left-justified content. This is the default setting.
  - Flex-end: Items are at the end of the flex direction, similar to right-justified content.
  - Space-between: Items are distributed evenly.
  - Space-around: Items are distributed evenly with equal space around them.
  - Space-evenly: The spacing between items is equal.
- Align items: Defines how you want your flex content displayed along the cross axis. Choices are as follows:

- Stretch: Stretch your content to fill the container. This is the default setting
- Flex-start: Place your content at the start of the cross axis.
- Flex-end: Place your content at the end of the cross axis.
- Center: Center your content in the cross axis.
- Base system: Align your content the same as your baseline alignment.

- Height: Set the height automatically or manually.
- Width: Set the height of your flexbox items automatically or manually.
- Margin: Set your minimal distance between flexbox items.
- Padding: Set the padding for each side of your flexbox items.

Create a Flexbox layout with the new layout system

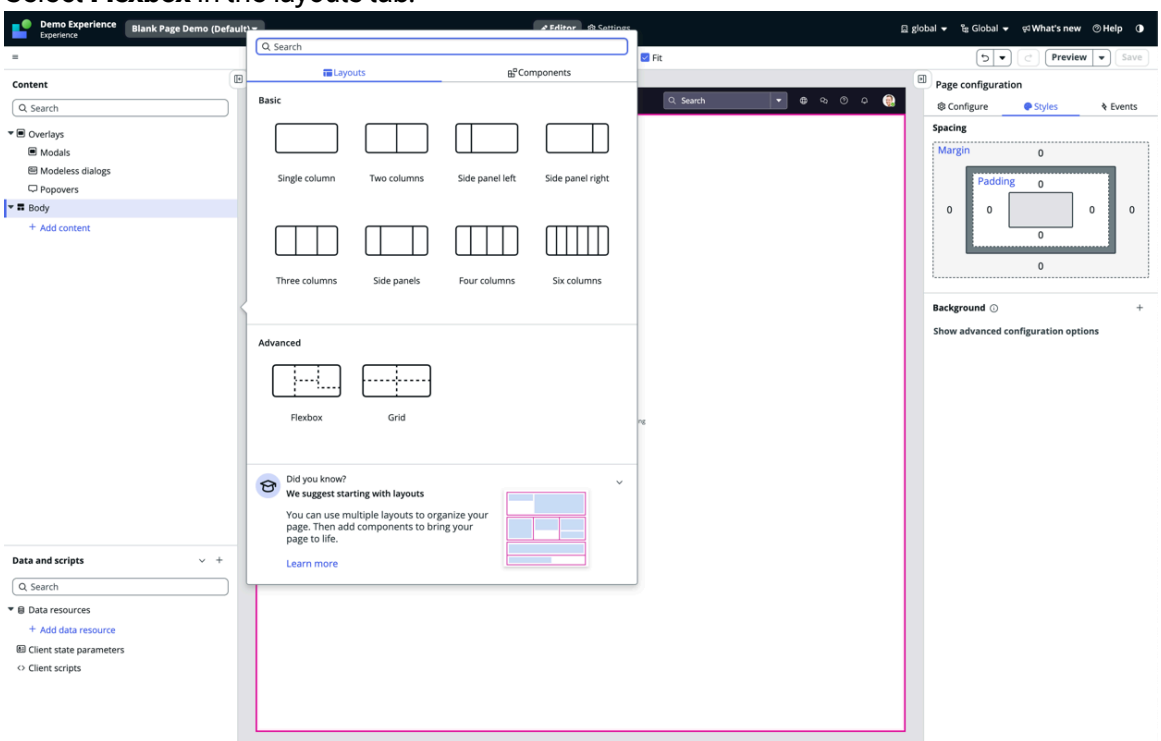
Create a Flexbox layout in UI Builder to build powerful pages in a low-code environment.

Before you begin

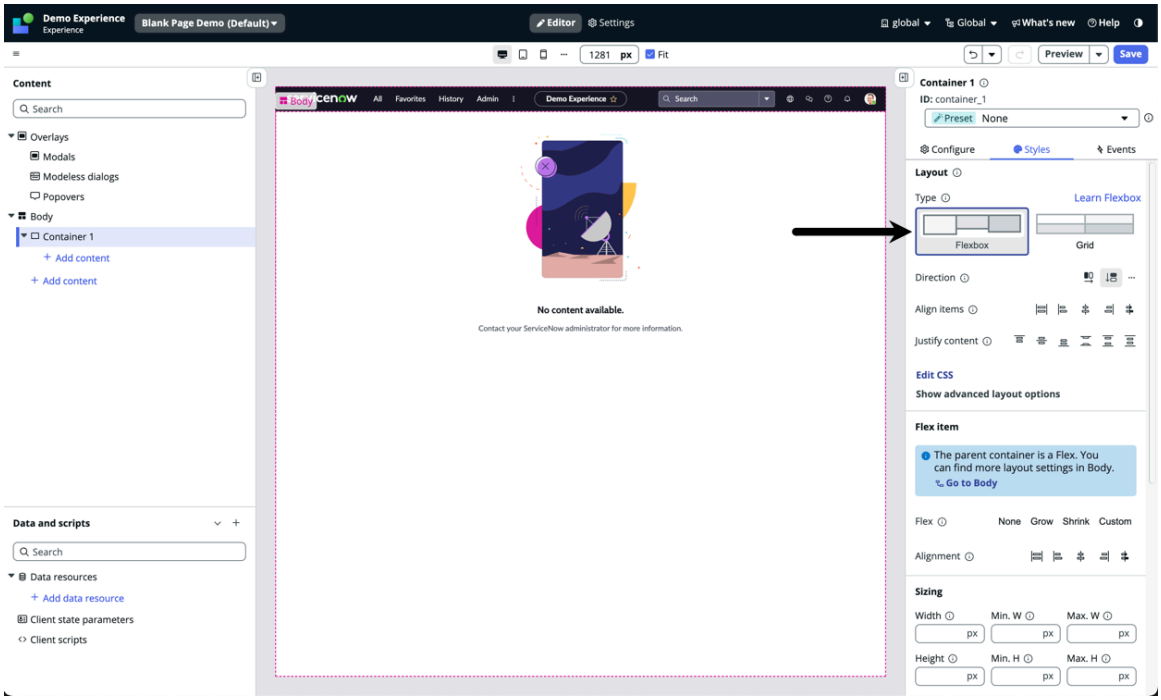
Role required: admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page in UI Builder.
4. Select the **+ Add content** button in the UI Builder stage.
5. Select **Flexbox** in the layouts tab.



6. Select the **Styles** tab in the configuration panel.
7. In the **Layout** section, you should see **Flexbox** highlighted.



8. In the **Layout** section of the styles tab, you can change the following:

   Direction

   Define how you want components to stack, horizontally or vertically.

   Align Items

   Define how components within containers align as a group.

   Justify content

   Define how to pack or space apart the components within the container.

   > **Note:**
   > Select **Show advanced layout options** to see the next two options.

   Gap

   Set the size of the gap between rows and columns.

   Wrap child elements to multiple lines

   Set whether components are forced onto one line or overflow onto multiple lines.

   Visit Mozilla to learn more about Flexbox layout configurations.

9. **Optional:** You can edit the CSS code by selecting **Show advanced configuration options** at the bottom of the **Styles** tab, and then selecting the **View and edit CSS** option.
10. Select **Save**.
    Your page refreshes and displays the changes made to the layout.
11. Add components to your page.
    See Add and configure components for more information.
12. View and test your page by selecting Preview.

Create a Flexbox layout with the old layout system

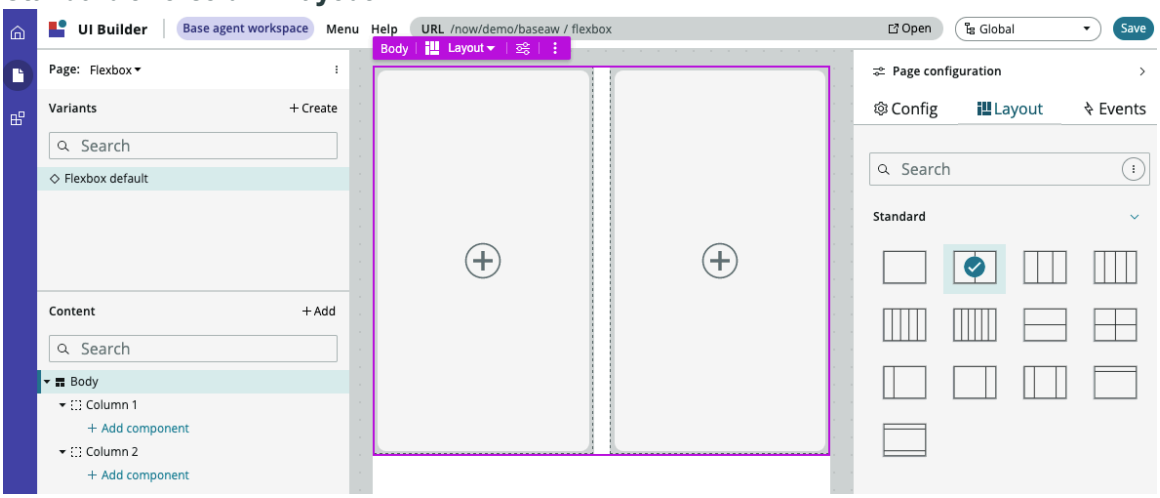
Create a Flexbox layout in UI Builder to build powerful pages so that you can customize with cascading style sheets (CSS) and can improve your performance.

Before you begin
Role required: admin

About this task
In the following procedure, you learn how to use Flexbox to change the CSS code to customize the layout of your page.

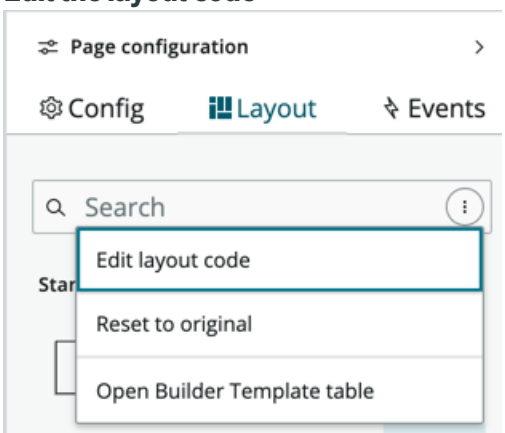
Procedure
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page in UI Builder or open a page.
4. Click the **Layout** tab and choose the two-column layout.

Standard two-column-layout



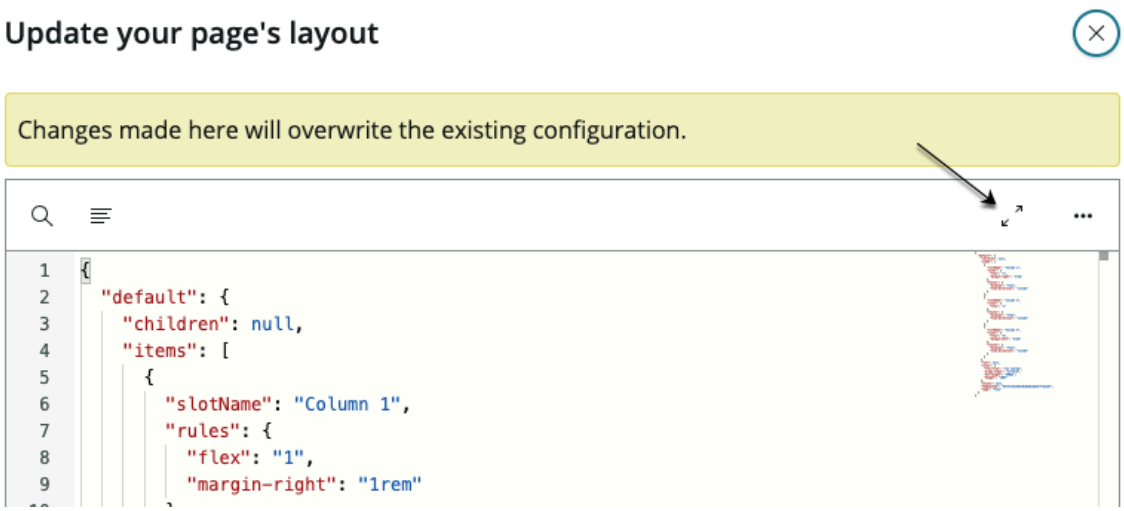
5. Click **Edit layout code**.

Edit the layout code



6. Expand the code editor so that you can easily view the CSS code.

Expand the code editor



7. Add another slot to the layout by copying the CSS code for an existing slot.

   Copying the code is easier than typing it into the layout.

Copy the CSS code for a slot

8. Below the code that you copied, place a comma and then paste the following CSS code:
   a. Change the *"slotName"* property to `"Column 3",`.
   b. Change the *"flex"* property to `"2"`.
   c. Add `"margin-left": "1rem"`.

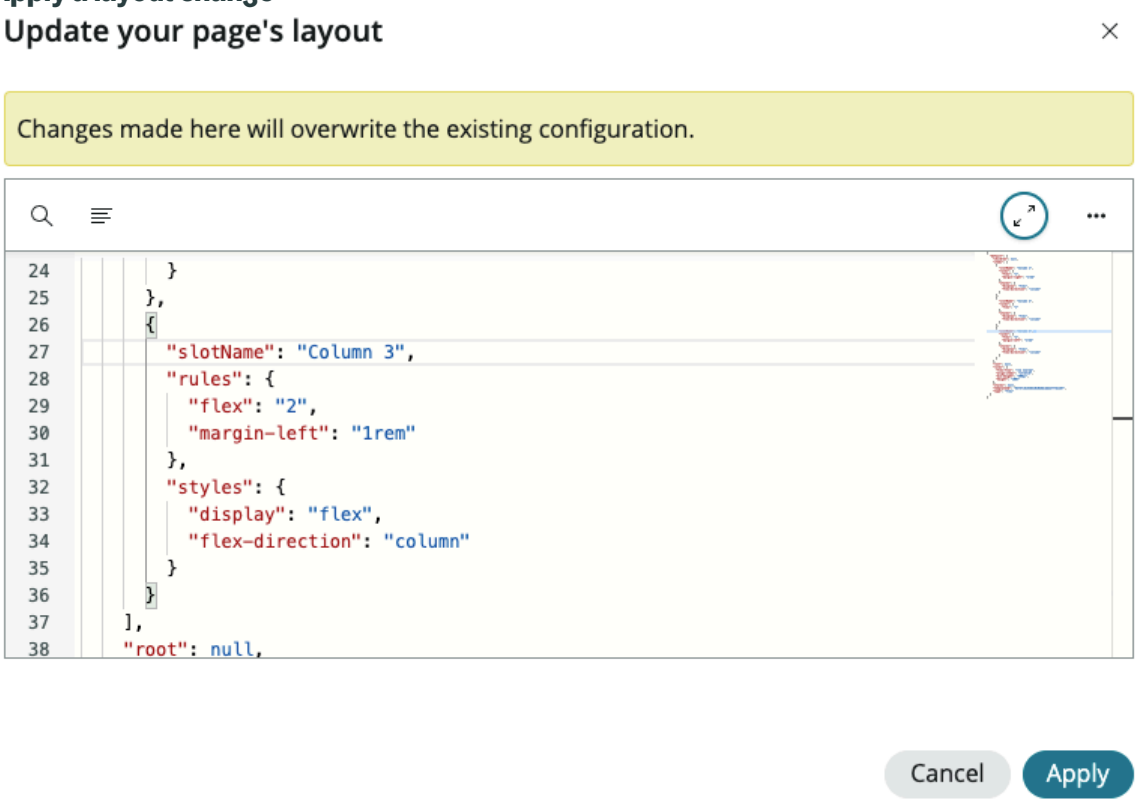
Paste the CSS code

```
        "flex": "1"
      },
      "styles": {
        "display": "flex",
        "flex-direction": "column"
      }
    },
    {
      "slotName": "Column 3",
      "rules": {
        "flex": "2",
        "margin-left": "1rem"
      },
      "styles": {
        "display": "flex",
        "flex-direction": "column"
      }
    }
  ],
  "root": null,
  "rules": {
    "flex-flow": "row nowrap",
    "align-items": "stretch",
    "min-height": "400px",
    "height": "100%"
```

Changing the *"flex"* property increases the size of the column. By using *"margin-left"*, you add space between the previous columns.

9. Collapse the expanded view, and then click **Apply**.

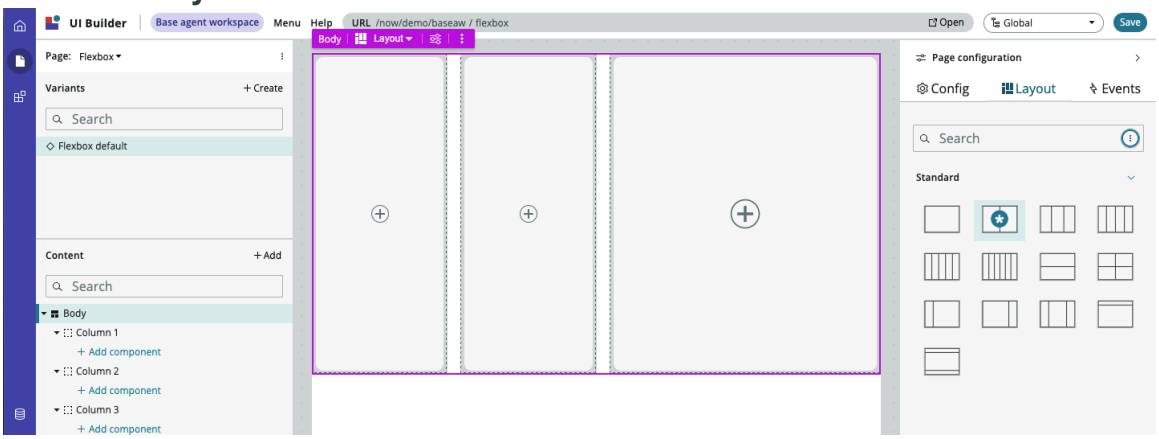
Apply a layout change



Result

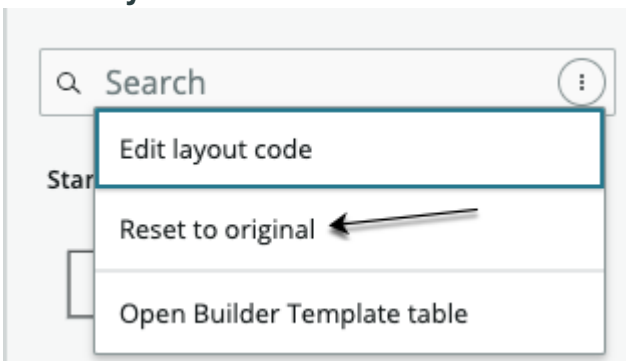
The new Flexbox layout that you created shows the new slot that you added to the original two-column layout.

New custom layout



Click **Reset to original** to reset layout changes back to the original.

Reset a layout



Related topics

Create a CSS Grid layout with the old layout system

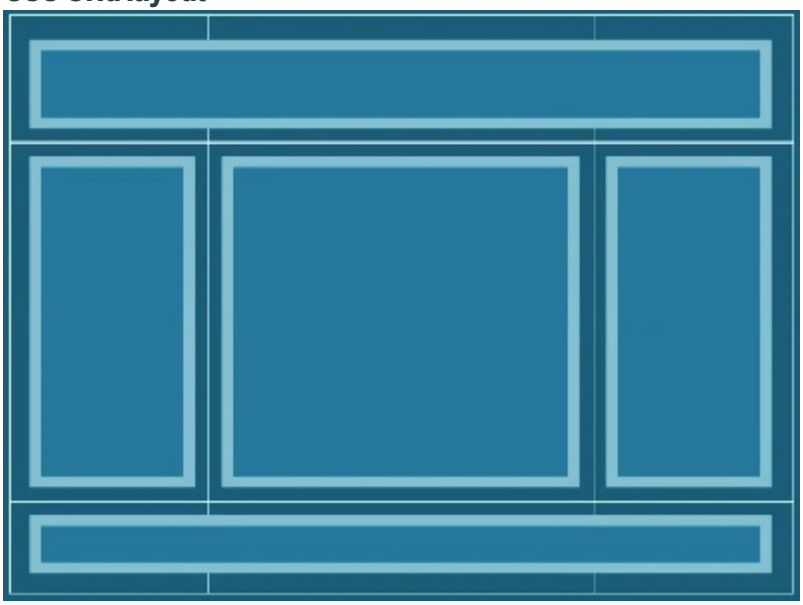
Using CSS Grid layouts to build a page

Create a CSS Grid layout in UI Builder to build powerful pages so that you can customize with cascading style sheets (CSS) and can improve your performance.

CSS Grid is the most powerful layout system in CSS. CSS Grid is built on top of a two-dimensional grid. CSS Grid gives you control over how you create your pages. Use UI Builder to implement a CSS Grid layout using low-code layout configuration properties. For advanced layouts, you can view and edit CSS code to customize your layouts. Visit Mozilla to learn more about CSS Grid.

For example, you can have a grid with three columns and three rows to make a grid nine cells. You can place components inside these cells or make the component span multiple cells.

CSS Grid layout



Customizing the layout of your page lets you take full advantage of CSS Grid so that you can achieve your overall page design. For more information, see Organize components in UI Builder pages.

To find out more about CSS layouts within your UI Builder instance, you can find them in the [sys_uib_template] table.

Create a CSS Grid layout with the new layout system

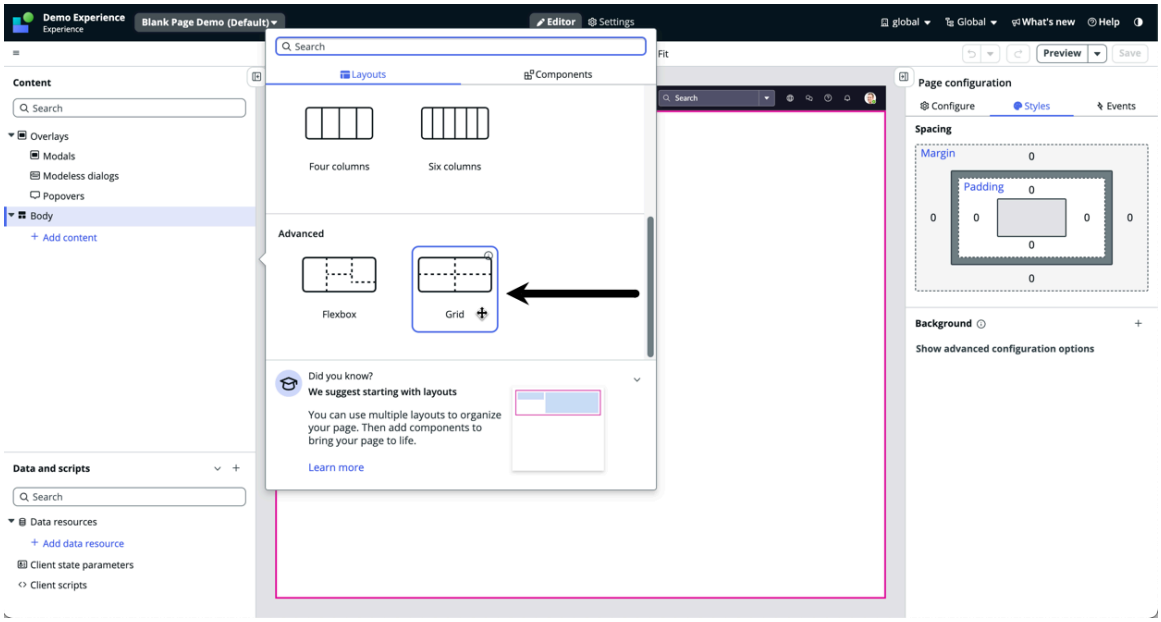
Create a CSS Grid layout in UI Builder to build powerful pages in a low-code environment.

Before you begin

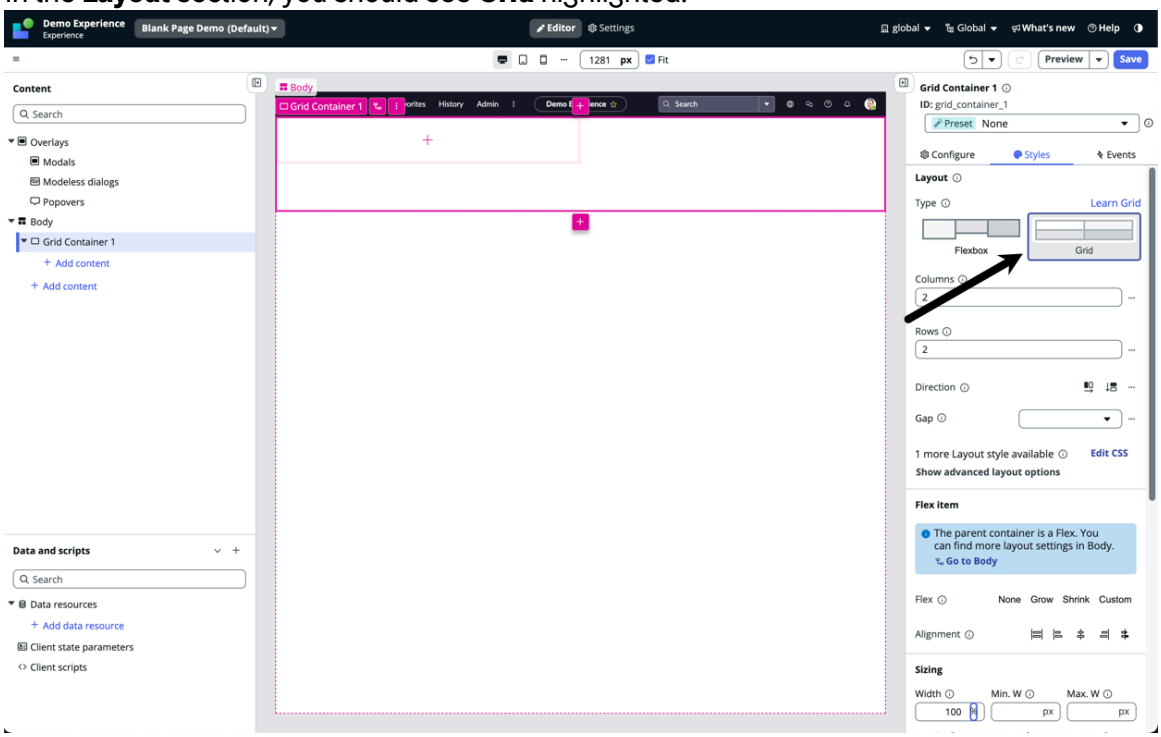
Role required: admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page in UI Builder.
4. Select the **+ Add content** button in the UI Builder stage.
5. Select **Grid** in the layouts tab.



6. Select the **Styles** tab in the configuration panel.
7. In the **Layout** section, you should see **Grid** highlighted.



8. In the **Layout** section of the styles tab, you can change the following:

Columns

Defines the number, order, and width of grid columns.

Rows

Defines the number, order, and width of grid rows.

Direction

Sets whether newly added components are stacked horizontally or vertically.

Gap

Sets the size of the gap between rows and columns.

Note:
Select **Show advanced layout options** to see the next four options.

Align items

Defines how components within the container align as a group.

Justify items

Defines how the browser distributes space between and around content items along the inline axis of a grid container.

Align content

Sets the distribution of space between and around content items along a grid's block axis.

Justify content

Defines how to pack or space apart the components within the container.

Visit Mozilla to learn more about CSS Grid layout configurations.

9. **Optional:** You can edit the CSS code by selecting **Show advanced configuration options** at the bottom of the **Styles** tab, and then selecting the **View and edit CSS** option.
10. Select **Save**.
    Your page refreshes and displays the changes made to the layout.
11. Add components to your page.
    See Add and configure components for more information.
12. View and test your page by selecting Preview.

Create a CSS Grid layout with the old layout system

Create a CSS Grid layout in UI Builder to build powerful pages so that you can customize with cascading style sheets (CSS) and can improve your performance.

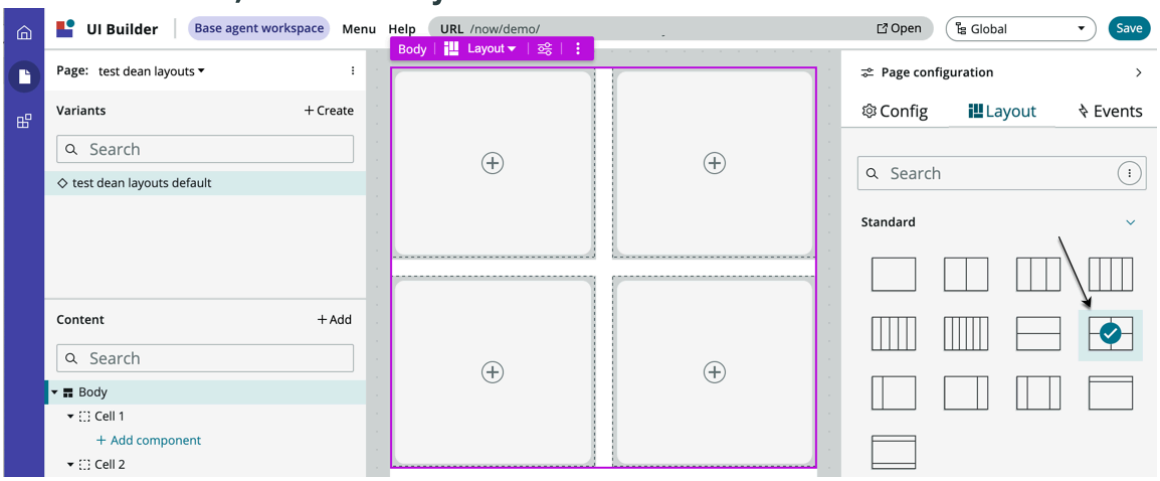
Before you begin
Role required: admin

About this task
In the following procedure, you learn how to use CSS Grid to modify your CSS to customize the layout of your page.

Procedure

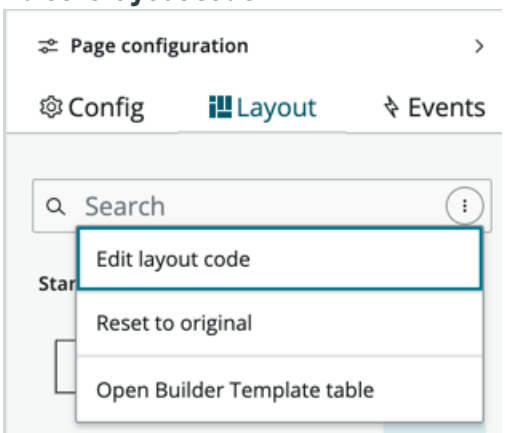
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page in UI Builder or open a page.
4. Click the **Layout** tab and choose the two-row by two-column layout.

Standard two-row, two-column layout



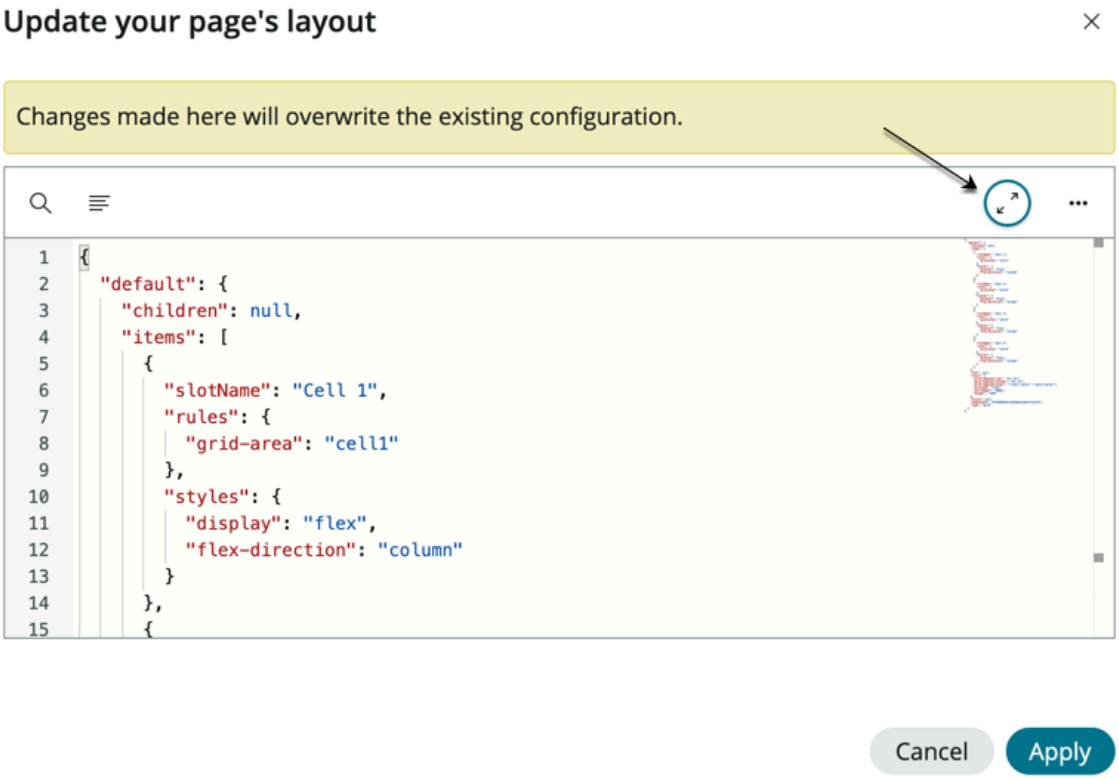
5. Click **Edit layout code**.

Edit the layout code



6. Expand the code editor so that you can easily view the CSS code.

Expand the code editor



7. Add another slot to the layout by copying the CSS code for an existing slot.

   Copying the CSS code is easier than typing it into the layout.

Copy the CSS code for a slot

```
23          }
24        },
25        {
26          "slotName": "Cell 3",
27          "rules": {
28            "grid-area": "cell3"
29          },
30          "styles": {
31            "display": "flex",
32            "flex-direction": "column"
33          }
34        },
35        {
36          "slotName": "Cell 4",
37          "rules": {
38            "grid-area": "cell4"
39          },
40          "styles": {
41            "display": "flex",
42            "flex-direction": "column"
43          }
44        }
45      ],
46      "root": null,
47      "rules": {
48        "grid-template-rows": "1fr 1fr",
49        "grid-template-columns": "1fr 1fr",
50        "grid-template-areas": "\"cell1 cell2\" \"cell3 cell4\"",
51        "grid-gap": "1rem",
52        "min-height": "400px",
53        "height": "100%"
54      },
55      "styles": null,
56      "templateId": "b719698953c31010e6bcddeeff7b1275",
57      "type": "grid"
58    }
59  }
```

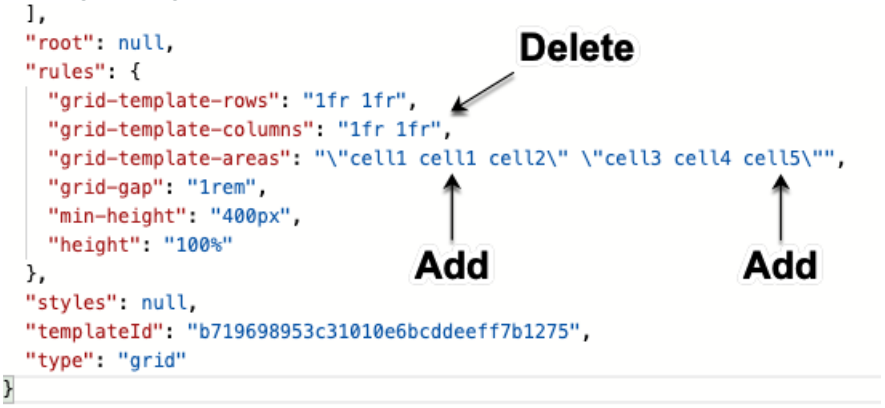
8. Paste the CSS code below the code that you copied and change the *`slotName`* and *`grid-area`* to a unique value.

Paste the CSS code

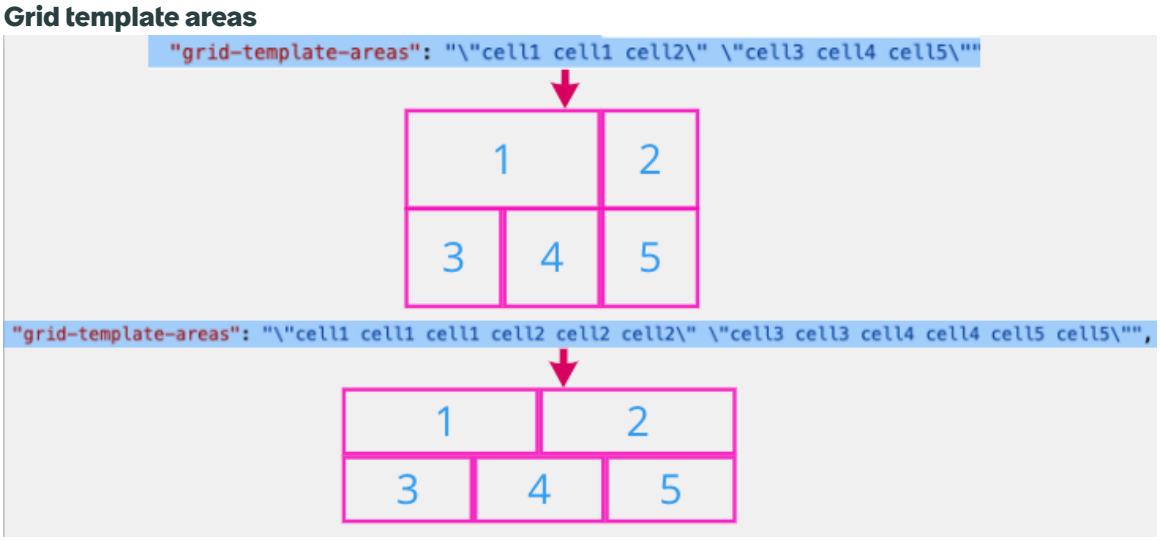
```
{
  "slotName": "Cell 4",
  "rules": {
    "grid-area": "cell4"
  },
  "styles": {
    "display": "flex",
    "flex-direction": "column"
  }
},
{
  "slotName": "Cell 5",
  "rules": {
    "grid-area": "cell5"
  },
  "styles": {
    "display": "flex",
    "flex-direction": "column"
  }
}
```

9. Modify the layout rules to include the new column and template area, as follows:
   a. Delete the `"Grid-template-columns": "1fr 1fr",` line.
   b. Add a second `cell1` after `cell1`, and then add `cell5` after `cell4`.

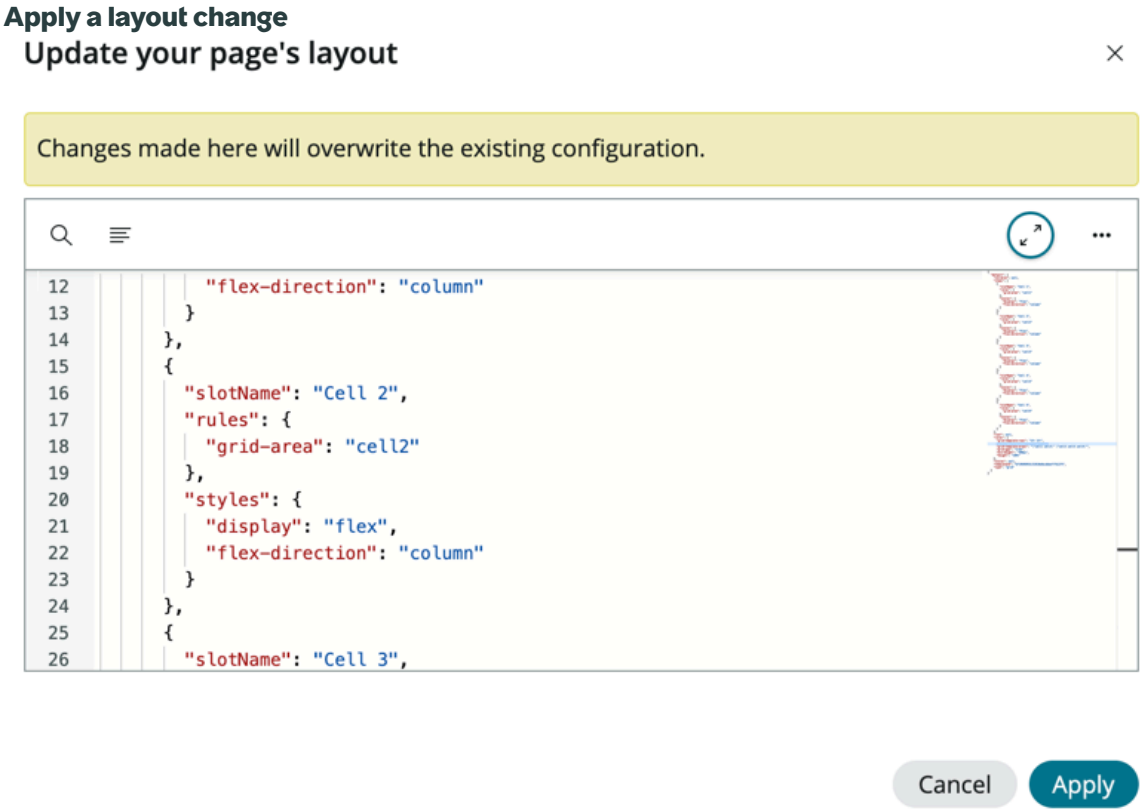
Modify the layout rules



You can set many different configurations. In this example, there are the two resulting grids from two different grid template areas. The *"grid-template-areas"* property being modified is setting the CSS property of the same name. For more information, see MDN grid-templates-areas.



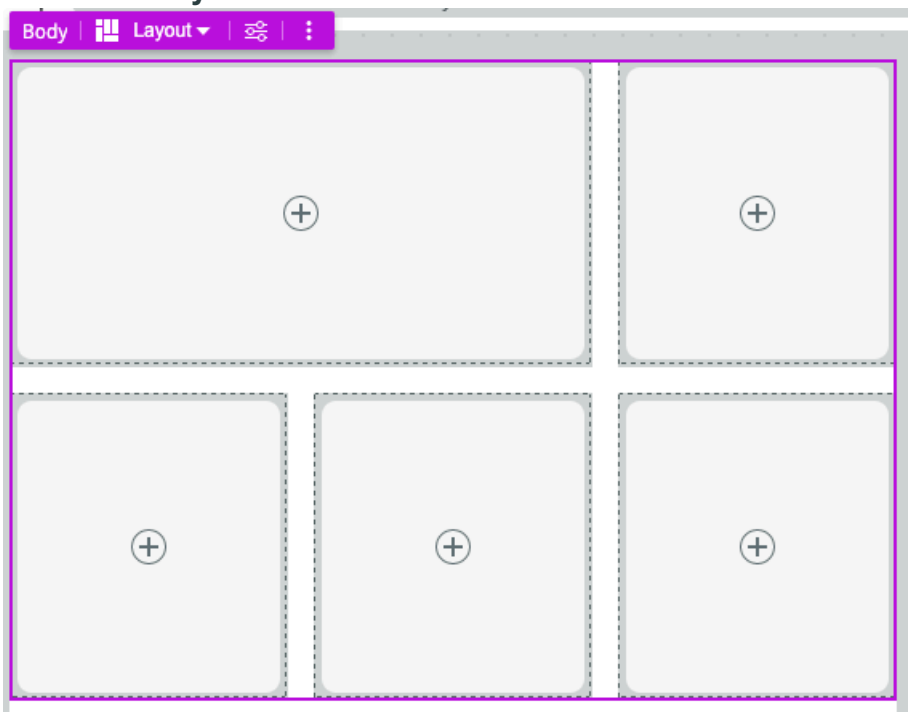
10. Collapse the expanded view, and then click **Apply**.



Result

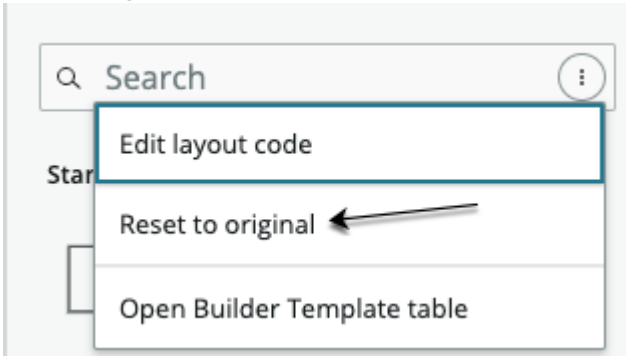
The new CSS Grid layout shows that the new slot was added to the original two-row, two-column layout.

New custom layout



Click **Reset to original** to reset the layout changes back to the original at any time.

Reset a layout



Related topics

Create a Flexbox layout with the old layout system

Change the layout of a page created in Quebec or Rome

Add and modify your layout design to change the way your page looks. Choose how components are displayed on a page through Cascading Style Sheets (CSS) web layout technologies, such as Flexbox and CSS Grid.

Before you begin

Role required: ui_builder_admin

About this task

Layouts control what containers and components are available on a page, and where they are. CSS rules apply to them. You can change the layout as follows.

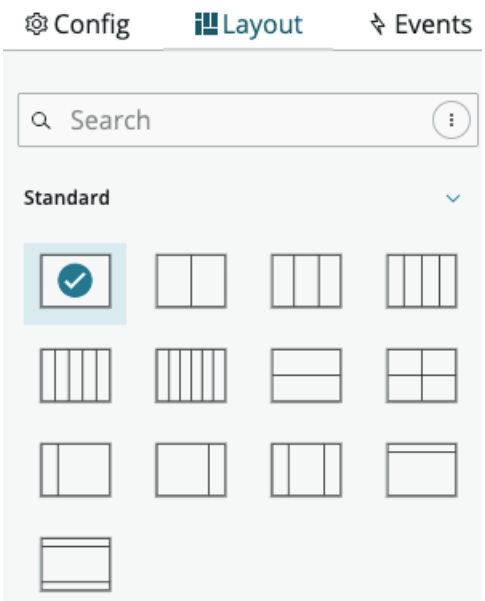
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page.
4. Select the **Layout** tab.
5. Depending on the type of page, do one of the following to select a container to update. If you are starting a new page, you can select the layout at the page level, and later at the container level. If you are not the owner of an existing page, be aware of the impact of changing the layout at the page level.
6. Change the layout of your container.

   a. In the Page configuration pane, select the **Layout** tab to select the layout that you want to use.

      You can change the current layout of an existing page into a new one. For example, in a three-column layout, you can click the four-column layout to change the layout.

      You can also set the layout for a container. The following image shows the layout options for the UI Builder.



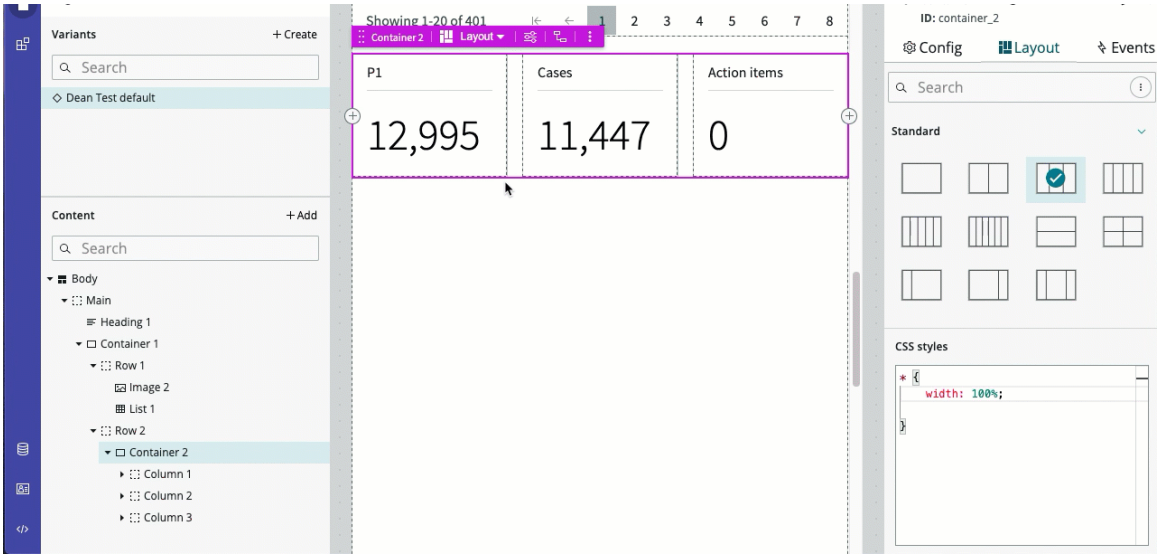
   b. If you are creating a page, add components to the new areas in your layout.

   c. Click **Save**.

   d. View and test your page by selecting Preview.

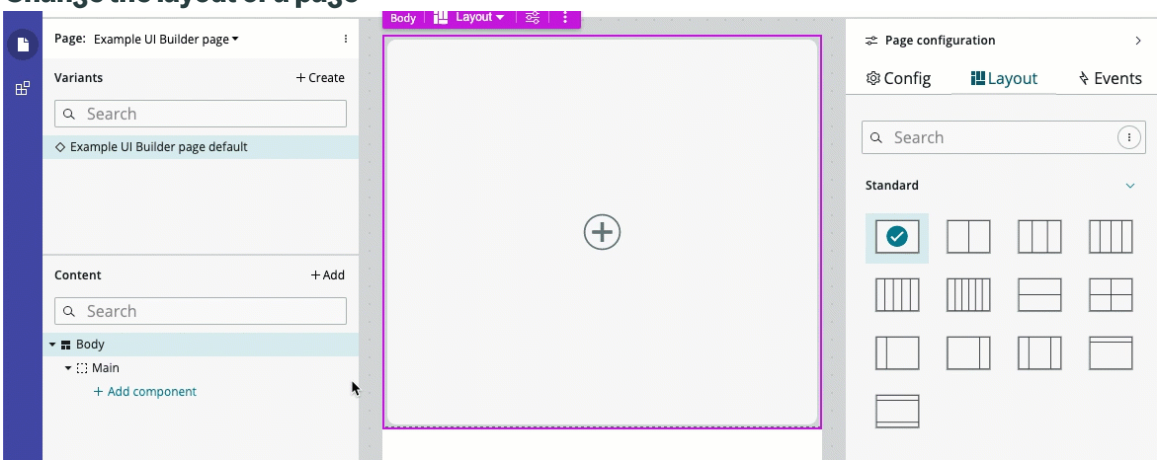
Example

For changing an existing page, the following video shows how you can change the container layout from three to four columns, and then add a new component to the fourth column.



For setting the layout of a new page, the following video shows you how to set the layout that you want for the page. For example, you can set your page to have two slots or columns. Then, you can add containers to each slot and change the layout for each container. You can also set the layout at the container level after you add containers to your page.

Change the layout of a page



7. Add components to slots.
   You build your page with containers components. See Customize UI Builder pages using components for more information.

   You can add components by using any of the following ways.

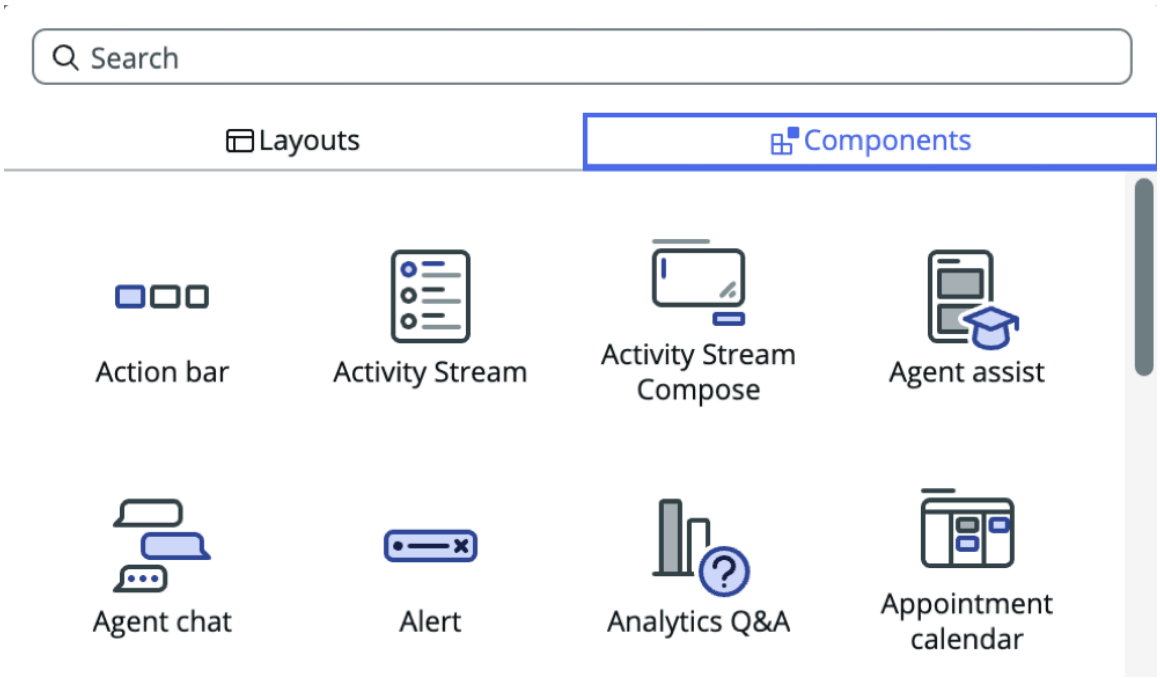
8. Modify the layout styling options in any of the following ways.

Customize UI Builder pages using components

Learn what a component is in UI Builder. Also, see how components work within UI Builder.

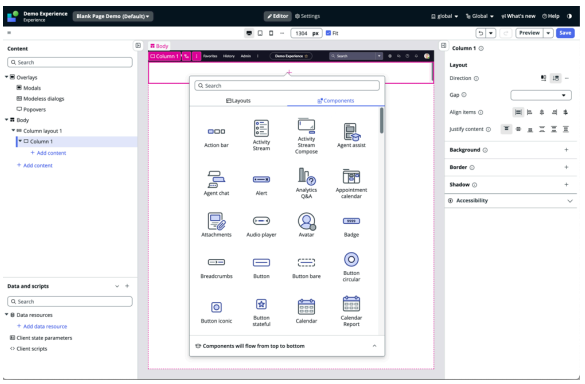
Components are the base elements of your UI Builder page. Components range from core elements like buttons and labels to more complex experience components like lists and forms.

You can add these components to your UI Builder page to build or customize your workspace or portal experience. For example, adding an **Activity stream** component to your page that lets users see their travel request activity.

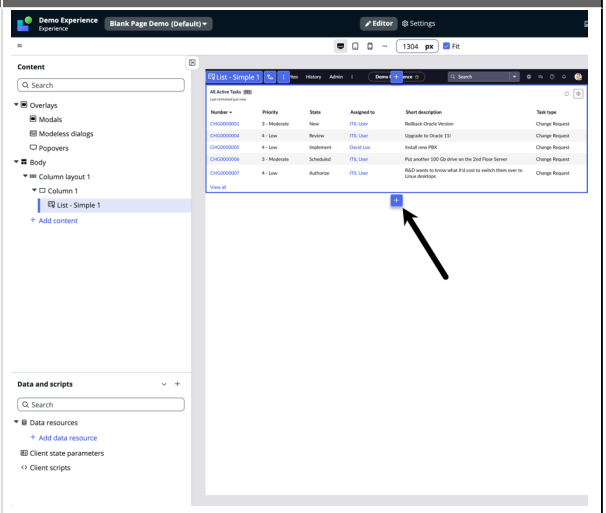
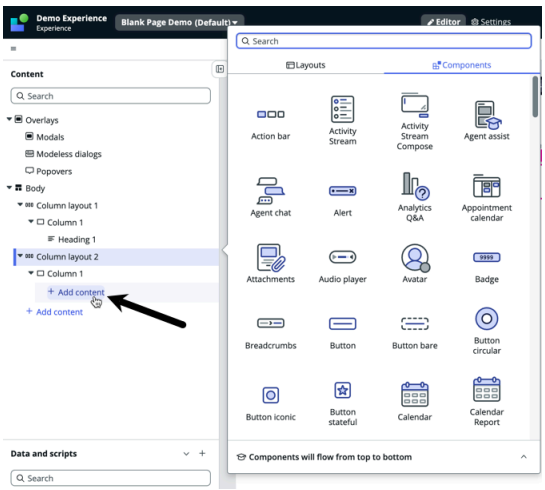


You can add components to your UI Builder page in the following ways.

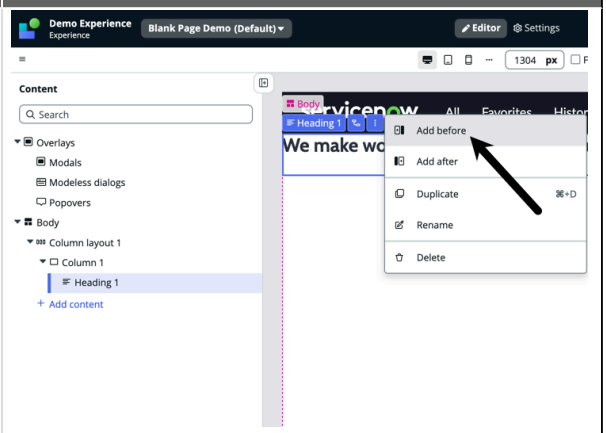
Ways to add a component to a page

Location	Example
Directly from a page in UI Builder (option 1)	Select the add content (plus) icon, select the **Components** tab, and then select a component. 
Directly from a page in UI Builder (option 2)	On a column layout or component, select the add before (plus) or add after (plus) icon, and then select a component.

Ways to add a component to a page (continued)

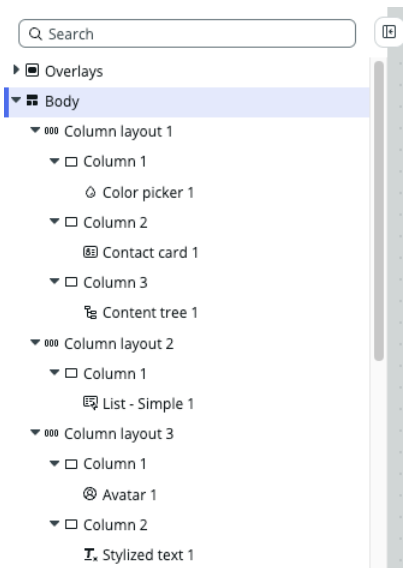
Location	Example
	
From the **Content** tree in UI Builder	In the content tree, select **+ Add content** and then select a component. 
From the floating menu above the page in UI Builder	On a column layout or component, select the Menu icon, select **Add before** or **Add after**, and then select a component.

Ways to add a component to a page (continued)

Location	Example
	

Column layouts in UI Builder

Add column layouts and columns to your UI Builder page first. Then add components to the columns in a column layout to build and customize your page. Think of a column layout as a defined part of the screen where you add components like lists and headings. You can have as many column layouts on a UI Builder page as you want, with as many as six columns in any column layout. Multiple components can be added to a single column. View the structure of your page in the **Content** tree.

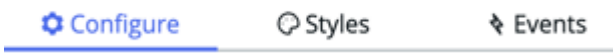


For more information, see Column layouts.

Configure components in UI Builder

There are three ways to configure a component in the configuration panel.

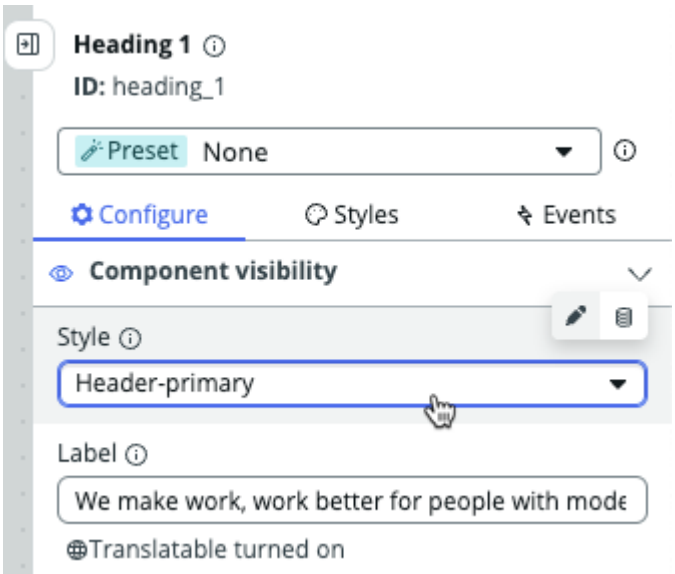
- Configure the component properties.
- Add CSS style overrides.
- Set up event handlers for the components.



Configure tab

Component properties vary based on each component. Component configuration can be simple, as is the case with simple elements like buttons, headings, and labels. Components like lists and forms require significant configuration. For icon and image components, select from a variety of options or use a custom icon or image. You can modify component properties with UI Builder's low-code JSON editor. For more information about configuring components, see Next Experience Components.

For each property in the component configure tab, you can choose from the following options.



- Static: Use data from a list, or enter your own data. The data doesn't connect to an external data source.
- Dynamic data binding: A way to bind a component property to a data resource, page property, or client state.
- Script: Enter JavaScript code to populate a property value.

Styles tab

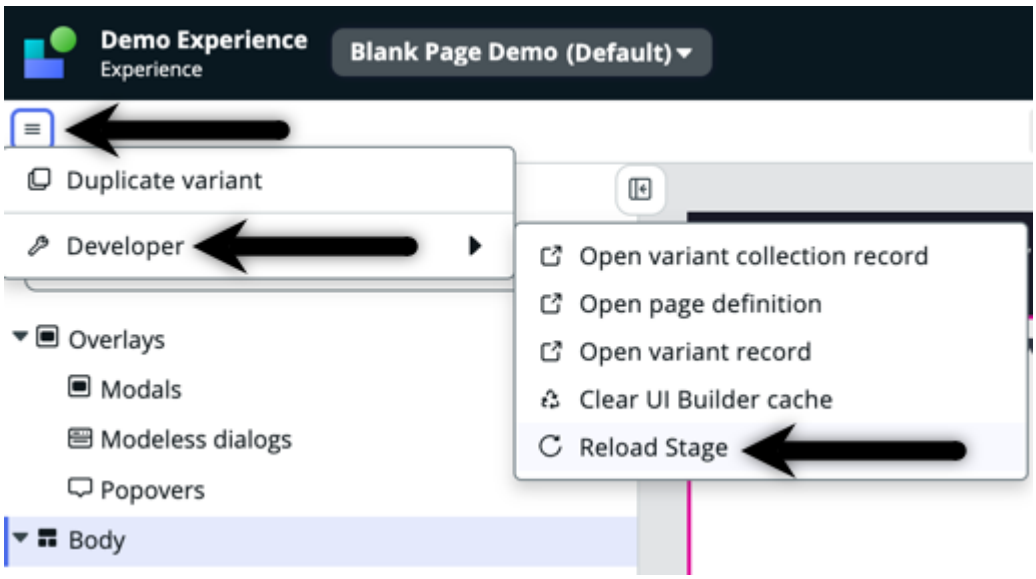
You can change the Cascading Style Sheets (CSS) styles for individual components. Change color borders, font size, and so on.

Events tab

Configure page-level and variant-level event mappings. Also add dispatched events and handled events for your variant. The more complex, experience components rely heavily on dynamic data that is provided by a data resource. Binding dynamic data to a component is an important feature. You dynamically expose data from tables, records, or other elements on your page. Exposing data enables you to reuse your components. Also, you can point to the configuration fields to see icons data or scripting options for each field.

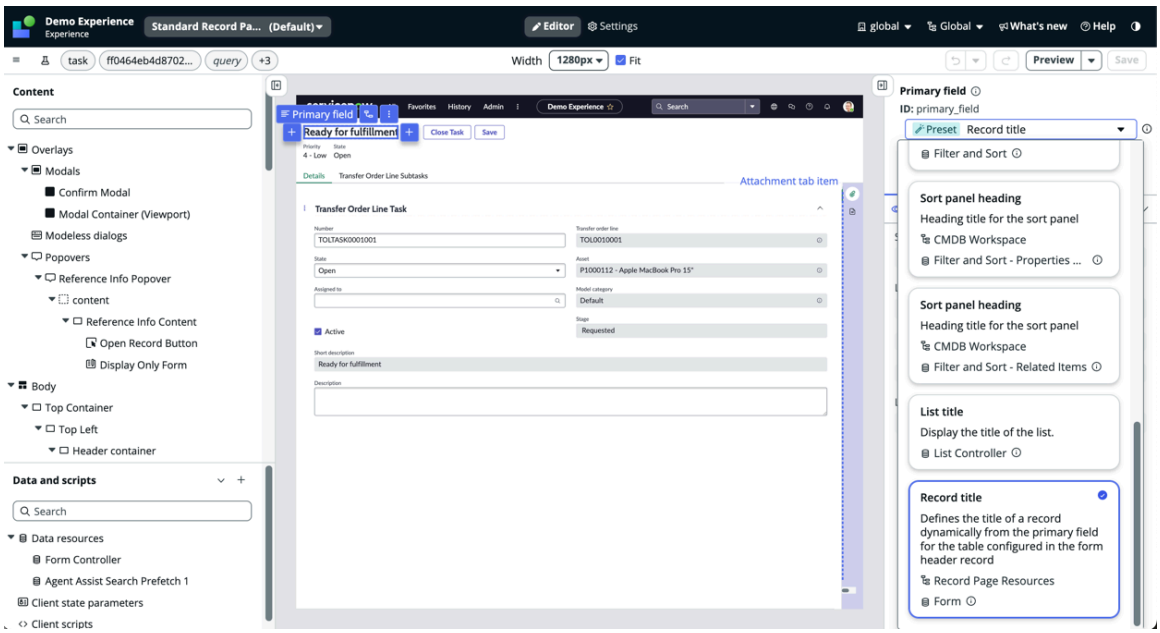
As you add and configure components on the page, the stage shows your work. If you make changes and the updates do not load on the stage, select the **Open menu** icon and then select

Developer > Reload Stage. Reloading the stage shows your changes, but does not save them. To save your work, select **Save** at the top right.



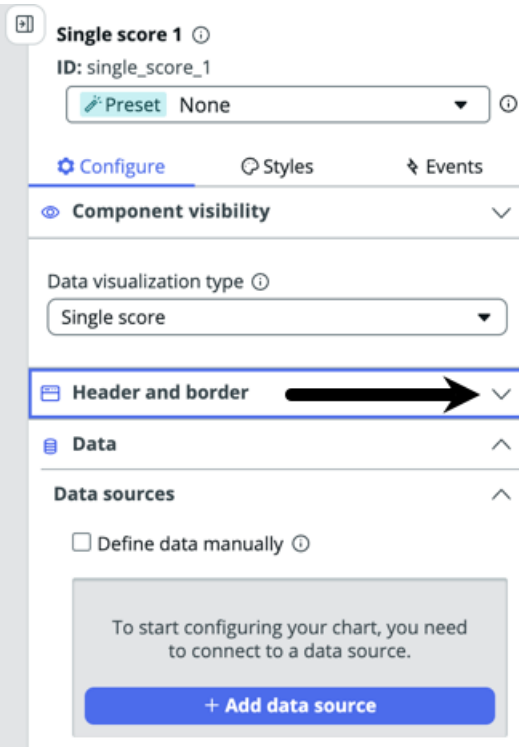
Component Presets in UI Builder

Use component presets to automatically configure components on compatible pages. UI Builder page templates contain controllers that presets use to define component configuration values. For more information, see Automatically configure components using presets.



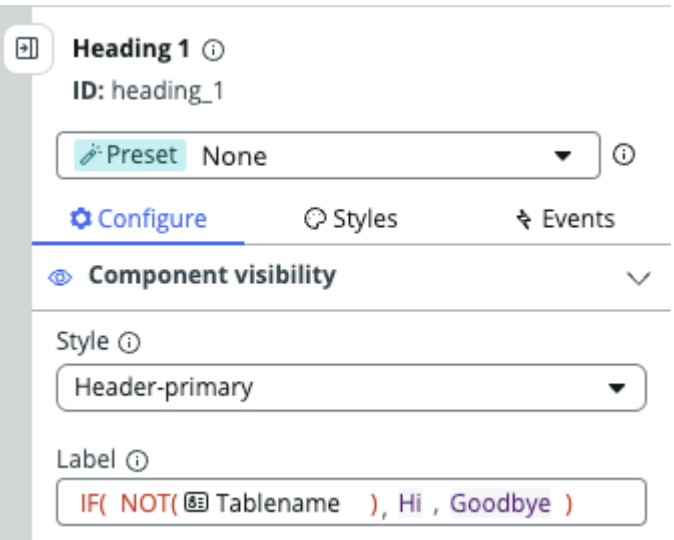
Component properties sections

Component properties are grouped into sections of similar properties. Open and close sections by selecting the arrows to the right of the property headings. UI Builder admins can select which component configuration properties are displayed or hidden based on UI policies.



Component formula editor

Use the UI Builder component formula editor in the configuration panel to bind or modify formulas. You can enter text, data bindings, and/or formulas. Formulas can consist of any combination of the three input types. The component formula editor supports logical, arithmetic, comparison, negation, and functional transform types.

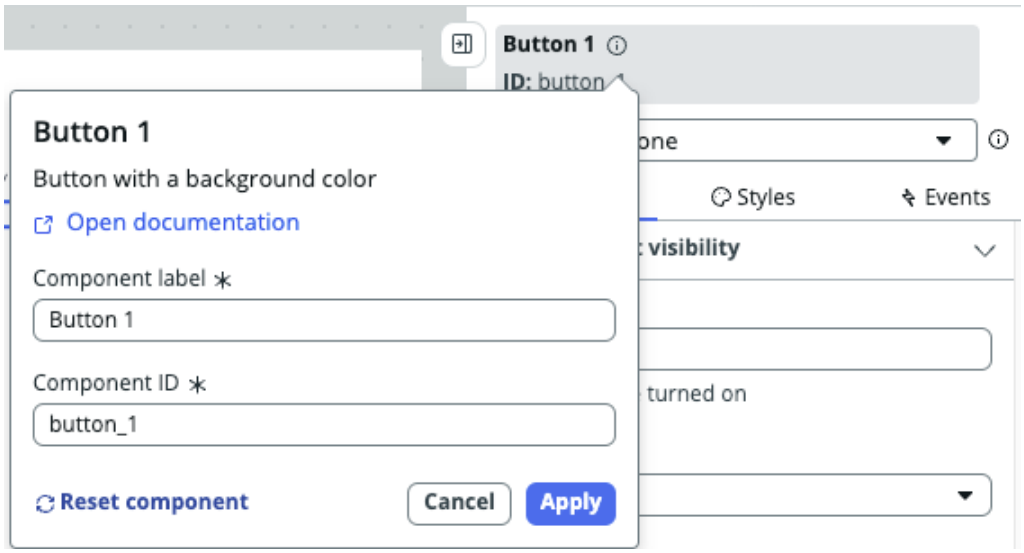


Each function added to the component formula editor auto-completes. The component formula editor displays a label for each input of the function so you know what parameters each function requires. For more information about the supported functions in the component formula editor, see Supported functions in the UI Builder component formula editor.

Component ID

Use the component ID when you add a script or bind data to the component as a way to reference the component. A component ID is automatically created and is based on the

component label when you add a component to a page. You can change the component ID to anything you want, as long as it is unique. Select the name of the component in the configuration panel to see the component ID.

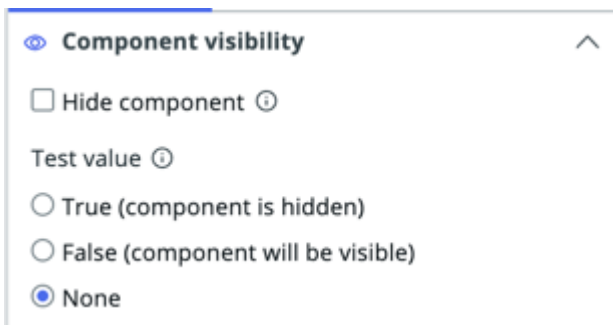


Component visibility

Select the eye icon in the configuration panel to set component visibility. Component visibility is based on a property of the component itself, not who is viewing it. You could show or hide a component based on conditions. For example, hiding an image if it has a broken link.

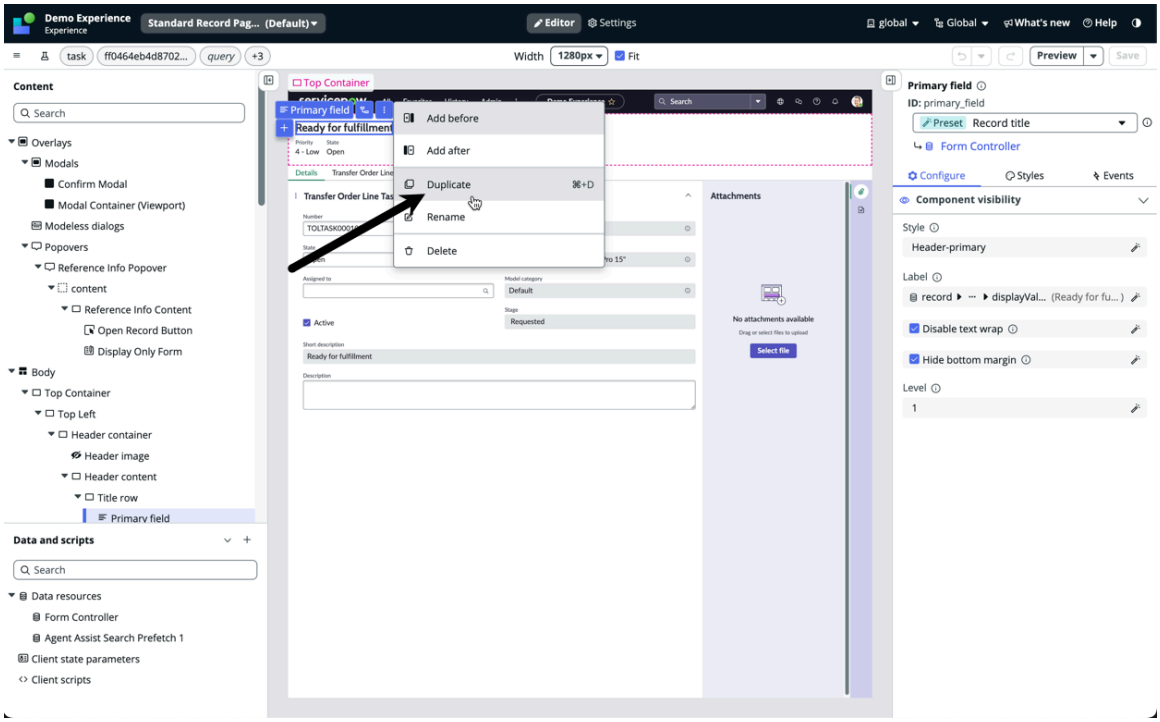
You can set the visibility based on dynamic data binding, or by editing a scripted property value.

Set the **Test value** to test what happens when the visibility is set to true, false, or none.



Duplicate components

Create an exact copy of a configured component on your UI Builder page except for the name and ID. A duplicated component copies all properties, bindings, and events. For more information, see Duplicate a component.



Automatically configure components using presets

Use component presets to automatically configure components on compatible pages.

Use component presets to apply predefined configuration values and event mappings to components. UI Builder page templates contain controllers that presets use to define component configuration values.

Presets apply pre-built configurations to component properties and event handlers and are only available for certain components. They are based on common use cases for components, such as configuring a **Form** component with fields that are typically included on a record page.

Applying a preset automatically configures components to work immediately, simplifying page creation and maintenance for page authors.

Presets offer the following benefits to page authors:

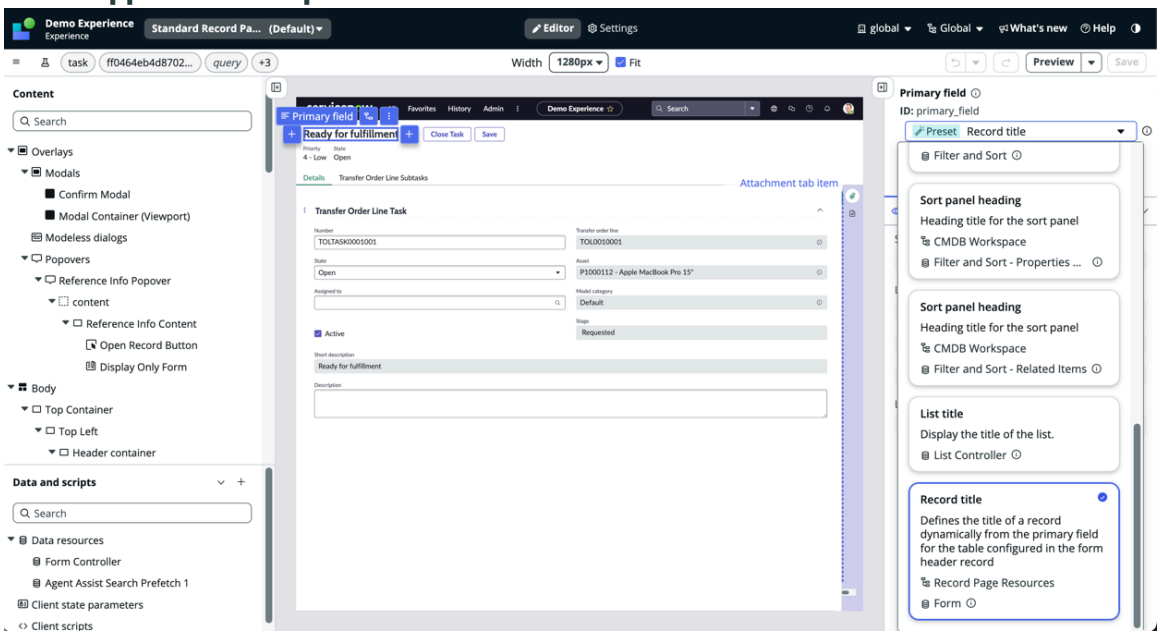
- Handling of complex data and event binding for components
- Reduced cost of ownership and maintenance due to being defined external to pages

From the Configuration pane, you can select whether to apply a preset to the selected component or instead manually configure it. Default presets are automatically applied for components when a page is created from a template or the controller required for the preset is already added to the page. Any properties or events configured by the preset display as read-only when a preset is applied.

You can override values configured by a preset but in doing so you assume ownership of the component configuration and maintenance.

Sub-pages do not inherit controllers and are not able to use presets in Xanadu.

Preset applied to a component



Data and event bindings

Presets can include bindings to:

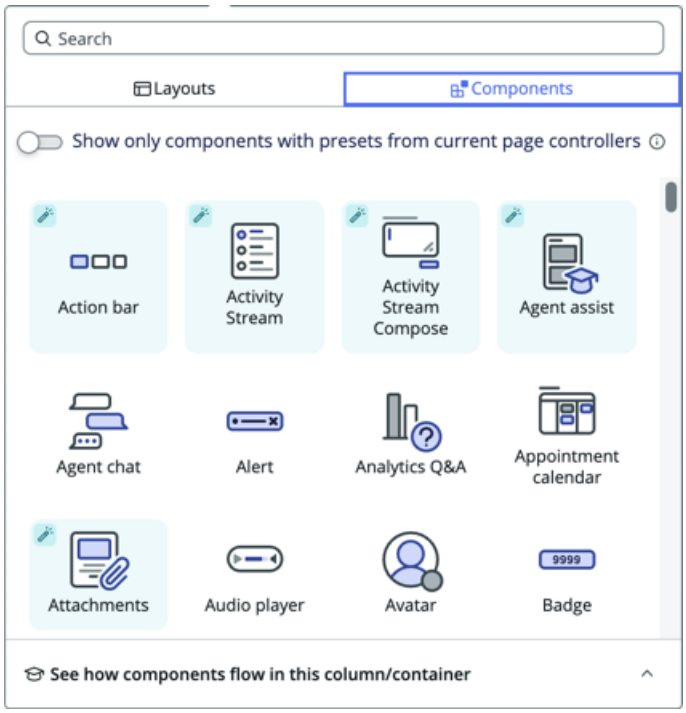
- Controller output properties (@data)
- Event payloads (@payload)
- Session context (@context.session)
- Complex formulas (client transforms)

Presets can also include event mappings to a controller's handled events. For more information, see Manage actions in UI Builder pages.

Controllers

Presets connect components to data and event mappings using a controller. If the controller required by a preset is not already on the page, the preset prompts you to configure the controller's required properties and adds the controller. After a controller is added to a page, components that have presets associated with the controller appear highlighted in the components list. For more information, see Bind data to UI Builder pages using controllers (advanced feature).

Components with presets



Select a component preset

Choose a component preset when adding a component to your page.

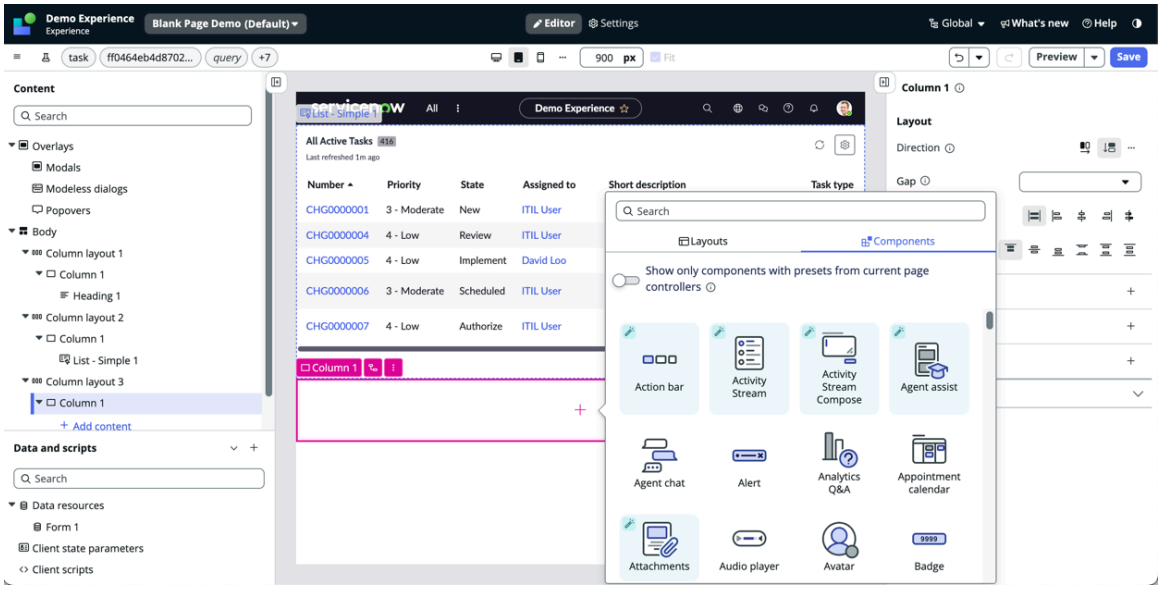
Before you begin

Role required: admin

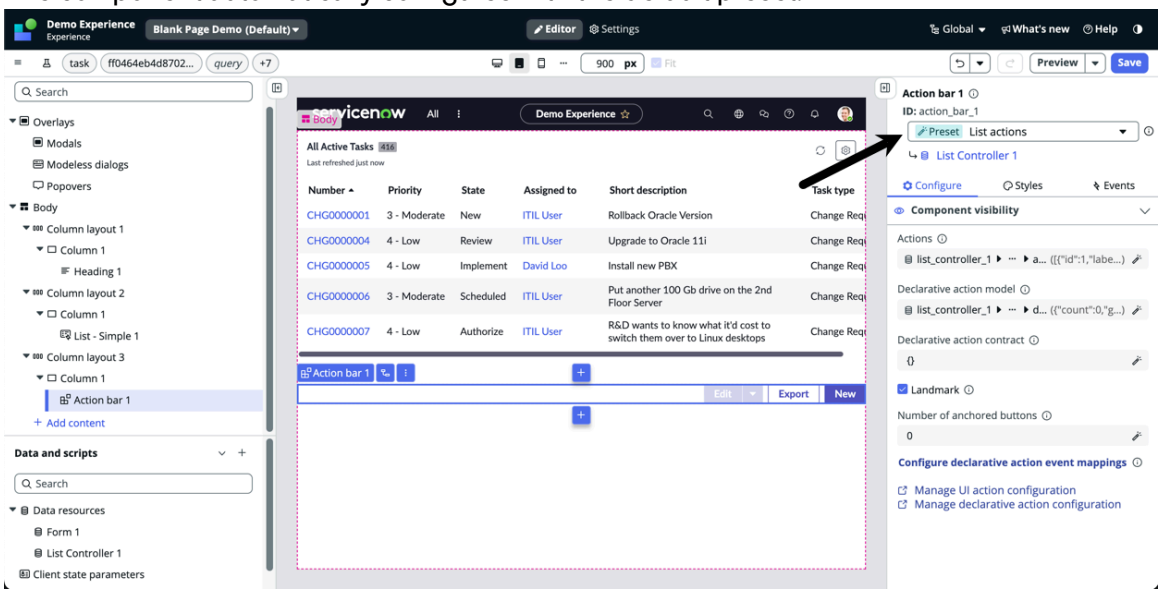
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Open the toolbox.
5. Add a component highlighted in green to your page.

   Components highlighted in green have presets available.

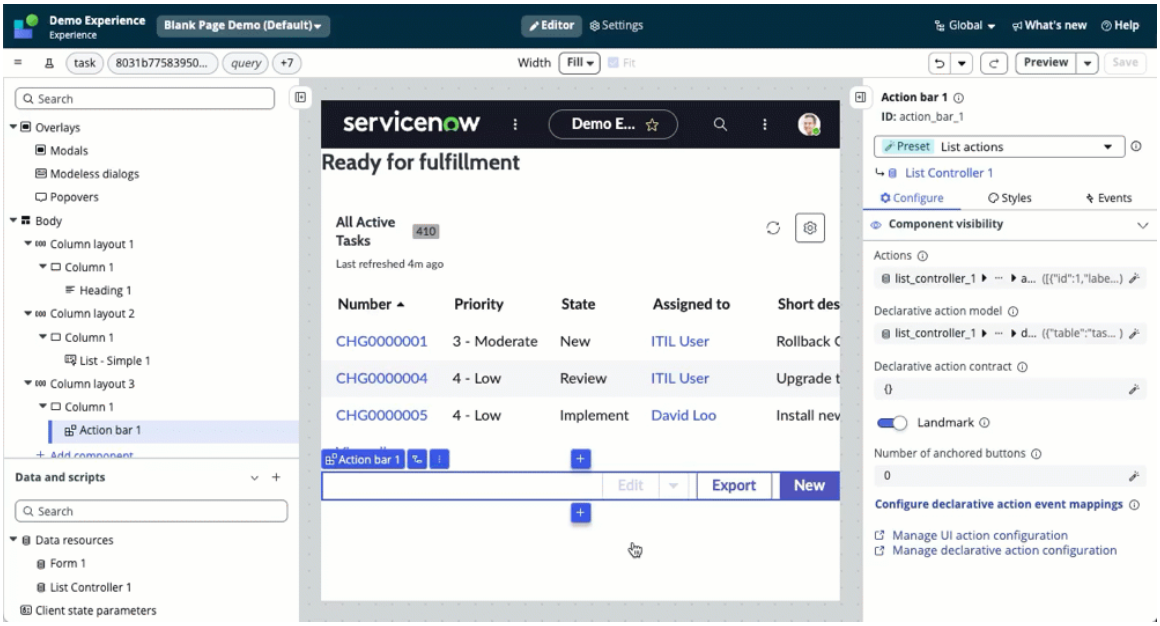


6. The component automatically configures with the default preset.



7. **Optional:** Select a different preset from the list in the configuration panel. You can configure a component not to use a preset by selecting **Remove**.

   a. Select **Continue** to replace the existing preset.



8. Click **Save**.

Override a component preset

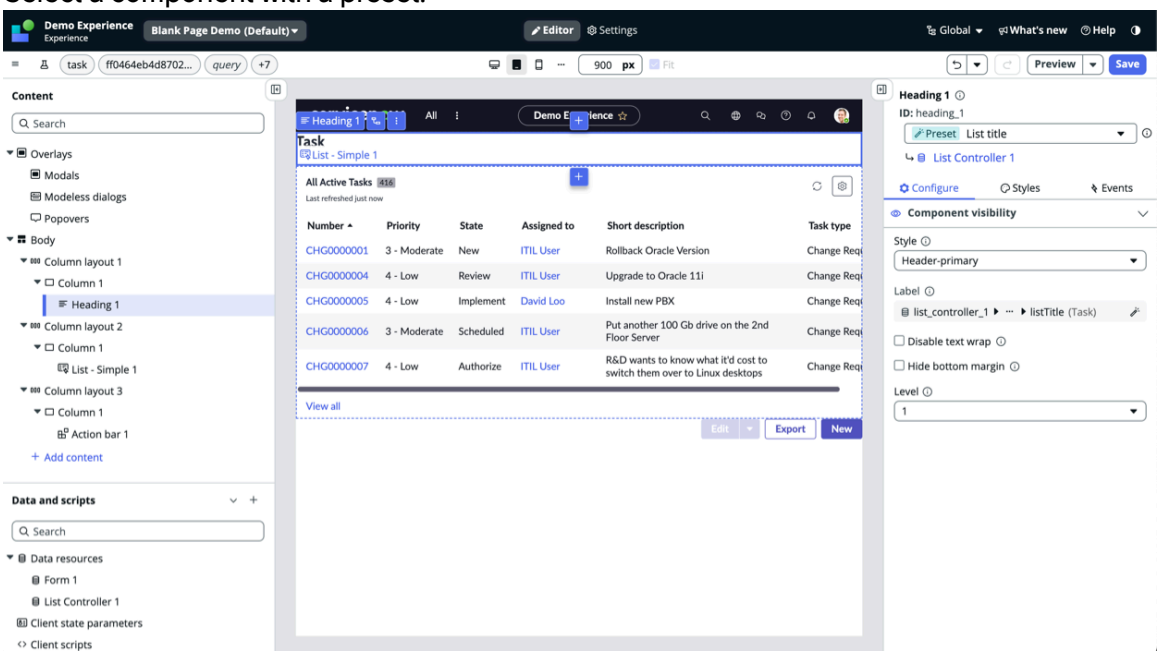
Override a component preset to enter your own custom values.

Before you begin

Role required: admin

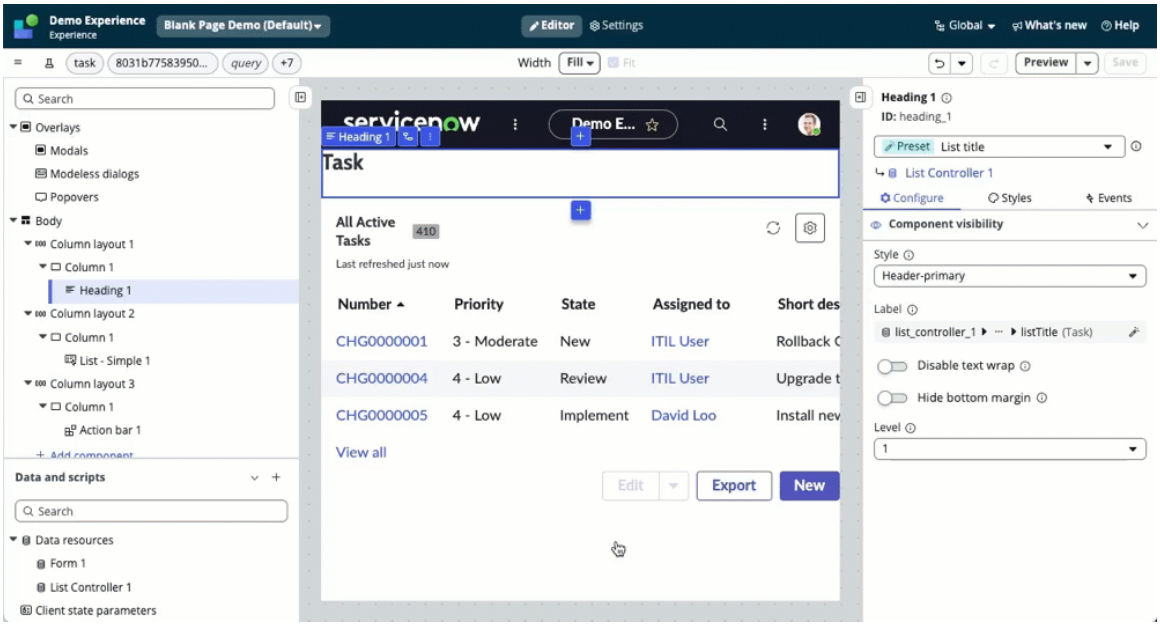
Procedure

1. Select a component with a preset.



2. Find the data value that you want to override in the configuration panel.
3. Click the lock icon in the data field.
4. Enter the data value that you want to override the preset value.

5. Select **Save**.



Result

The configuration panel displays the custom value in the field.

Reset a component

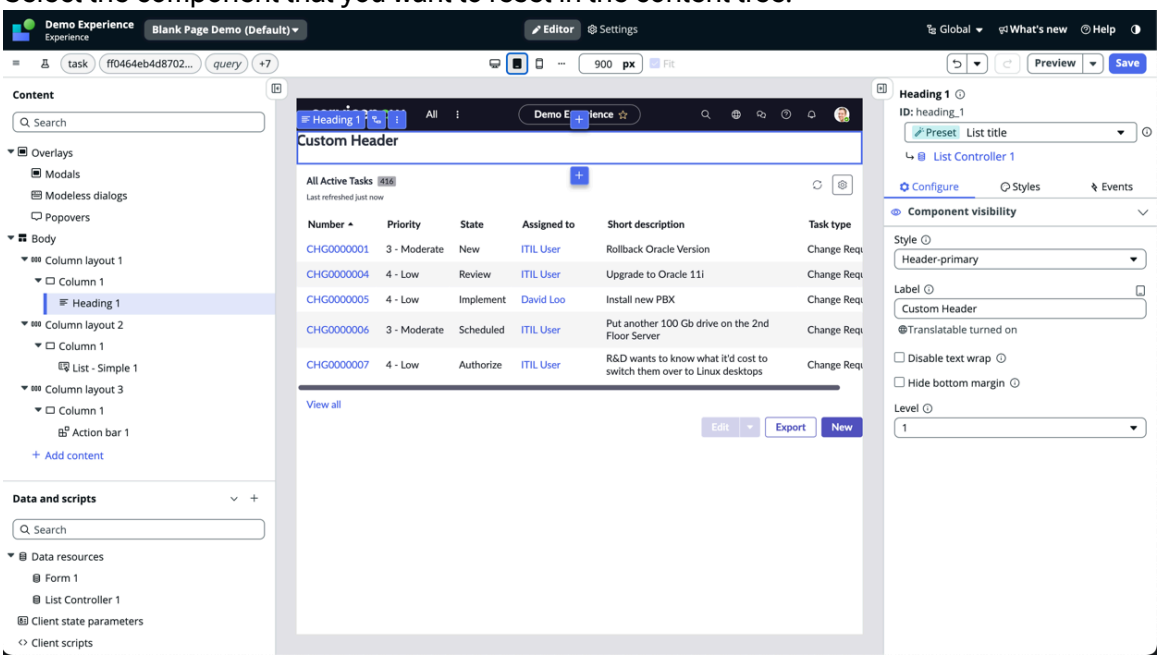
Reset component presets to their default values.

Before you begin

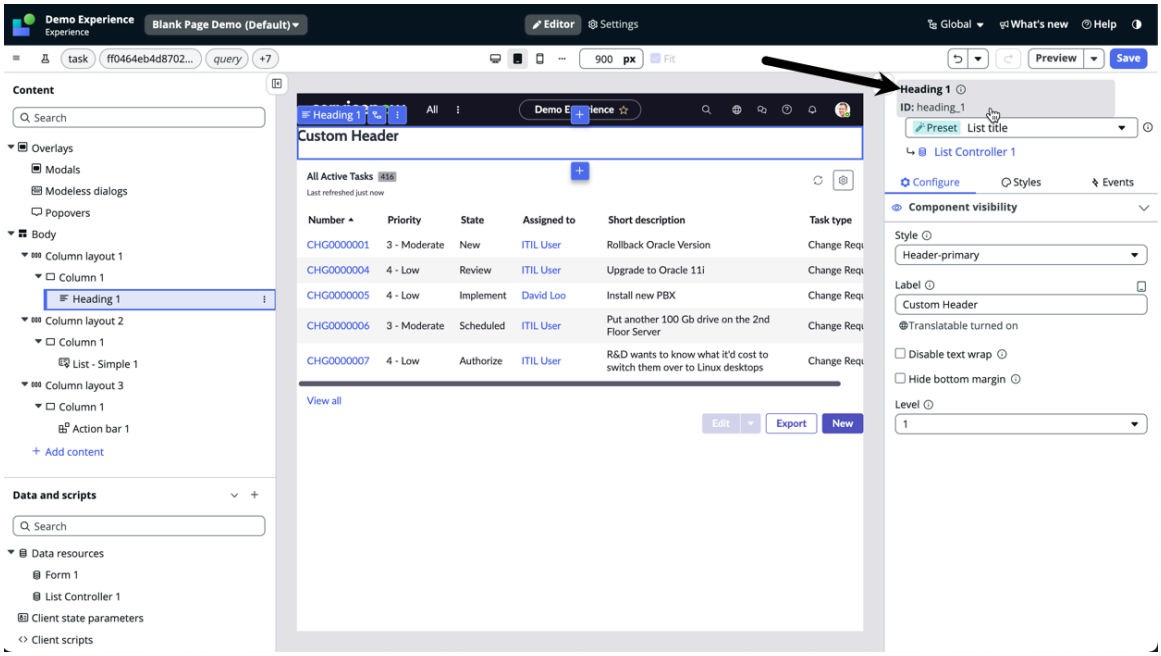
Role required: admin

Procedure

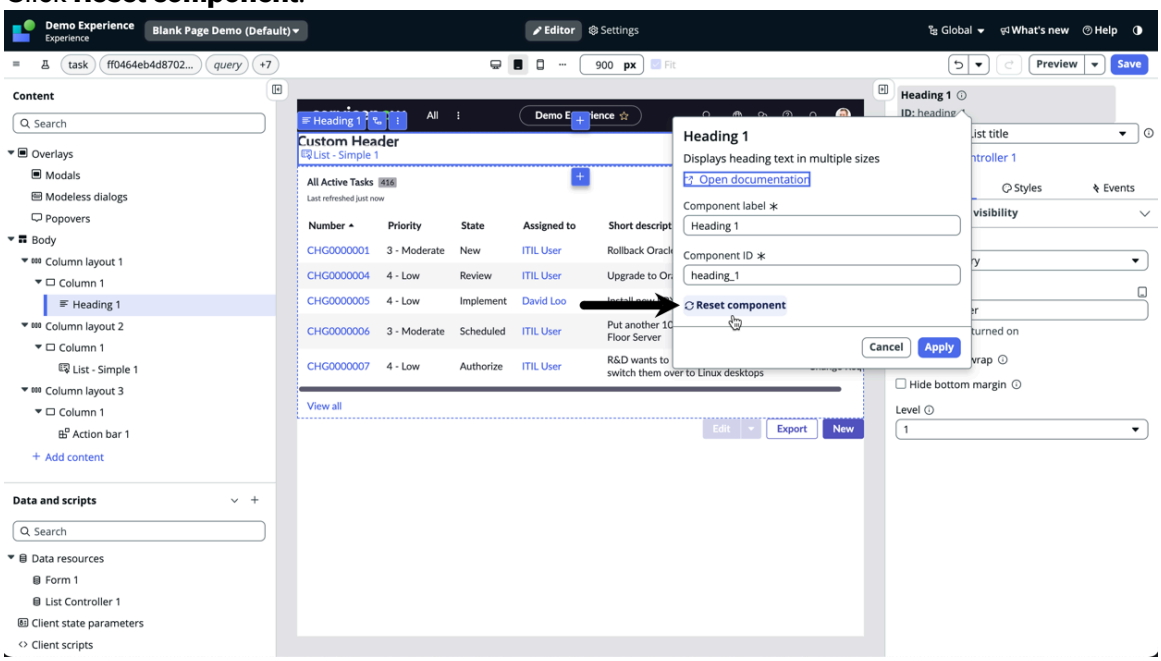
1. Select the component that you want to reset in the content tree.



2. Select the component name in the configuration panel.



3. Click **Reset component**.



4. Click **Reset**.

Result

The component preset resets to the default values.

Change the default appearance of components

Set the styles for components and wrappers to change the default appearance.

This video shows you how to perform the following procedure. This video shows you how to change the default appearance of components on a UI Builder page.

Before you begin

Role required: ui_builder_admin

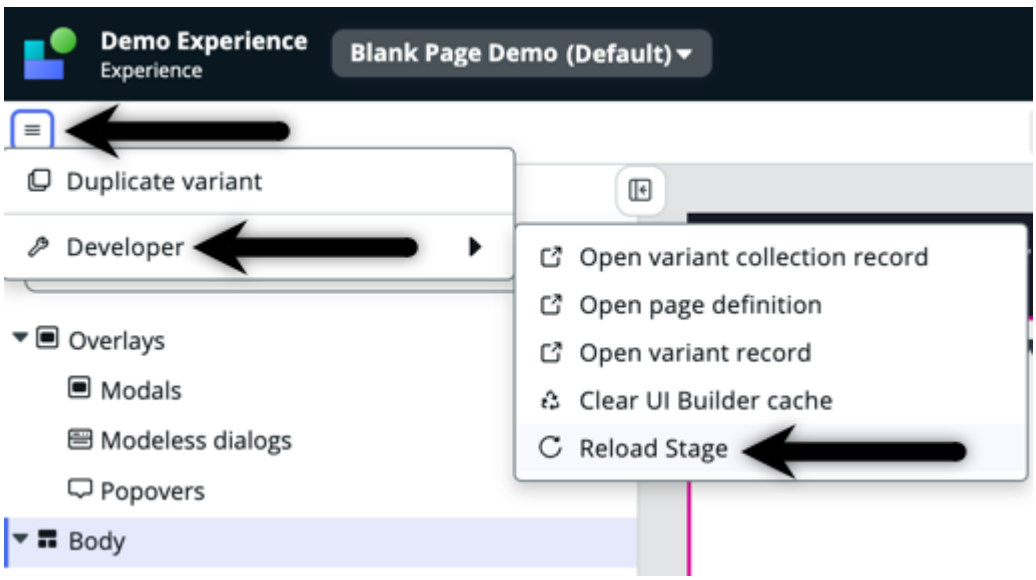
About this task

This task describes how to add styles to components and the wrappers (such as Body, column layouts, or a column) containing components. Customize component styling by selecting the component in the Content tree or apply styles to multiple components by placing components inside a wrapper.

For information about adding styling to your entire experience, see Manage the visual style of UI Builder experiences.

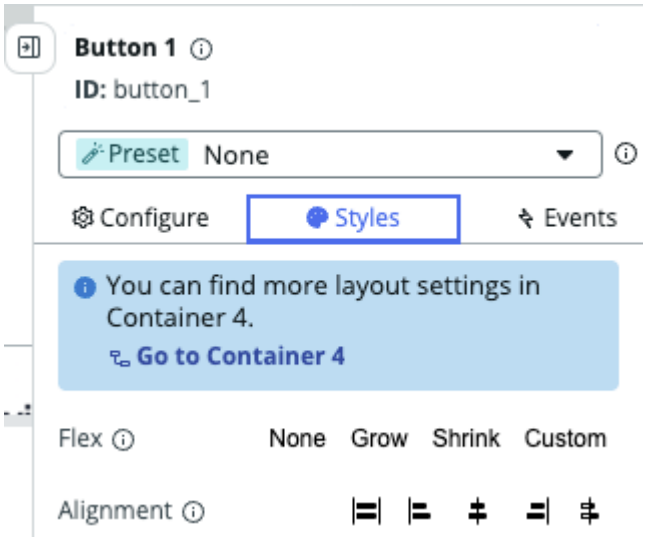
This task applies to the new layout system introduced in Xanadu. If your page is using the old layout system, see Add styling to a component using the old layout system for more information.

Note:
As you add and configure components on the page, the stage shows your work. If you make changes and the updates do not load on the stage, select the **Open menu** icon and then select **Developer > Reload Stage**. Reloading the stage shows your changes, but does not save them. To save your work, select **Save** at the top right.



Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. In the Content panel, select a component or wrapper (for example, Body, a column layout, or a column) for which you want to add styling.
5. For Body or a component, select the **Styles** tab in the configuration panel.



6. Use the visual representations on the **Styles** tab to decide what is best for the look and feel of your page.

Accessibility (Column layout and Column)

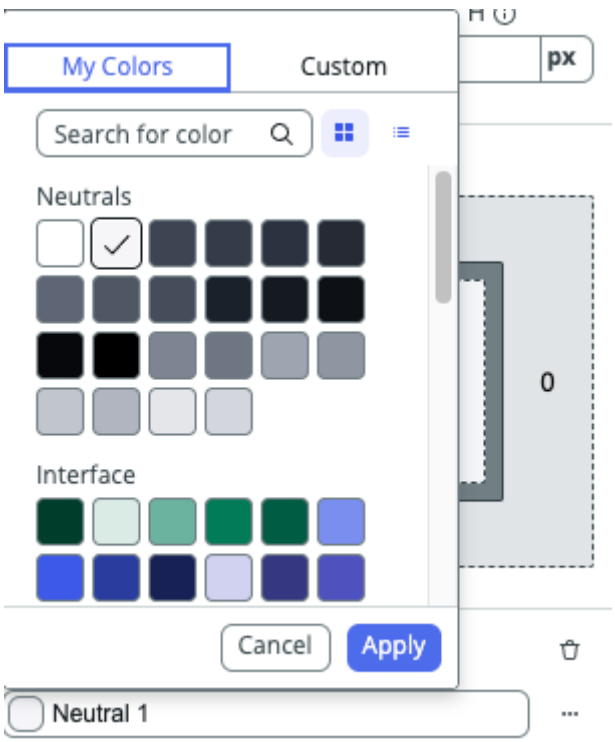
Specifies accessibility settings such as ARIA Region Name, ARIA Region Heading Level, ARIA Role, and inclusion of ARIA Heading. For detailed information, see the ARIA documentation on developer.servicenow.com.

Alignment (Column and Component)

For a column, defines how components align within the column as a group. For a component, defines how the component aligns within the parent layout element, such as a column or column layout.

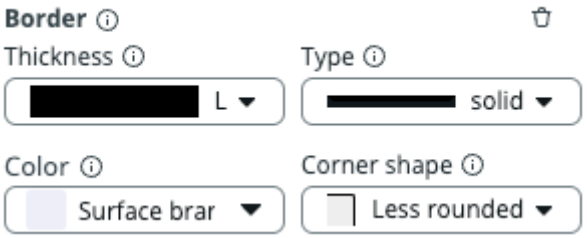
Background (Body, Column layout, Column, and Component)

Sets the background color behind the layout element. Select a color from the color picker **My Colors** tab where you can view available colors in a grid or a list. Alternatively, use the **Custom** tab to specify any color in HEX, RGB, or HSL.



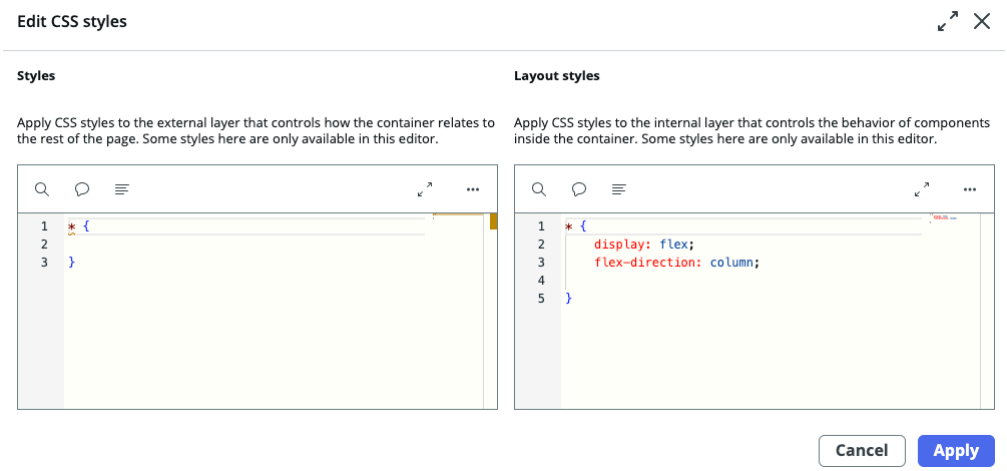
Border (Column layout, Column, and Component)

Adds a border around the content. The border is placed on the immediate inside of the margin and the immediate outside of the padding. Specify the border thickness, type, color, and corner shape. For components, you have the option to use custom CSS for the border.



Layout (Body, Column layout, Column, and Component)

For body and column, specifies direction, alignment, and content justification. For column layout, specifies the number of columns and the column gap (space between columns). For some components, such as Card Base Container, select **Edit CSS** to apply specific CSS styles to the container layer and to the internal layer controlling components within the container.



Shadow (Column layout, Column, and Component)

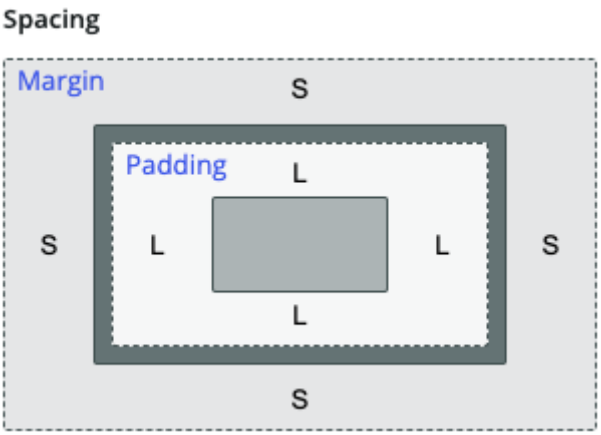
Adds shadow effects around the content. You have the option to use custom CSS.

Sizing (Component)

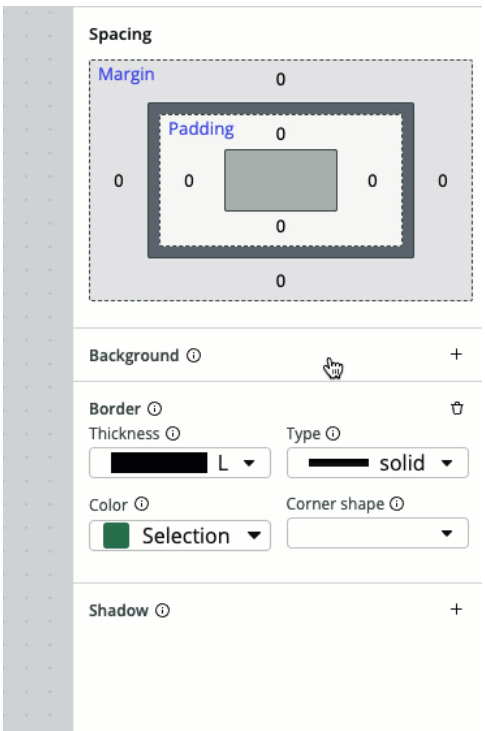
For applicable components, sets the default, minimum, and maximum height and width for the component (in px, %, em, rem, or a custom value). This value is relevant for when the browser window is resized.

Spacing (Body, Column layout, and Component)

Margin sets the size of the space on the immediate inside of the body, column layout, or component. Padding sets the size of the space on the immediate outside of the body, column layout, or component. Select Margin or Padding to set the same size for all four sides.

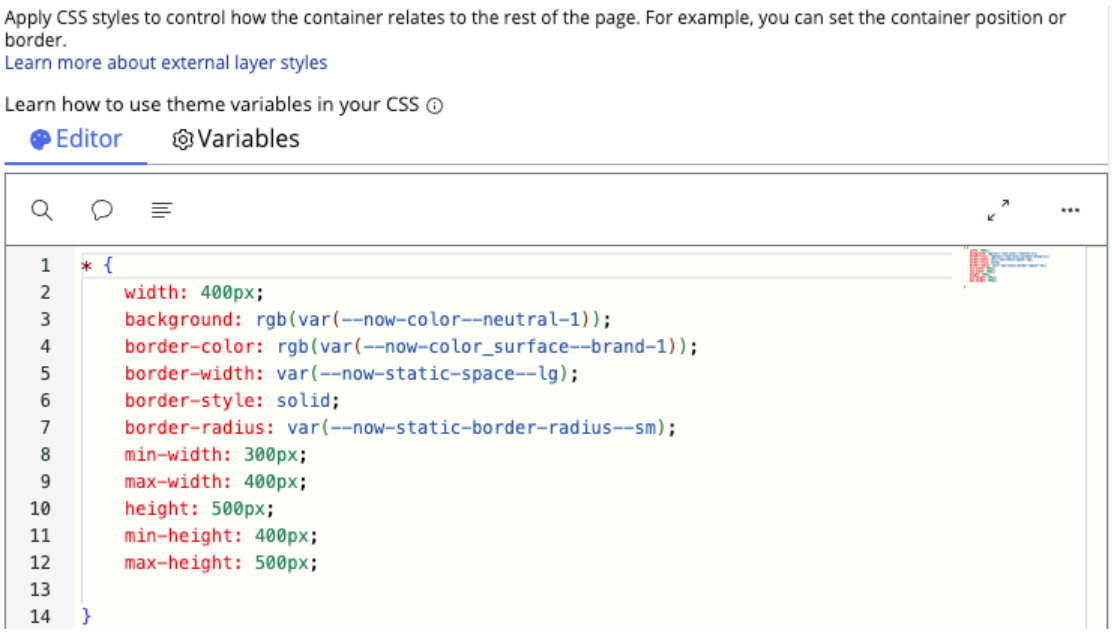


You can set the size for each margin or padding side by selecting the current setting and then selecting an option.



7. **Optional:** For Body and components, you can edit the CSS code by selecting the **View and edit CSS** link at the bottom of the **Styles** tab.
   The **Styles tab** displays a **CSS styles** code editor. The following CSS properties are the most commonly used:
   - `background-color`
   - `background-image`
   - `border-style`
   - `border-width`
   - `border-color`

- `border-radius`
- `box-shadow`
- `height`
- `min-height`
- `max-height`
- `margin`
- `overflow`
- `padding`
- `width`
- `min-width`
- `max-width`
- `z-index`



Important:
Some components contain built-in styling configurations that you can't override with CSS in UI Builder. For information about overriding these style configurations, see Manage the visual style of UI Builder experiences.

8. In the main header, select **Save** to save your changes.

Add styling to a component using the old layout system

Set CSS styles for a component to change its default appearance.

Before you begin

Role required: ui_builder_admin

About this task

This task describes how to add styles to the wrapper containing your component, which is generally recommended instead of applying styles to the component directly. Make sure that the component whose styles that you want to define is placed within a container component to put

the component in a wrapper. Your component's wrapper is one level higher than the component in the Content hierarchy and is labeled by default as `Main`.

To add styling to an entire page, you can use standards-based CSS in the wrapper for the page. For information about adding styling to your entire experience, see Manage the visual style of UI Builder experiences.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.

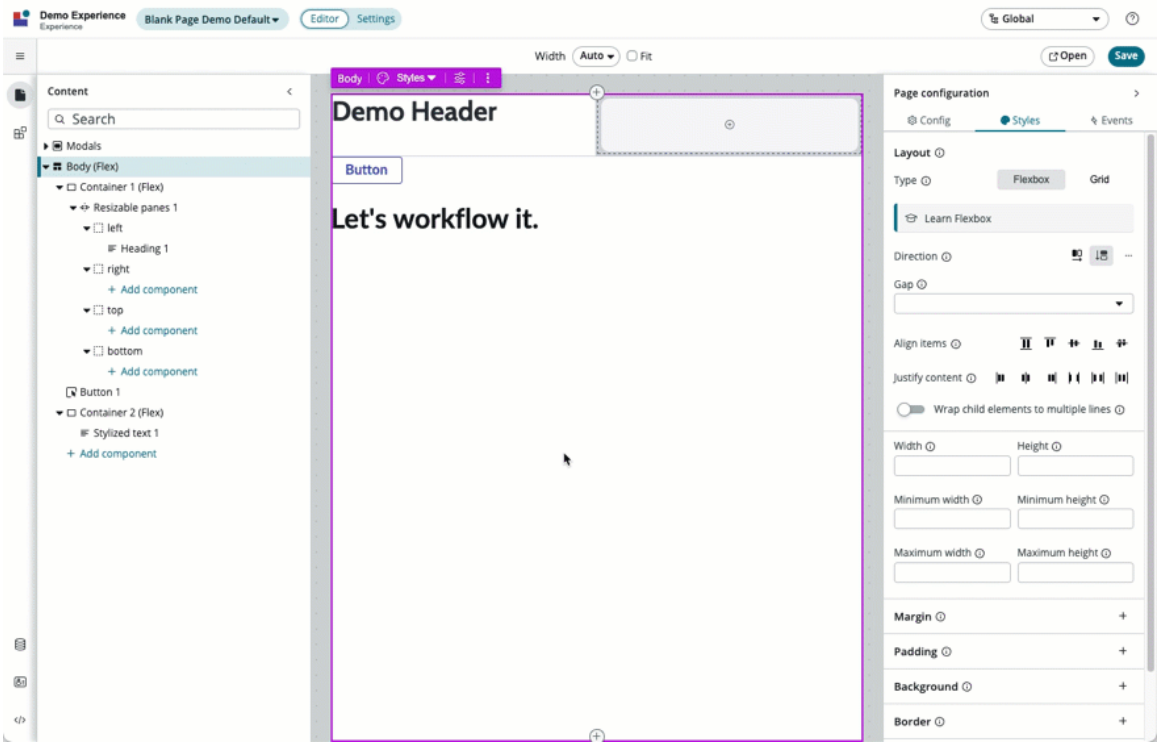
   Note:
   A component must be placed in a container before adding styles to the component.

4. Select the container holding the component you want to add styling to.
   The container is typically one level higher than the component in the content tree.
   The Styles tab appears in the configuration panel.
5. In the CSS Styles window, enter standards-based CSS properties and values.
   The following CSS properties are the most commonly used to apply styles for components within containers:
   - `background-color`
   - `background-image`
   - `border-style`
   - `border-width`
   - `border-color`
   - `border-radius`
   - `box-shadow`
   - `height`
   - `min-height`
   - `max-height`
   - `margin`
   - `overflow`
   - `padding`
   - `width`
   - `min-width`
   - `max-width`
   - `z-index`

Important:
Some components contain built-in styling configurations that you cannot override with CSS in UI Builder. For information about overriding these style configurations, see Manage the visual style of UI Builder experiences.

6. In the main header, select **Save** to save your changes.

Add styling to a component in the Styles configuration panel



Duplicate a component

Duplicate a configured component to reuse on a page.

Before you begin

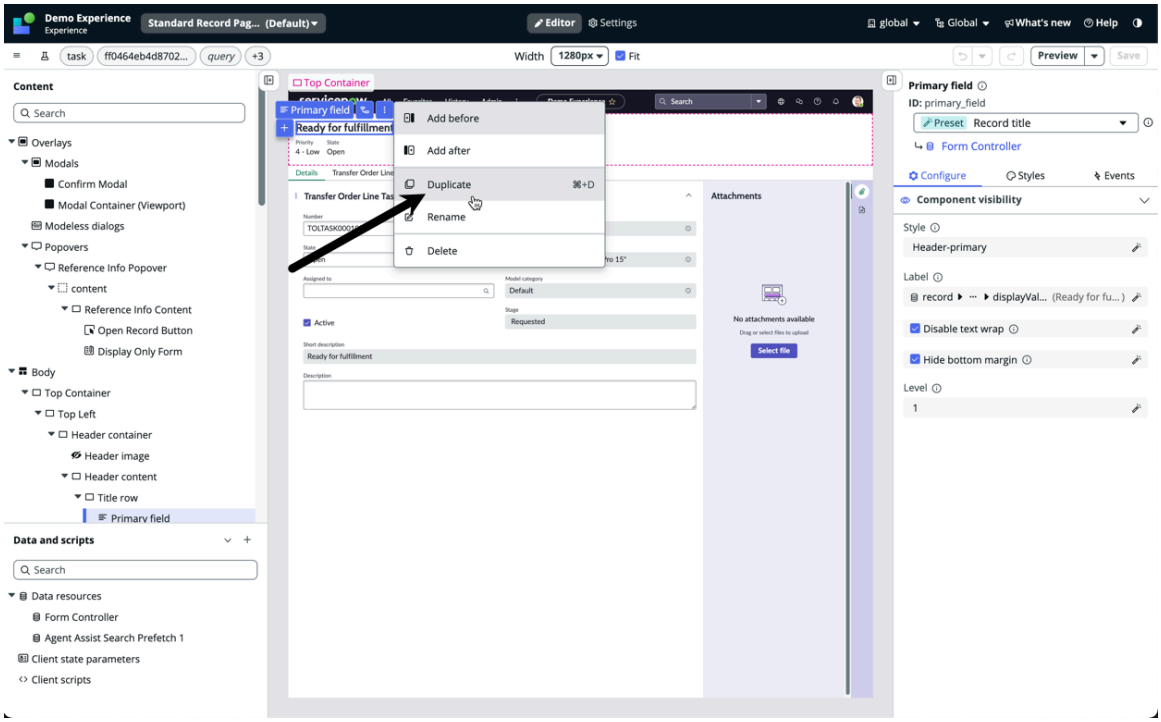
Role required: ui_builder_admin or admin

About this task

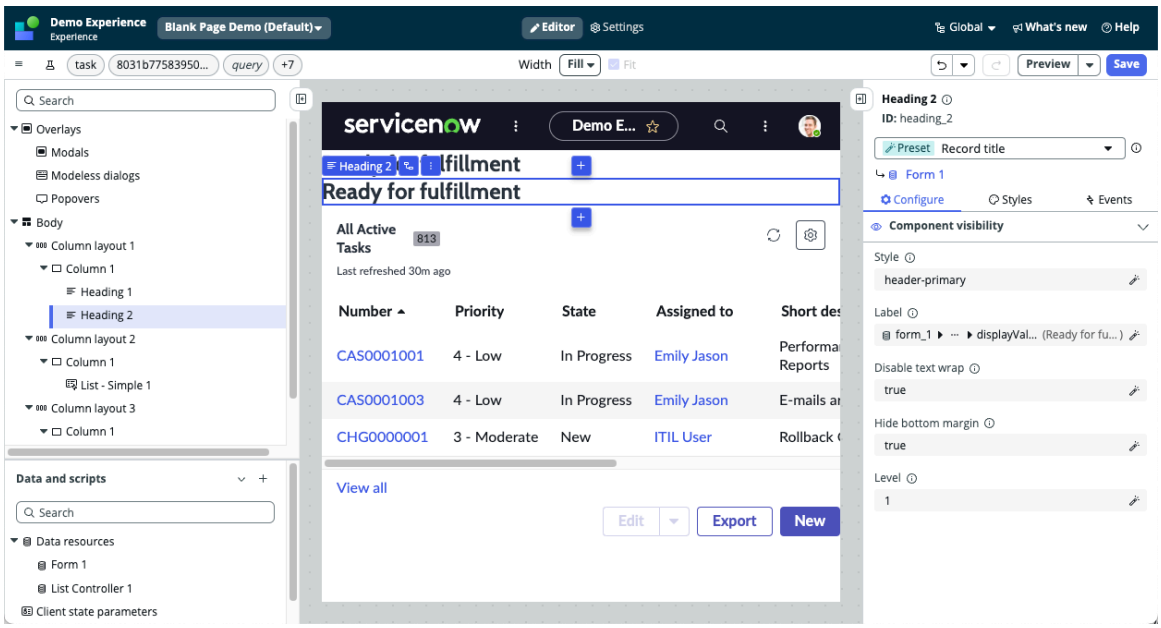
Create an exact copy of a configured component on your page except for the name and ID. A duplicated component copies all properties, bindings, and events. The duplicated component appears directly after the component that is copied. If a tabs component is duplicated, all tabs and their children are copied as well as their names and IDs are made unique.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Open the page variant with the component you want to duplicate.
4. Select the component that you want to duplicate.
5. Right-click the component.
6. Select **Duplicate**.
   You can also use the keyboard shortcut **cmd + D** (Mac) or **ctrl + D** (Windows).



The duplicated component appears below the copied component in the content tree.



7. Select **Save**.

Page collections

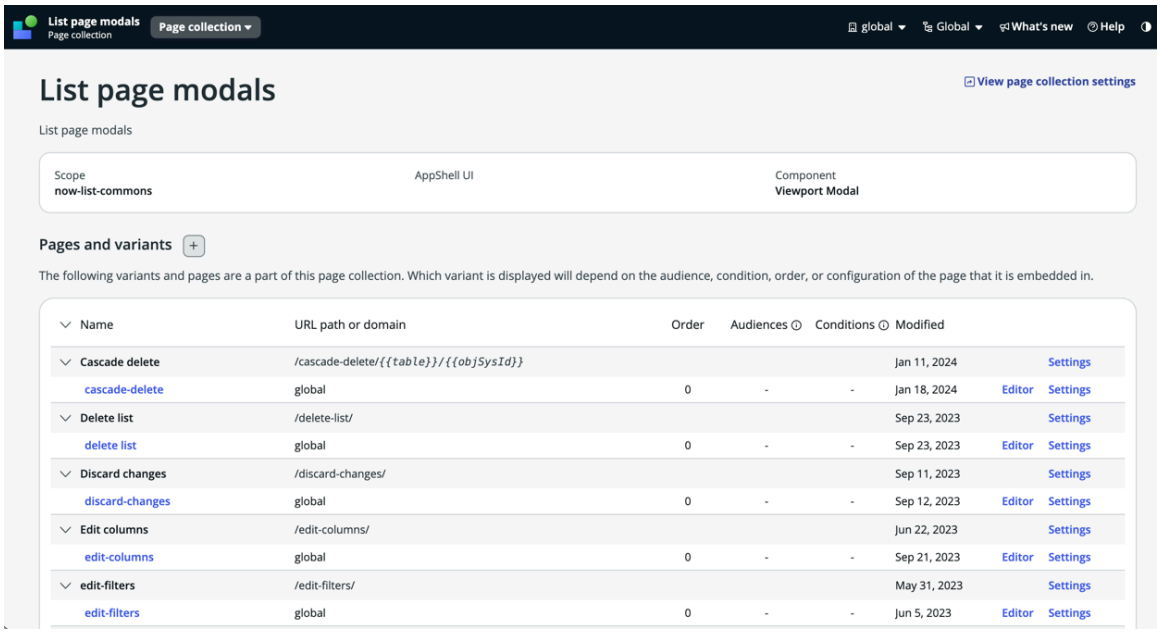
Page Collections are groups of pages that can be used across multiple experiences.

What is a page collection

Page Collections (sys_ux_extension_point) are groups of pages that are meant to be re-used across multiple experiences. These sets of pages can be used in a modal viewport or in a tabs component. These pages are sandboxed and do not have access to things like URL params or data resources from the page that is calling them. All they have access to is the Controller record

that is specified on the Page Collection record and that is passed down by the calling page in runtime.

Page collections are sandboxed inside of the pages they are within, this means that they do not have direct access to change the parent page. However, because of the way seismic works, your page collection page can dispatch an event that the calling page can listen for to take action. This is the only way to achieve a change in the calling page.



Why should you use page collections

Page collections allow you to reuse custom pages across experiences. This will save you time if you need to replicate a page for multiple experiences and help pass information to different groups of users.

You can also use page collections to provide additional information within an existing topic by using a viewport modal. A viewport modal will allow your audience to view pages outside of their experience to make changes without leaving the parent page.

Create collection of tabs that can be shared across experiences. Each tab is an individual page in a page collection.

Where can you use page collections

You can access a page collection from the UI Builder home screen. The list of page collections details which component and type each page collection is.

You can create a page collection from the UI Builder home screen, or from a component that uses page collections.

Page collections can be used with the following components:

- Viewports
- Viewport Modals
- Tabs

Things to consider

The page collection creation screen requires a component and controller selection.

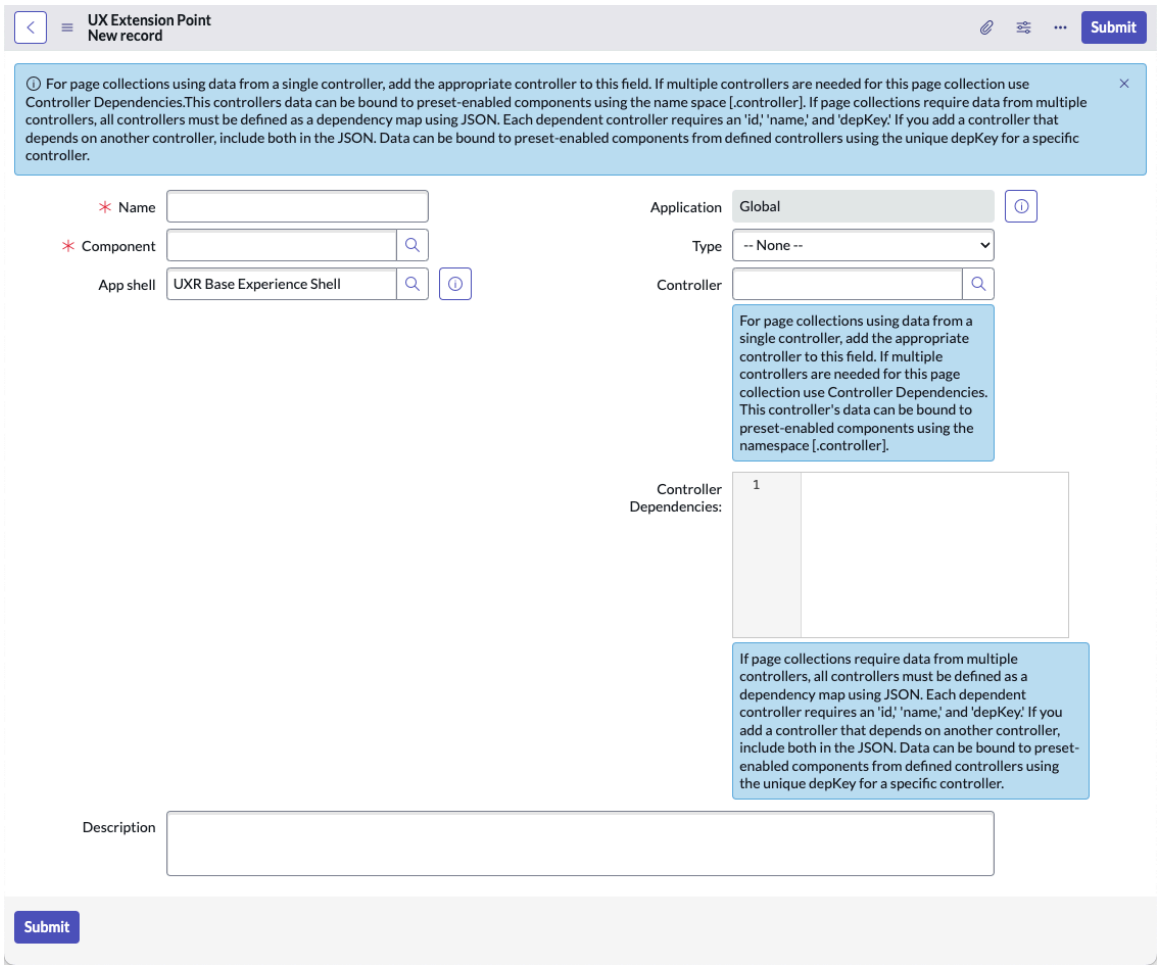
Data resources need to be selected to pass information to a page collection. Need to bind data from the parent page.

Need to route to the page in a page collection you want to display in a viewport.

Add a link to creating a page collection.

Page collection on the platform

You create a page collection within the ServiceNow platform. At this time, only the Tab and Viewport component can be used when creating a page collection. You can select a controller or pass controller dependencies using JSON.



Create a page collection across multiple UI pages

Create a page collection to accommodate tabbed content that can be used across experiences.

Before you begin

Role required: admin

About this task

Use page collections to create tabbed content in your experiences with the Tabs component in UI Builder.

Pages within a page collection don’t have access to the parent page's URL parameters or data resources. You can set conditions for pages in a page collection to define which page to display to certain audiences. Page templates are not supported inside a page collection.

Procedure

1. Click **Create** > **Page collection**.
2. On the form, fill in the fields.

Create a page collection form

Field	Description
Name	Name to track your page collection internally.
Controller	Controller that defines the data for the page collection.
App shell UI	Type of app shell UI that you want to use with the page collection. The app shell is the wrapper of the page contents, which is similar to the functionality of a web page. The app shell can show things like the logo of your company, user preferences, and the search icon. For more information, see Define UI experiences using app shells.
Description	Short description to help find your page collection. Write a description that helps page builders understand what content is included in the page collection.
Application scope	Application scope defines which application that the page collection is part of. To create a page collection in a different application scope, change the application scope in the platform and then create a page collection in UI Builder.

3. Click **Create**.
4. Click **Edit page collection**.
   A new tab opens to the page collection editing screen.
5. Click **Start editing**.
6. Click **Create a page**
7. Enter a unique name for the page in the **Name** field.
8. Specify a path for your page in the **Path** field. UI Builder generates a default path based on the name that you provided in the previous step.

   Note:
   The application scope defaults to the scope that the user is in. For more information about the application scope, see Learn about security and roles.

9. Click **Create**.
10. **Optional:** Set advanced settings such as required and optional parameters, as well as variant settings.

a. Select **Add required parameters** to add any required parameters to your page URL.

A required parameter is a piece of data that your page requires, such as a sys_id, table, or query. Required parameters are useful for components, because they can bind to the value of the required parameter.

For more information, see Manage UI Builder pages and page variants.

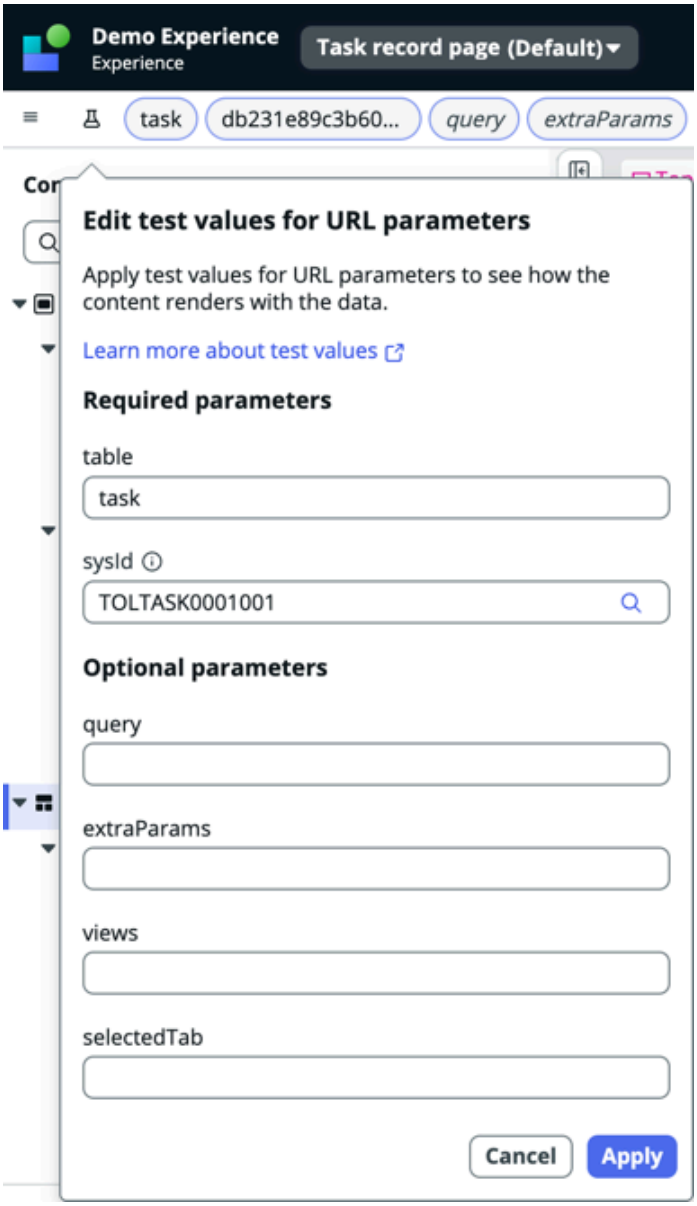
b. Select **Add optional parameters** to add any optional pieces of data that you want to add to the URL of your page.

Unlike required parameters, optional parameters are always name and value pairs that work regardless of the order they're provided in.

For more information, see Manage UI Builder pages and page variants.

c. Click the required or optional parameter in the URL, and in the **Test values** field type a value, such as `incident`.

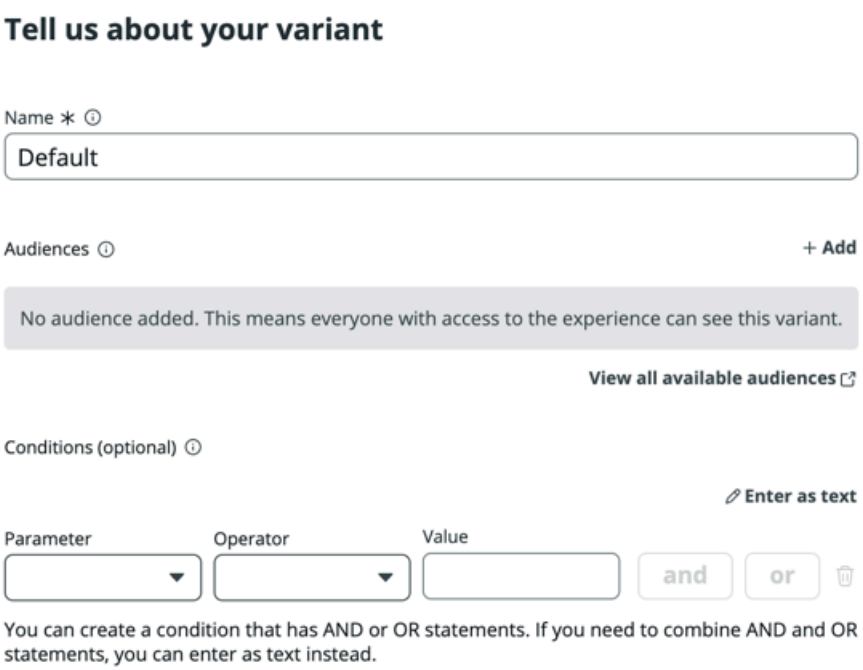
You add a test value to your page to populate data into the page as a way to test it. For example, if you add a table as a required parameter, you could add a test value of incident to bring in test data on the incident for that table.



For more information on test values, see Test values in a page.

d. Select the **Variant** tab to set the audience and conditions settings for your page.

(Optional) When you create a page, UI Builder also creates a variant of the page for you by default. A page variant is a variation of your page at the same path that lets you target experiences for different audiences using user criteria. For example, a page for managers, and a variant of that page for the manager's direct reports. For more information about creating a variant, see Create a page variant.



For more information about audiences, see Learn about audiences.

e. Click **Done**.

11. Click **Done**.

12. Add and configure components on your page.

13. Click **Save**.

14. Optional: Add more pages to your page collection.

a. Click **Menu**.

b. Select **Create page**.

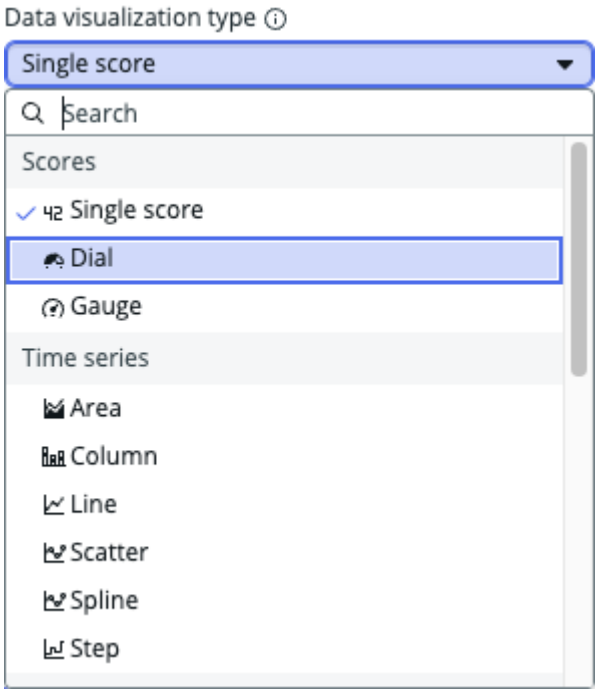
c. Repeat steps 6 through 13.

15. Close the page collection window.

Change data visualizations in UI Builder

Change data visualizations in real time using a drop-down list to preview data in your experience.

You can change data visualization types via a drop-down in the configuration panel to preview different data visualization types. Each data visualization can be configured to display different data depending on who is using your experience. Select the drop-down under **Data visualization type** to preview different types of data visualization.



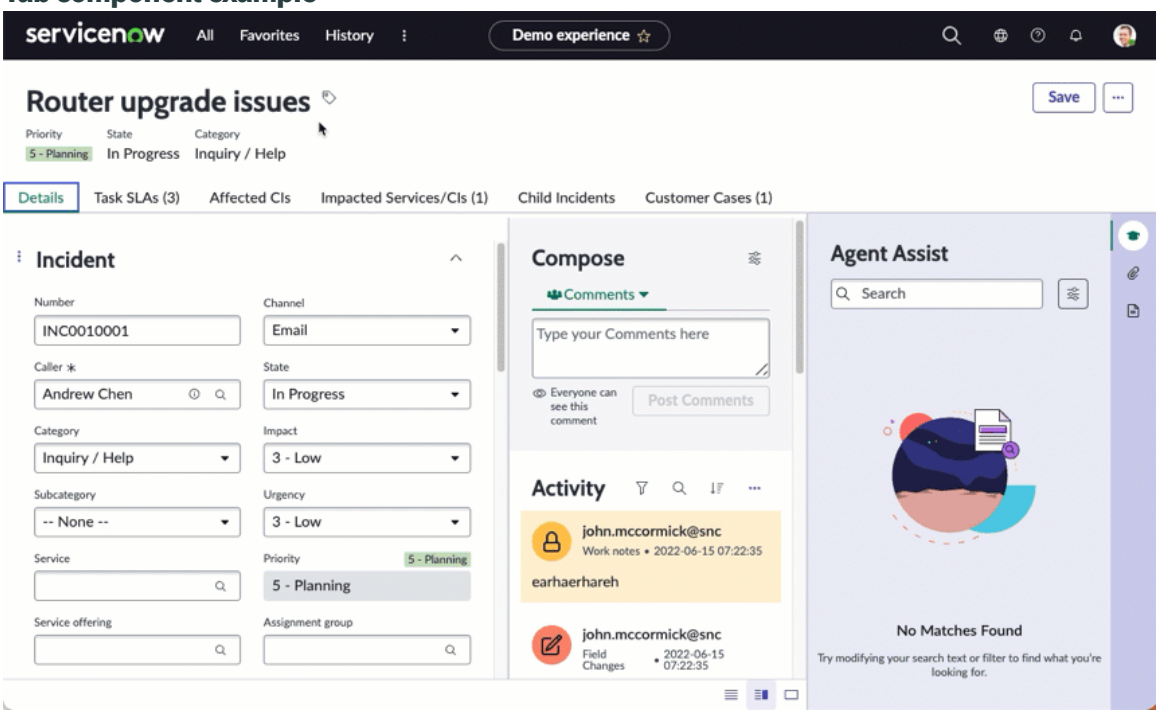
A preview of the data visualization you selected appears in your page. For information about adding data visualization components to your experience, see Add and configure components.

Add tabbed content to UI Builder pages

Use the Tabs component to add tabbed content to pages in UI Builder.

Create additional navigation on your UI Builder page by adding a Tabs component to your page. Use the Tabs component to nest components within tabs on your page. Tabs can be configured as either horizontal tabs at the top of the content, or as vertical tabs on the left or right side of the content.

Tab component example



Tabs can be organized in the configuration panel to reorder how they are displayed on the UI Builder page. Add custom labels and icons to tabs to provide unique visual identifiers for navigating across tabs. These changes can be viewed in real-time as you edit the Tabs component.

You can add one of the following types of tabs to your UI Builder page:

- Empty container tab
- Repeater tab
- Related list tab
- Page collection tabs

The tabs component supports any combination of static tabs, repeated tabs, related list tabs, and page collection tabs. Alternatively, you can still use the Viewport tab mode to add a series of viewports.

Add empty container tabs

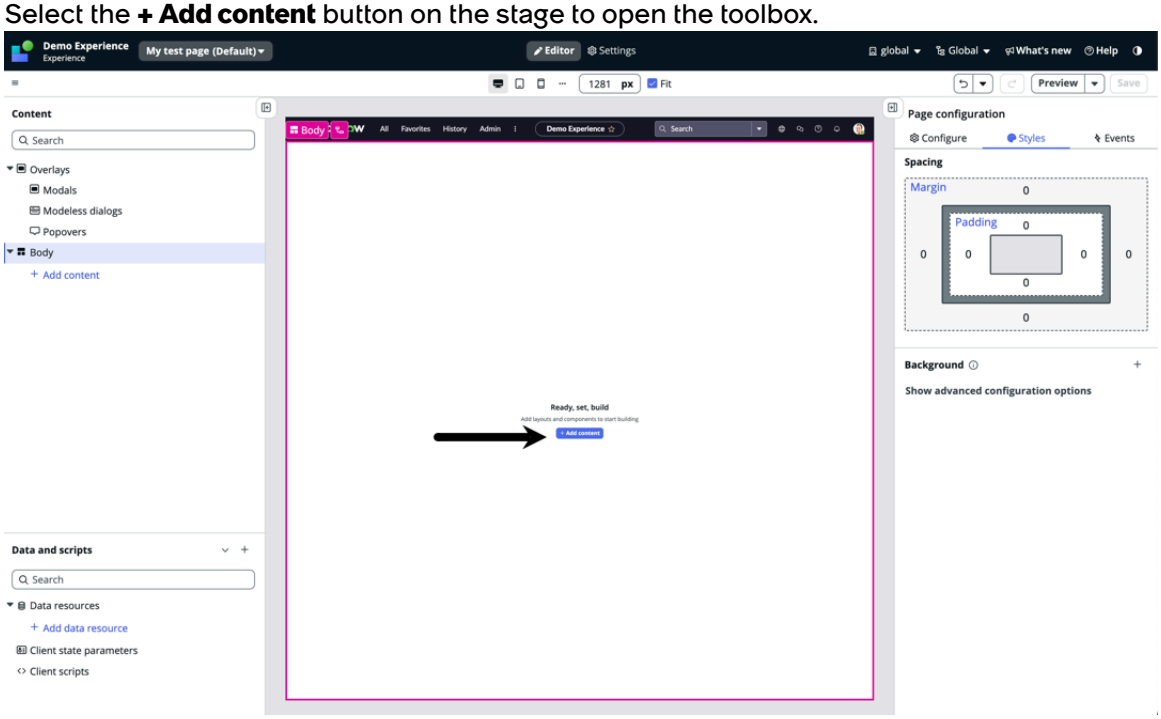
Use an empty container tab to manually create a static set of tabs on your page.

Before you begin

Role required: admin

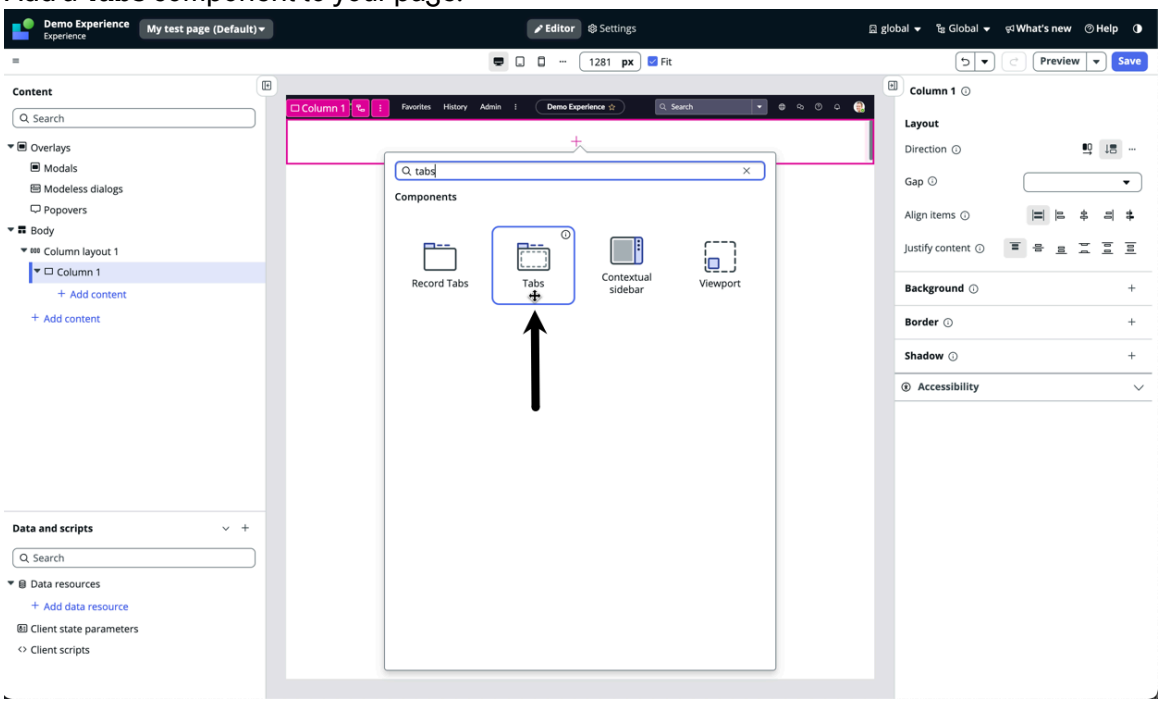
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Select the **+ Add content** button on the stage to open the toolbox.



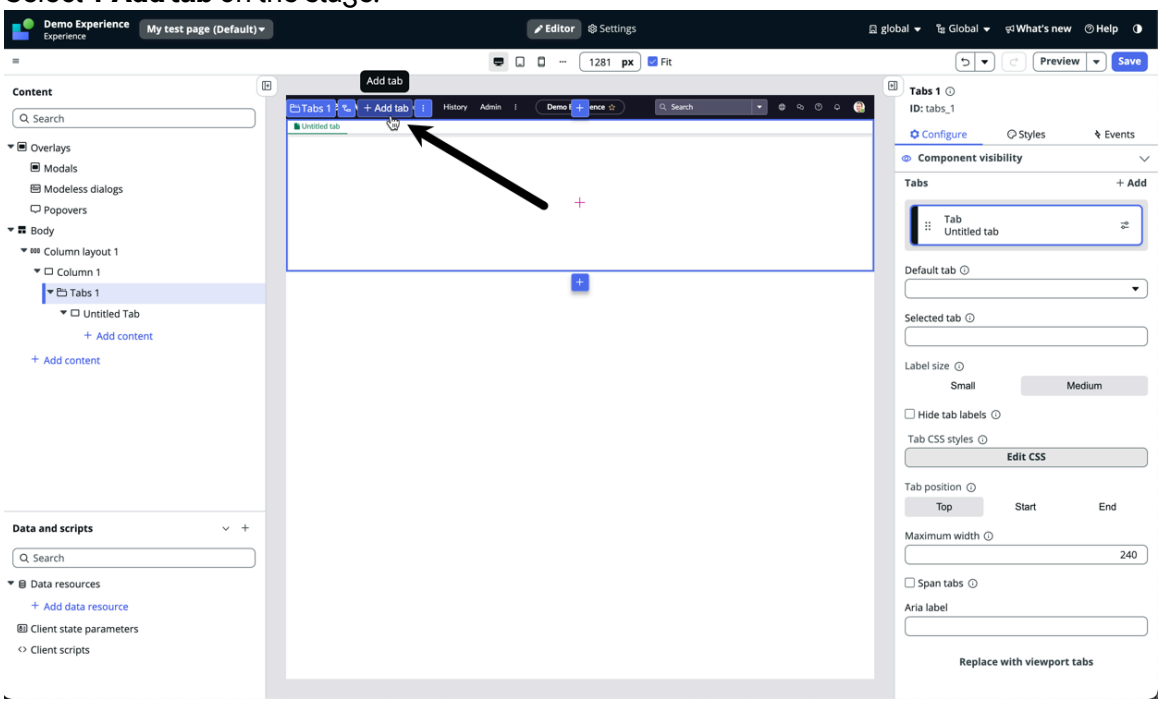
5. Select a **Single column** layout.
6. Next, select the **+** icon in the column to open the toolbox.

7. Add a **Tabs** component to your page.

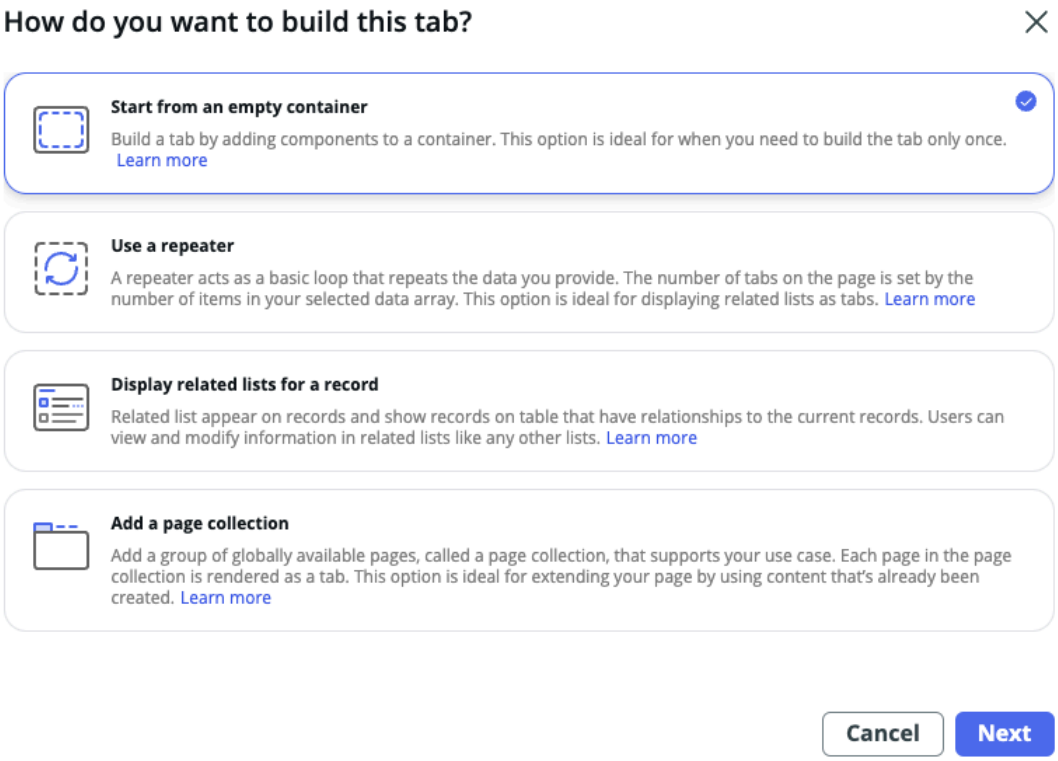


For more information on how to add a component to a page, see Add and configure components.

8. Select **+ Add tab** on the stage.



9. Select **Start from an empty container**.

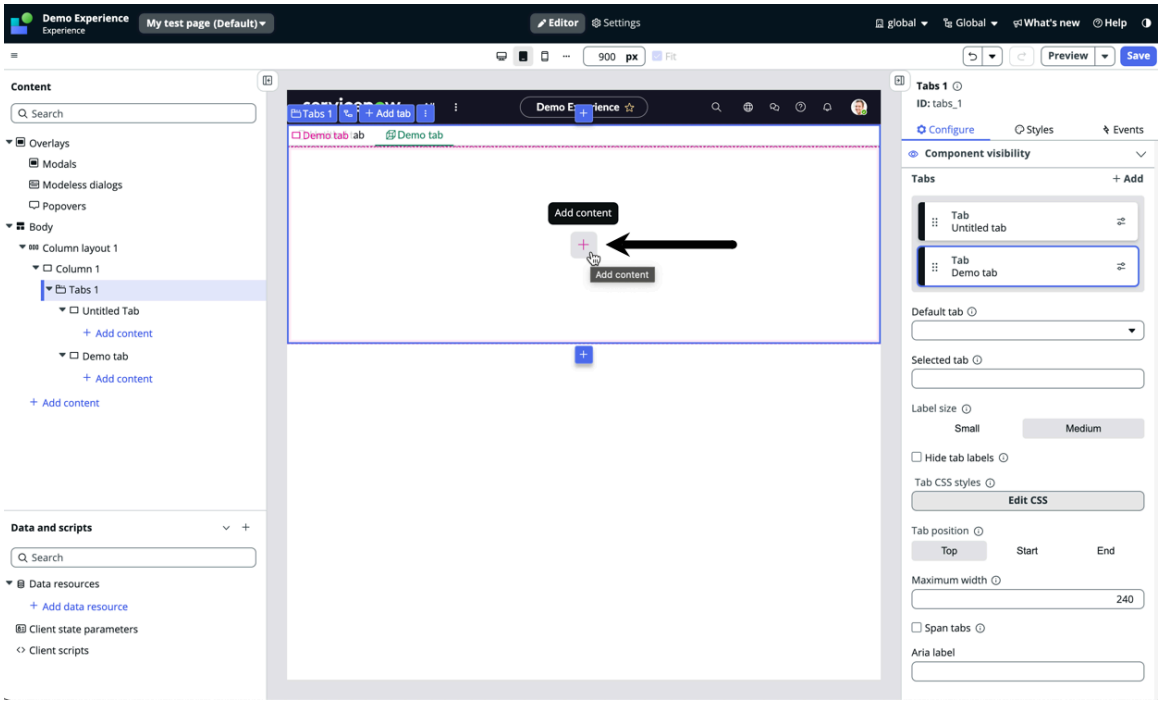


10. Select **Next**.
11. On the form, fill in the fields.

Tab settings form

Field	Description
Tab label	Label for the tab to display on your page.
Icon	Icon to appear next to the tab label. Icons are not required.
Hide tab	Whether to hide or display the tab.

12. Select **Create**.
    The new tab displays in the **Tabs** component.
13. Select the new tab.
14. Select **+** icon.



15. Select the component that you want to add to the tab.
    Components display under the tab in which they're nested in the content tree. For more information, see Add and configure components.

Result

Your page shows the two tabs that you created. Select each tab to further configure them, add styling, or add an event handler. For more information on styling, see Change the default appearance of components. For more information on adding an event handler, see Manage actions in UI Builder pages.

Add repeater tabs

Use the Tabs component to create a set of repeater tabs by linking tabs to a data array.

Before you begin

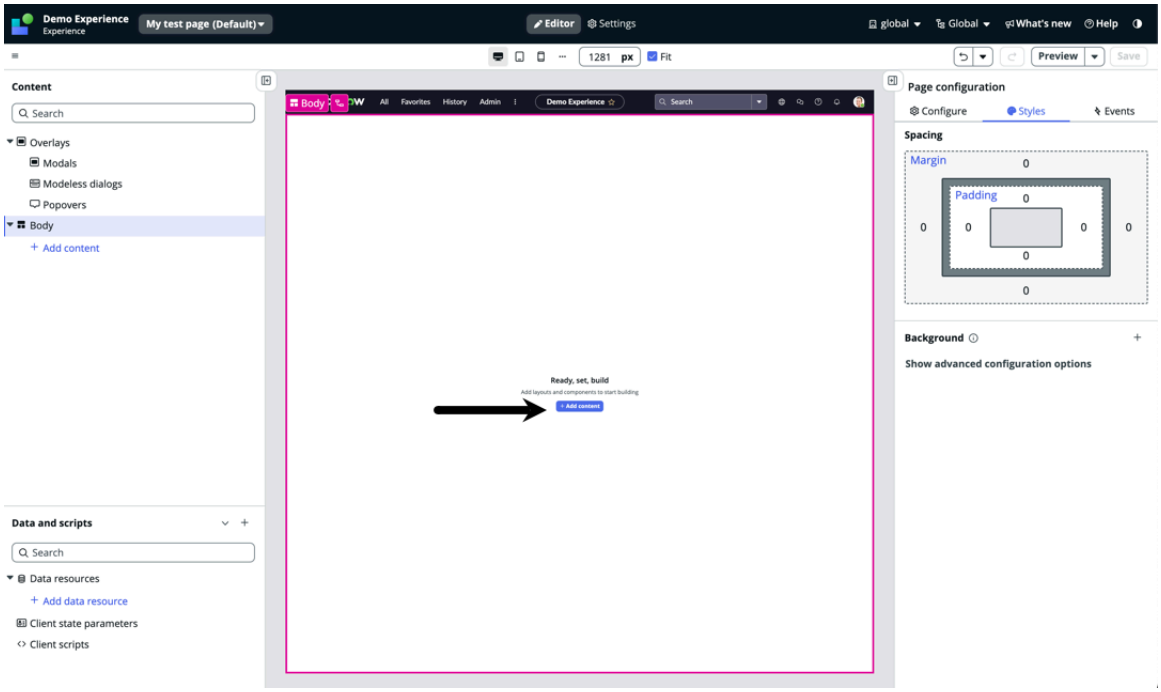
Role required: admin

About this task

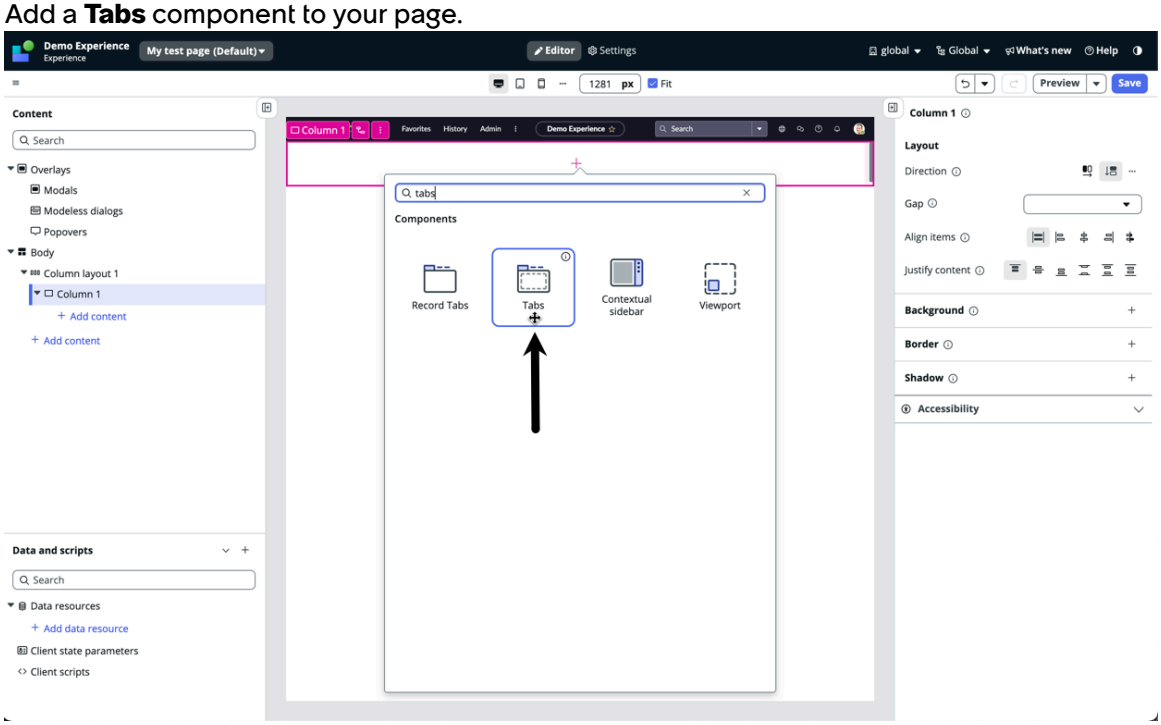
You can use repeater tabs to create multiple tabs based on the data array you provide. You can pass the icon, label, count, and fields as a key in the object. Use the label key to add names to tabs. Use the field key to pass information to repeater tabs. You can bind a data broker, client state, or a client script to the data array provider to return the array of objects with the correct schema.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Select the **+ Add content** button on the stage to open the toolbox.

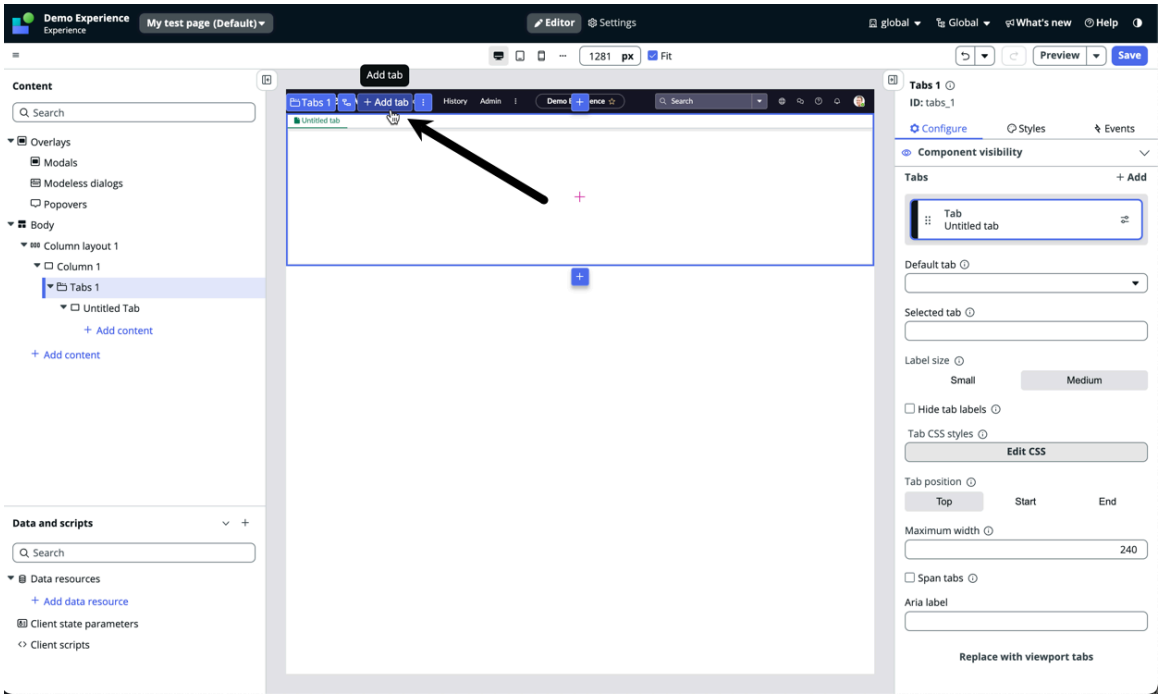


5. Select a **Single column** layout.
6. Next, select the **+** icon in the column to open the toolbox.
7. Add a **Tabs** component to your page.

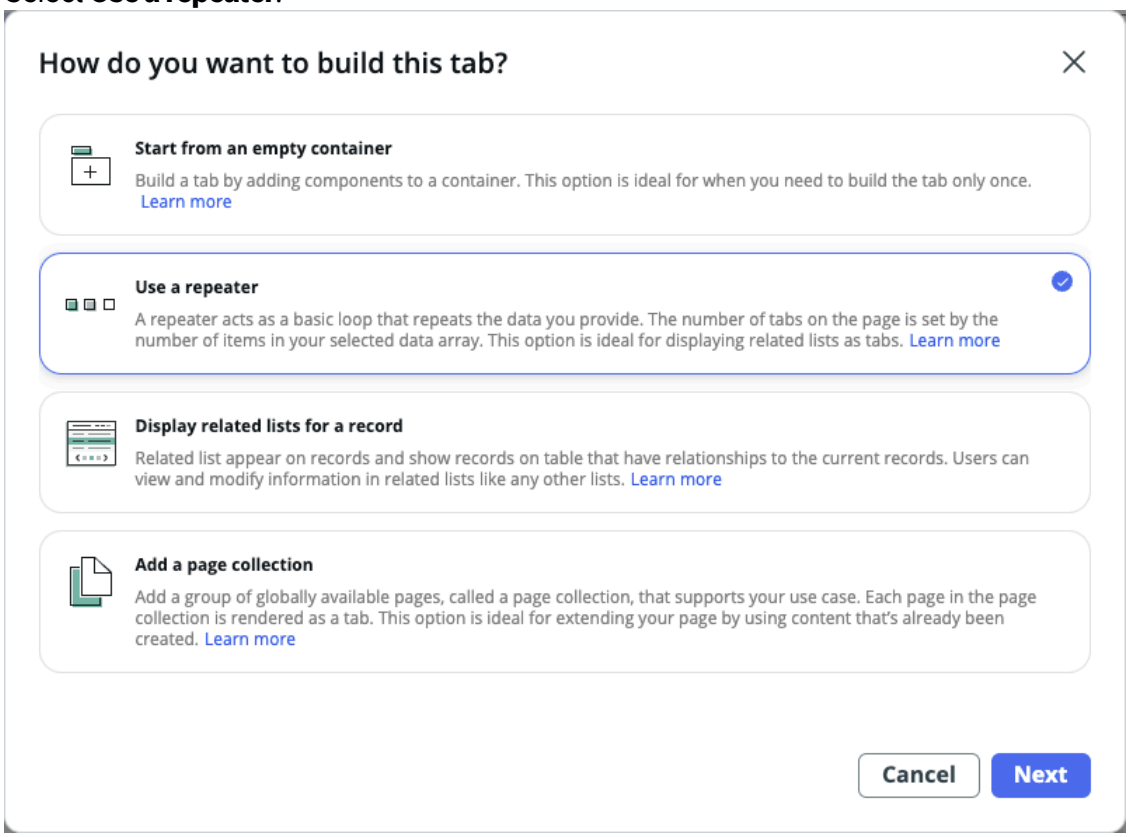


For more information on how to add a component to a page, see Add and configure components.

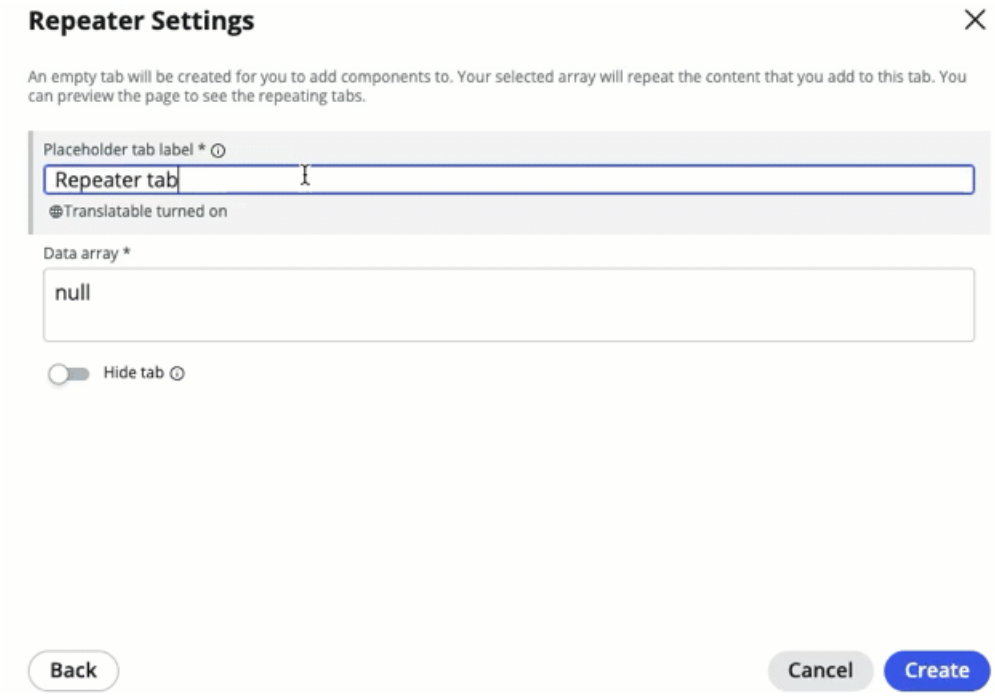
8. Select **+ Add tab** on the stage.



9. Select **Use a repeater**.



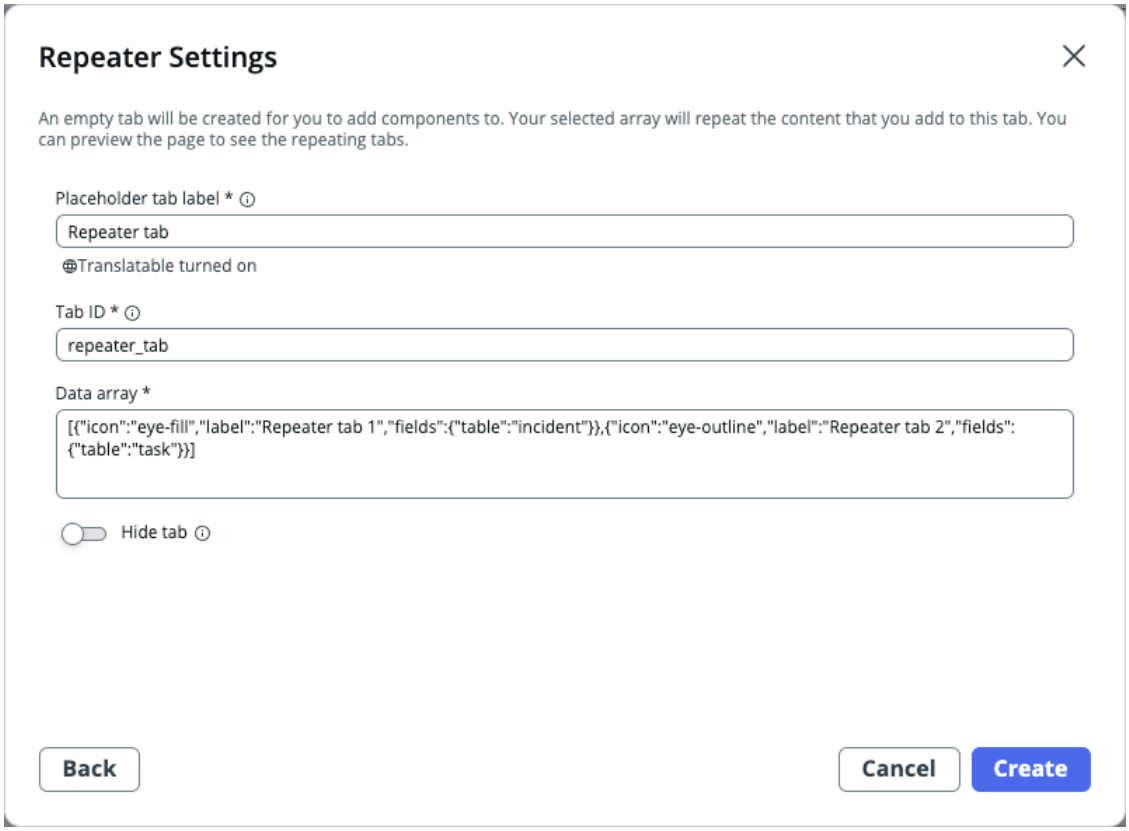
10. Select **Next**.
11. Enter a label for the repeater tab in the **Placeholder tab label** field.
    This label is assigned to the first repeater tab and is the only tab displayed in the page preview.
12. Select **Edit** in the **Data array** field.



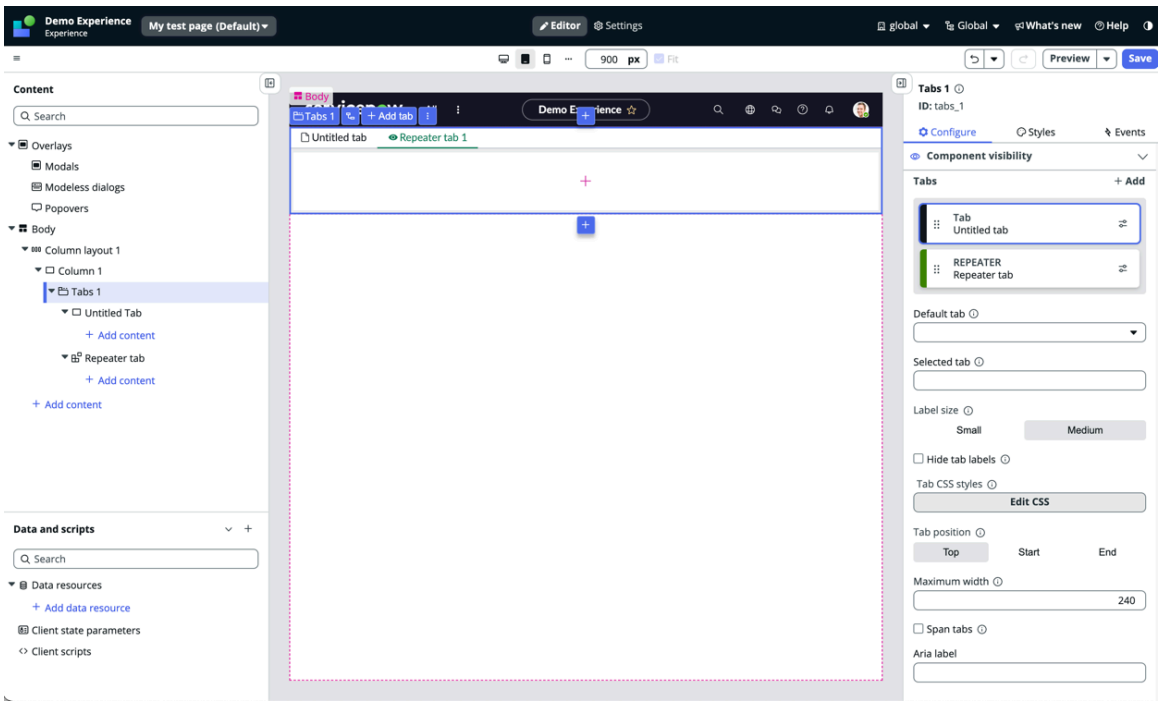
13. Configure the data array in the JSON editor.

Key	Data type	Description
icon	String	Name of the icon that appears in UI Builder.
label	String	Required. Display name of the tab in UI Builder.
count	Number	Value to provide on the tab label. Dynamically bind to a data broker or a client script that will be displayed on the label.
fields	Object	Object that can be used to store information and to pass down to the components in the tabs.

14. Select **Apply**.
The data array field displays the data array configuration.



15. Select **Create**.
    The new repeater tab appears on the page and in the content tree.



16. Select **Save**.
17. Add components to the repeater tab.
18. Select Preview to verify that the repeater tabs display correctly on your page.

Add related list tabs

Create a set of tabs on a page by linking to a related list.

Before you begin

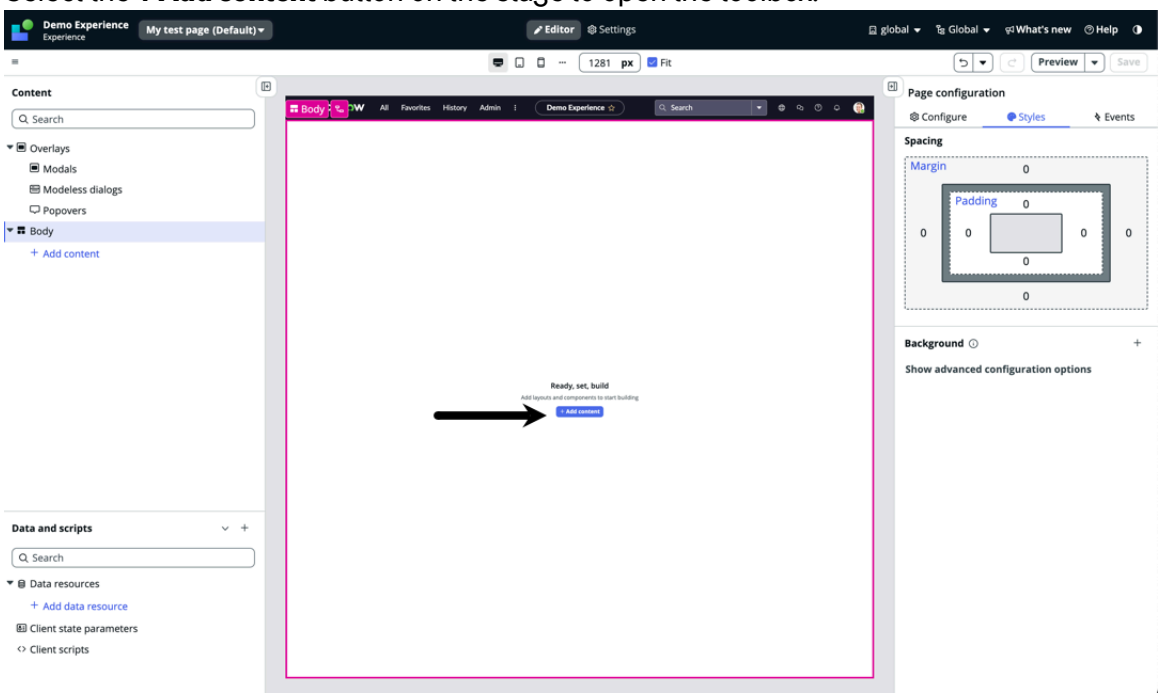
Role required: admin

About this task

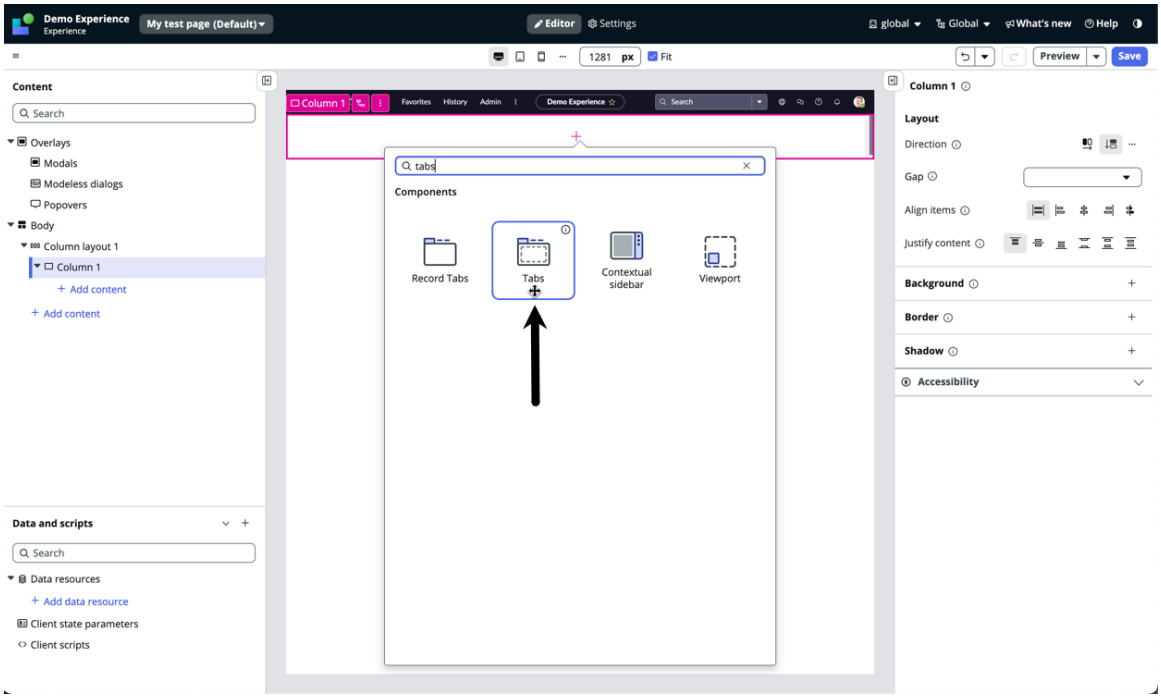
The related list tab automatically populates tabs based on the record that your page is displaying. For example, by default the related lists for the user table are Roles, Groups, Delegates, and Visibility domains. If your page is displaying a user record, then creating a related list tab adds all four of these related lists as tabs on your page. You can preview the page to see the related lists that were added. A record controller is required to create related list tabs. To add a controller to your page, see Bind data to UI Builder pages using controllers (advanced feature) for more information.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Select the **+ Add content** button on the stage to open the toolbox.

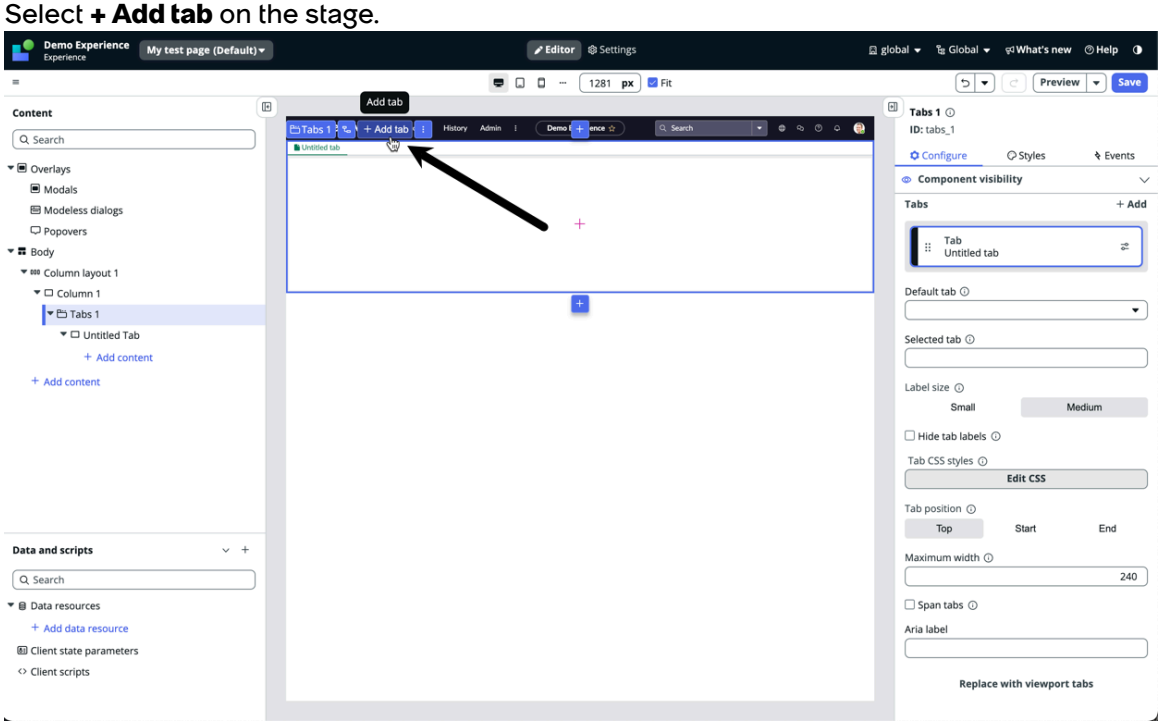


5. Select a **Single column** layout.
6. Next, select the **+** icon in the column to open the toolbox.
7. Add a **Tabs** component to your page.

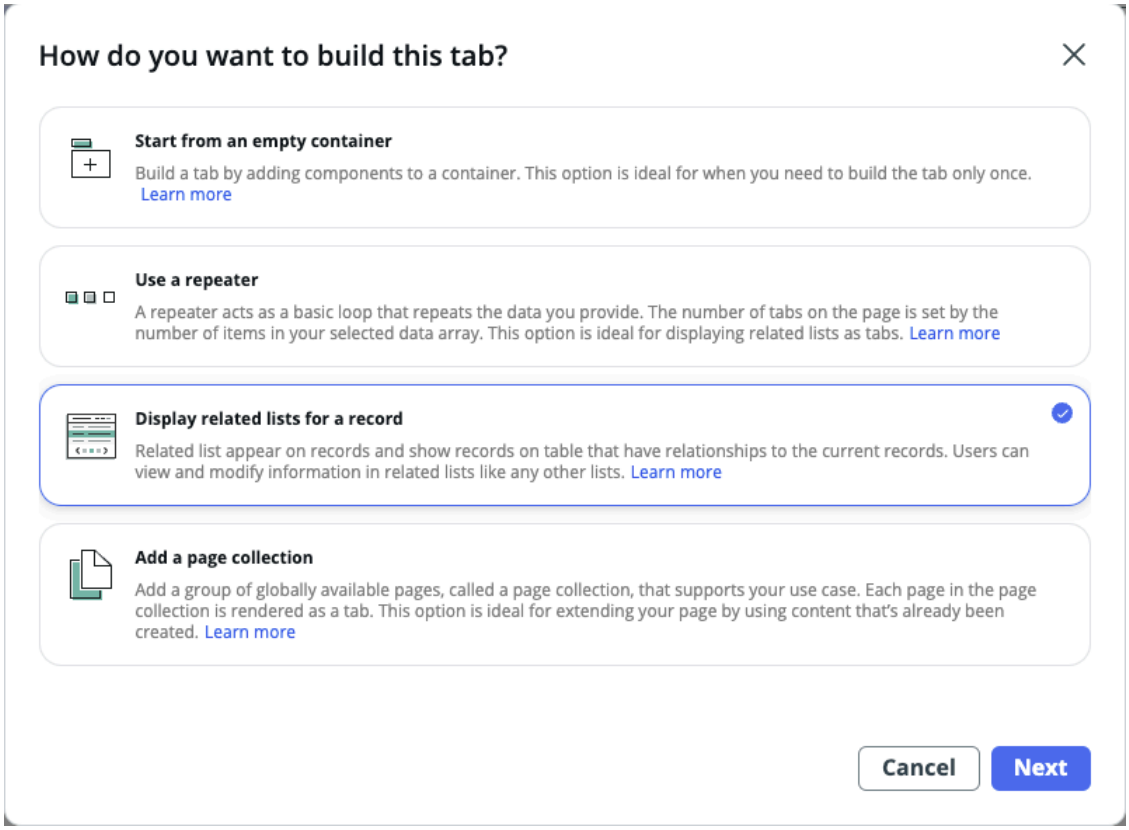


For more information on how to add a component to a page, see Add and configure components.

8. Select **+ Add tab** on the stage.



9. Select **Display related lists for a record**.

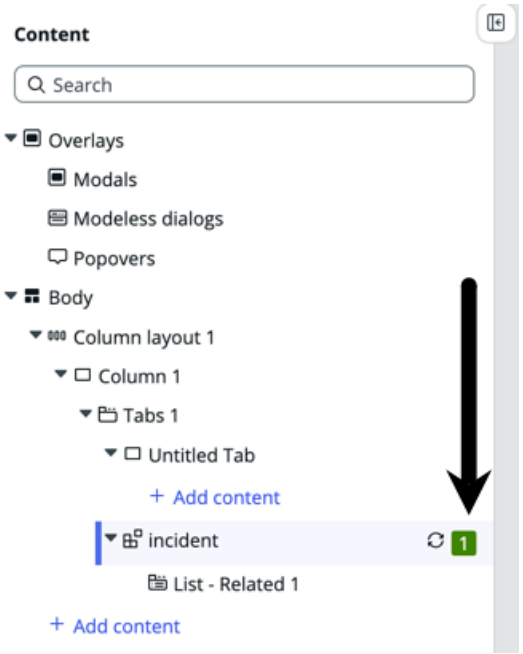


10. Select **Next**.
11. On the form, fill in the fields.

Tab settings form

Field	Description
Placeholder tab label	Placeholder label to appear on the preview tab of your page.
Hide tab	Whether you want to hide or display the tab.
Record Controller	Record controller with the related list.

12. Select **Create**.
    The new related list tab displays in the **Tabs** section of the configuration panel. Only one placeholder tab appears in the page preview. The green icon next to the related list tab in the content tree shows how many tabs appear on your page.



13. Select **Save**.
14. View and test your page by selecting Preview.

Add page collection tabs

Use page collection tabs to recreate the same tabs across multiple pages in UI Builder.

Before you begin

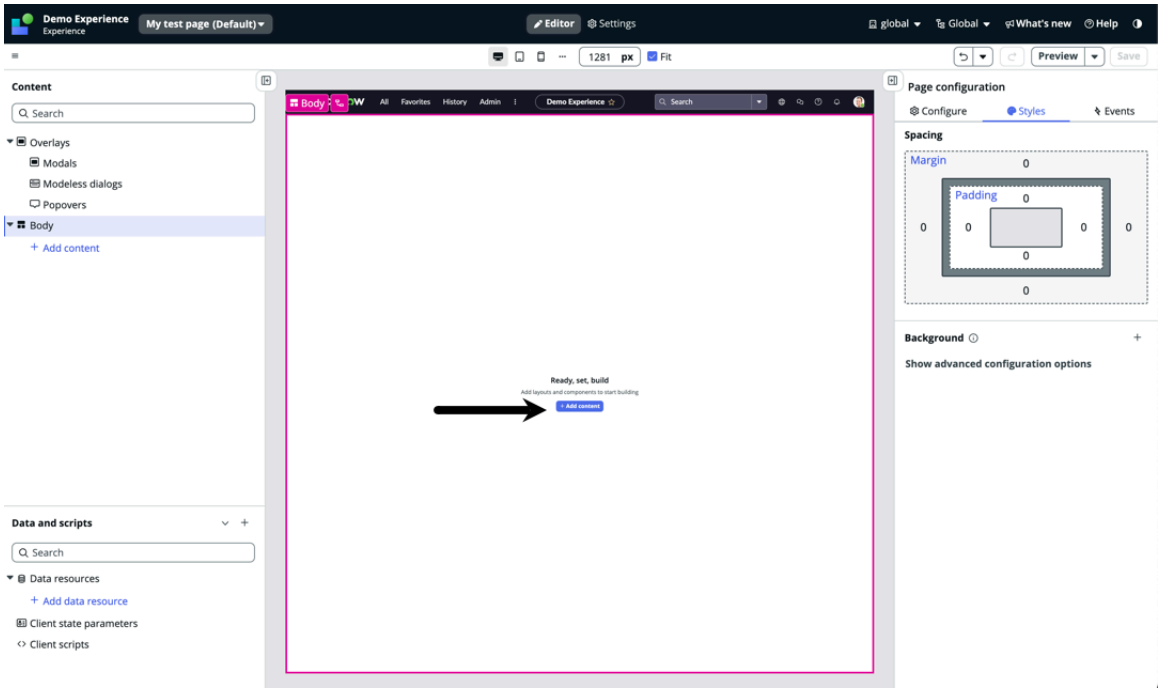
Role required: admin

About this task

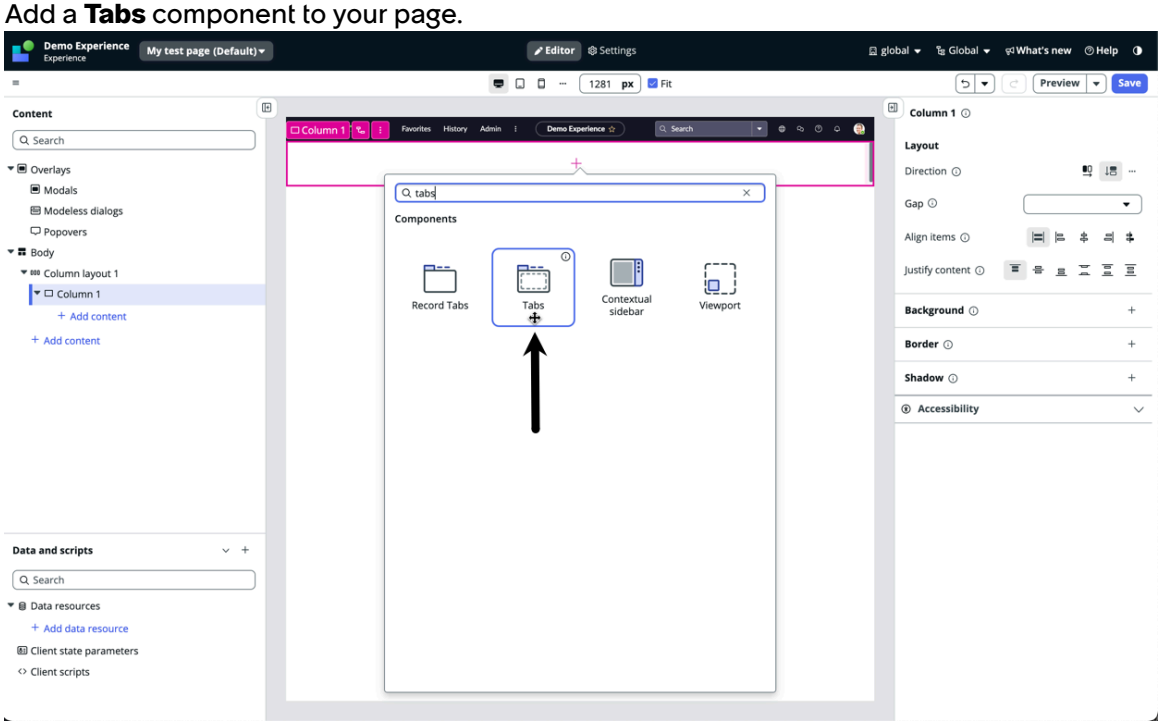
A page collection is a group of prebuilt globally available pages. Use page collection tabs to render each page in a page collection as a tab. You can select an existing page collection or can create your own. A controller is required to add a page collection. For more information about page collections, see Page collections.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Select the **+ Add content** button on the stage to open the toolbox.

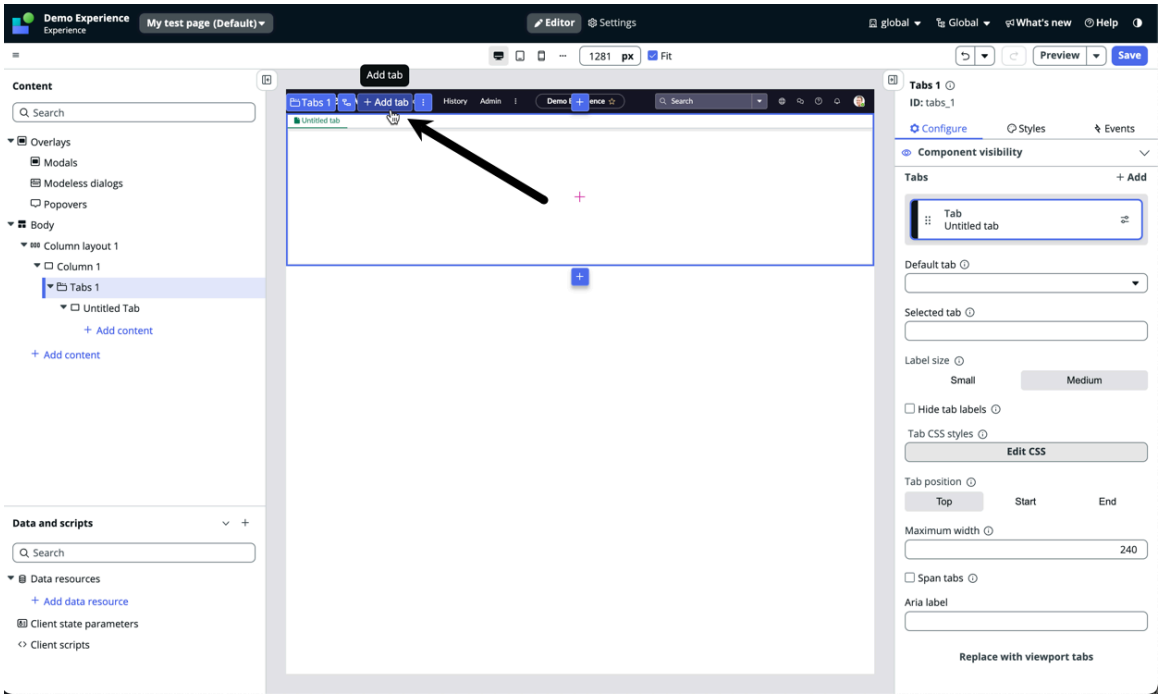


5. Select a **Single column** layout.
6. Next, select the **+** icon in the column to open the toolbox.
7. Add a **Tabs** component to your page.

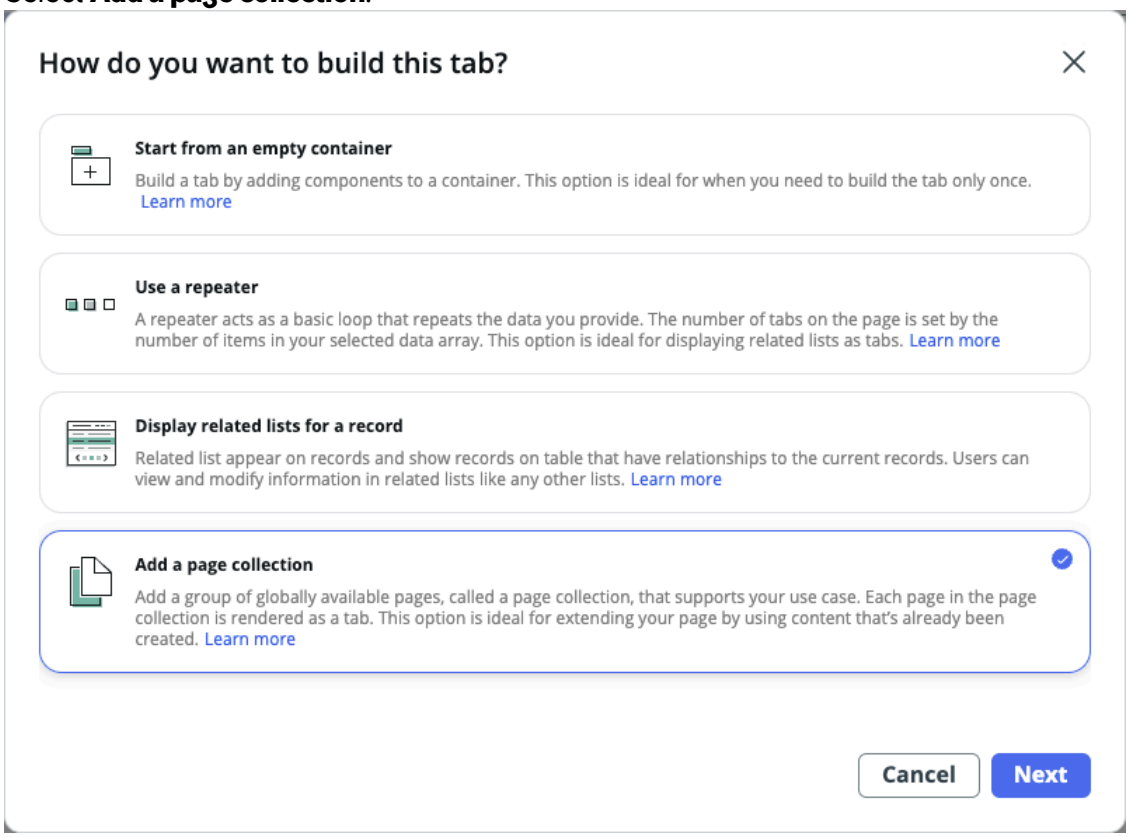


For more information on how to add a component to a page, see Add and configure components.

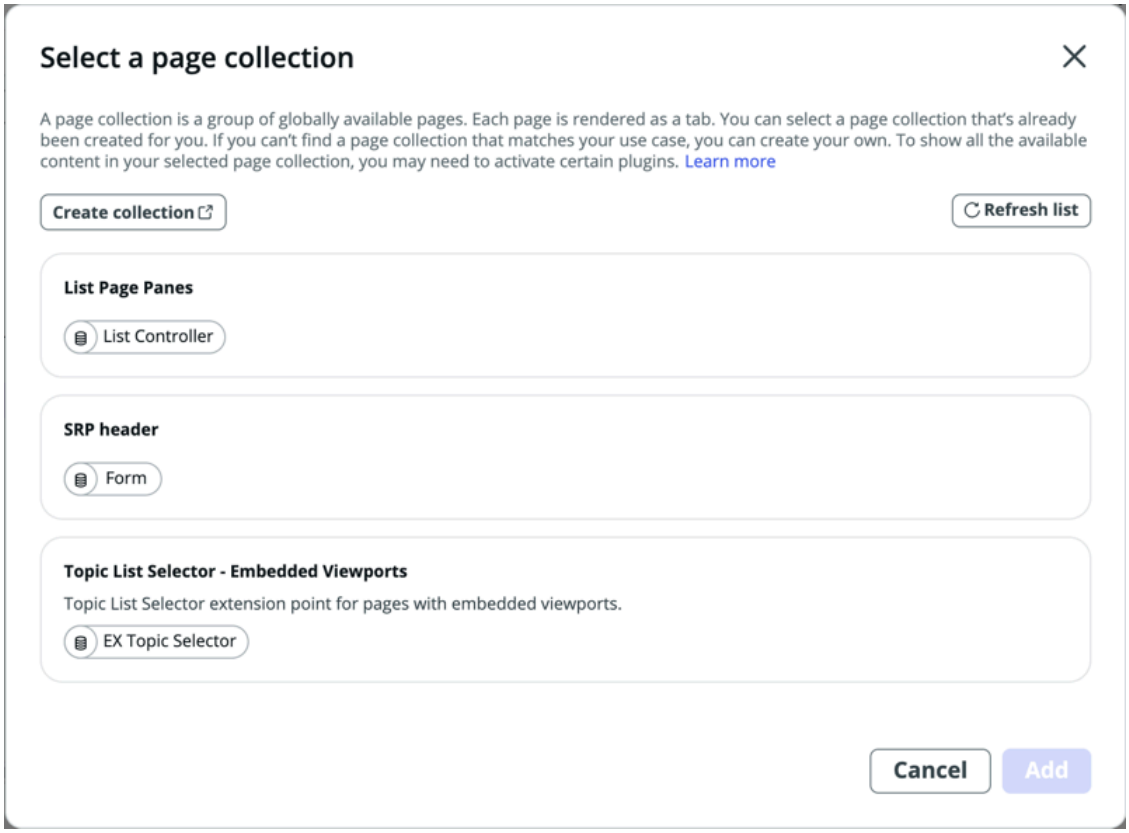
8. Select **+ Add tab** on the stage.



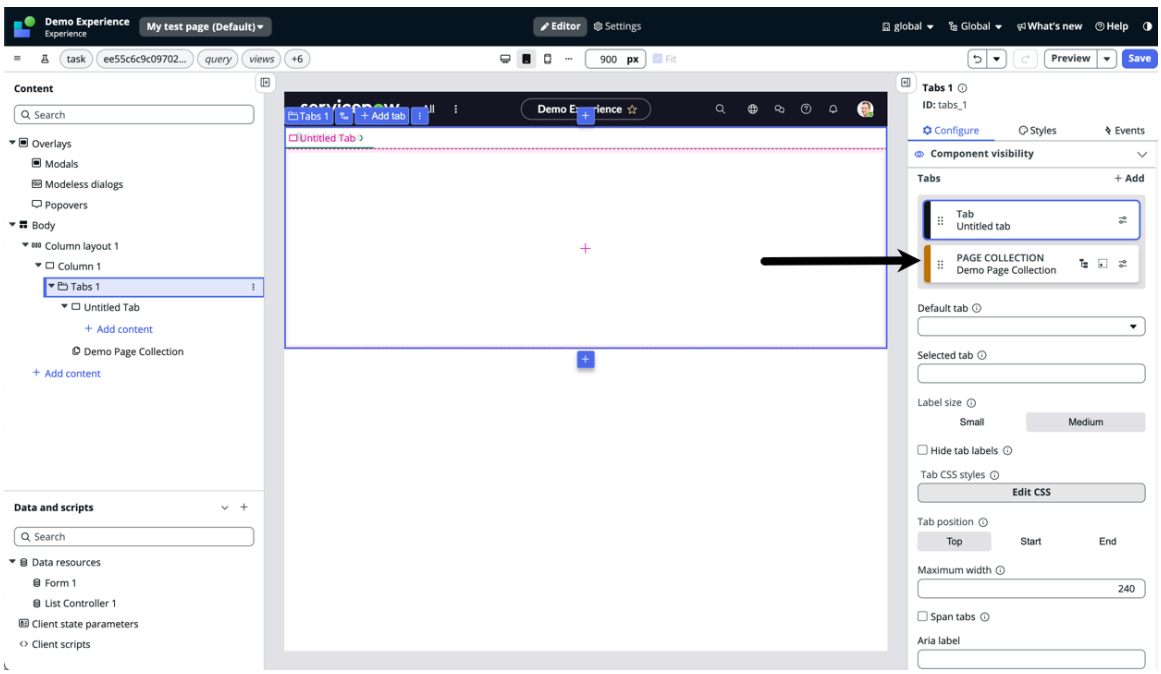
9. Select **Add a page collection**.



10. Select **Next**.
11. Select a page collection from the list or create a collection by selecting **+ Create collection**. To create your own page collection, see Create a page collection across multiple UI pages.



12. Select **Add**.
The new page collection tab displays in the **Tabs** section of the configuration panel. Only one placeholder tab displays in the page preview.



13. Select **Save**.

14. View and test your page by selecting Preview.

Add a contextual sidebar

Add a contextual sidebar to a page with UI Builder to display related content using a vertical tab structure.

Before you begin

Role required: ui_builder_admin

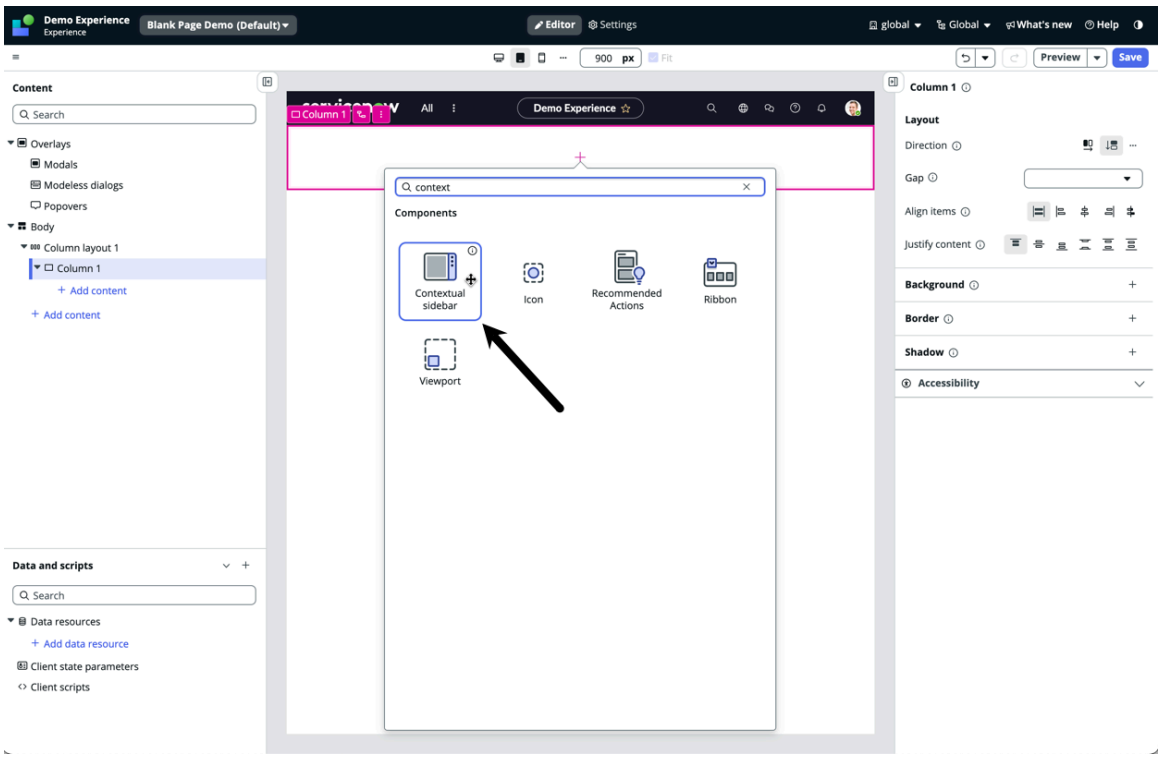
About this task

You can configure the contextual sidebar in the same way you would configure a tab component. Each vertical tab in the contextual sidebar renders custom content. You can place components in each tab for quick access directly from the page.

For example, a document creation page might have the attachments component on one tab, enabling you to attach files related to the document that you're working on. On another tab, there could be a nested comments component, enabling you to add and view comments related to the document.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder. For information about creating a page variant, see Create a page variant
4. On the stage, open the toolbox by selecting the **+ Add content** button.
5. Select the **Single column** layout tile.
6. In the column, open the toolbox by selecting the **+** icon.
7. Select the **Contextual sidebar** component to add it to your page.



The contextual sidebar appears on the stage, and the Tabs section appears in the configuration panel.

8. In the configuration panel under the Tabs section, select the settings icon (⚙) to configure the tab.

   a. On the form, fill in the fields.

   Tab settings form

Field	Description
Tab label	Label that displays on the tab for your page.
Icon	Icon that appears next to the tab label. **Note:** Icons aren't required.
Hide tab	Option to hide or display the tab.

   b. Select **Save**.

9. On the stage in the contextual sidebar, open the toolbox by selecting the **+** icon.
10. Select the component that you want to add to the tab.
    The component displays under the tab in the content tree. For more information, see Add and configure components.
11. **Optional:** To add another tab, select **+ Add** under the Tabs section.

What to do next

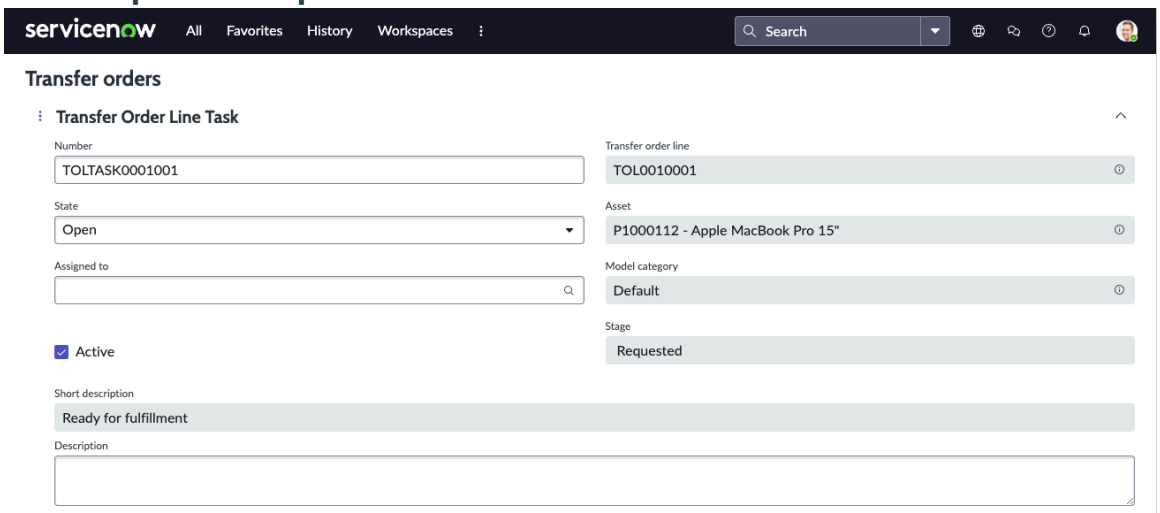
For detailed information about the configuration of the contextual sidebar and its properties, see Tabs UIB Setup on the ServiceNow Developer Site.

Add forms to UI Builder pages

Use the Form component to add one or more forms to UI Builder pages.

Add functionality to your UI Builder pages by including forms. You define the fields on the form and their properties, such as making them required. Then, collect data as the form is completed and submitted.

Form component example

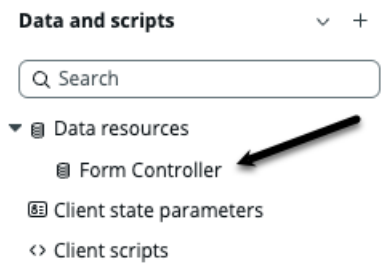


You can add more than one form to a single page. You can also add a form to a page that already contains a component with a nested form. Sample use cases include:

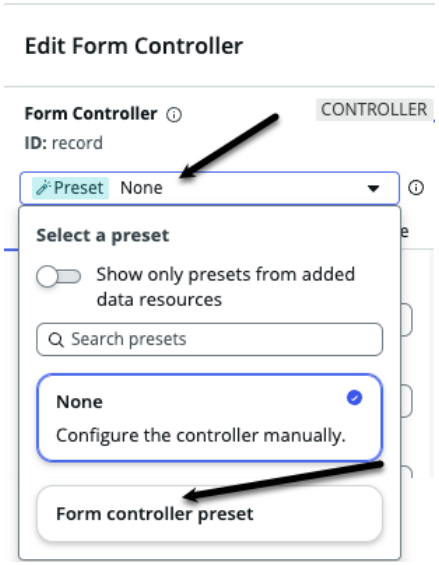
- Extend record pages by adding an inline tab with a form using its own form controller instance.
- Add modals with a form on a record page.

For existing pages with forms created in a pre-Xanadu ServiceNow release, you must apply a preset to the original form before adding another form to the page. Applying the preset is a prerequisite to adding multiple forms to a page and enables multiple forms to work as expected on a page. The form controller preset should be applied onto all form controllers.

1. Open the page containing an existing form.
2. In the data drawer, expand the **Data resources** list and select the original form controller.



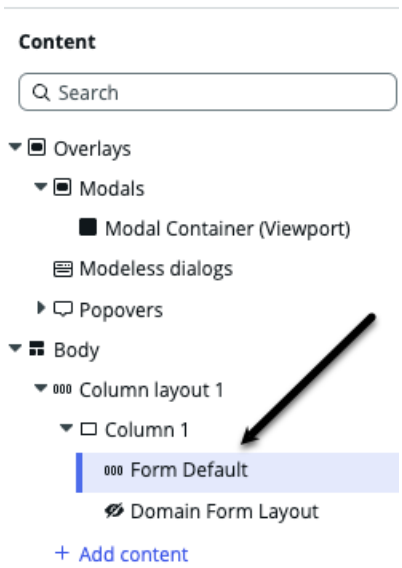
3. Select the **Preset** field.
4. Select **Form controller preset**.



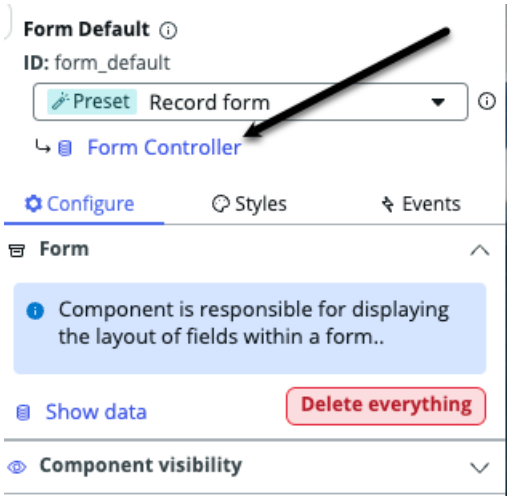
5. Select **Apply**.
6. Select the **X** to close the **Edit Form Controller** pop-up.

   Exactly one of your form controllers should have the **Is mapped to app Shell** property set to true. This property is used to specify the primary form on the page. The primary form is responsible for handling global events. You shouldn't set the property to true for more than one form controller or have zero form controllers with the property set to true.

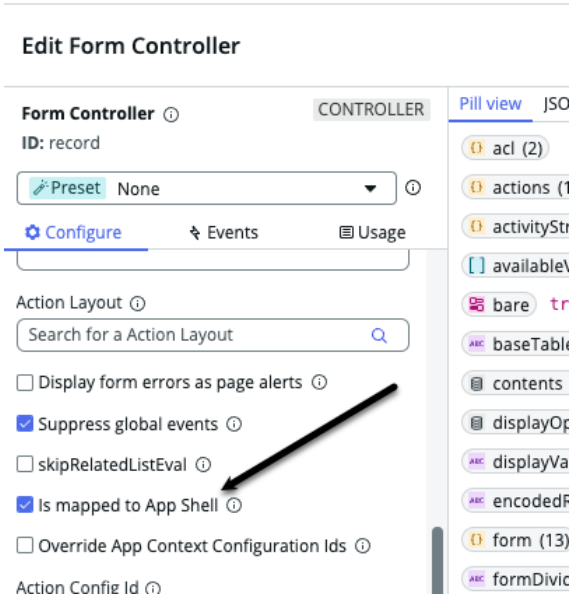
7. Open the page containing one or more forms.
8. In the content tree, select a form.



9. In the configuration panel, on the **Configure** tab, select **Form Controller**.



10. On the **Edit Form Controller** pop-up, scroll down in the **Form Controller** list to find the **Is mapped to App Shell** option.



11. Select or clear the option for each form component on the page to confirm that exactly one form controller is mapped to the app shell.

Advanced form event handling

Experienced developers with knowledge of conflict event handling may find the following details useful.

If isMapped to app shell is set to true, the form handles these events automatically:

Screen Status Changed

- Description: Action to indicate that a form is dirty.
- Output: `CTRL_RECORD#SCREEN_STATUS_CHANGED`

Update Configuration Menu Requested

- Description: Action to set record configuration menu items on the avatar menu.
- Output: `CTRL_RECORD#UPDATE_CONFIGURATION_MENU_REQUEST`

Phone Requested

- Description: Action to make a call when the CTI plugin is enabled.
- Output: `CTRL_RECORD#PHONE_REQUESTED`

Form Loading State Changed

- Description: Action to show a loading spinning when that form is loading data.
- Output: `CTRL_RECORD#FORM_LOADING_STATE_CHANGED`

For detailed information about the Form component and its properties, see Form Overview on the ServiceNow Developer Site.

Create modals in UI Builder

Use modals in UI Builder with components to provide alerts or calls to action for a user. UI Builder comes with modals to save time and effort.

What Modals are

Modals, also known as dialogs, are windows that overlay another content window. Modals take control of the user experience. Users cannot interact with the overlaid window until the modal is closed. Modals can contain different types of content such as:

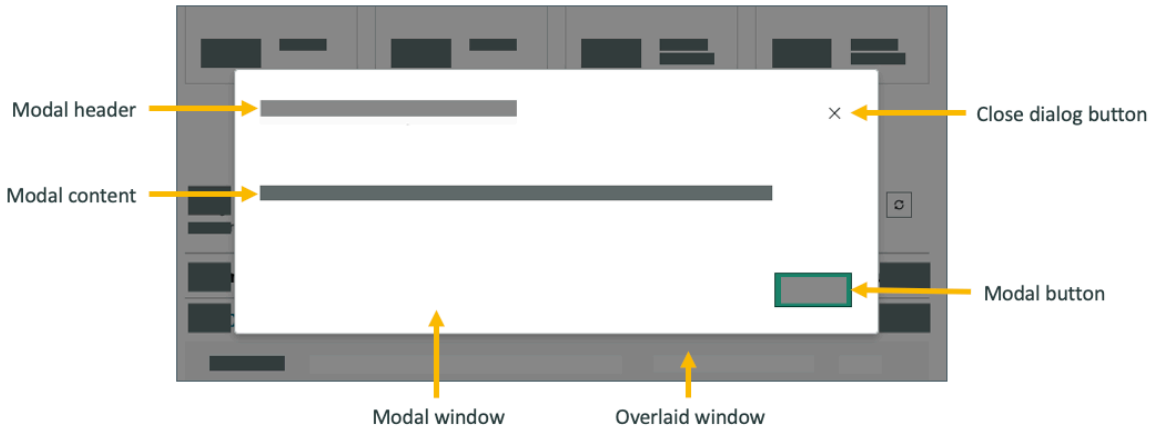
- Static text
- Dynamic text
- Forms
- Images
- Buttons

UI Builder has preconfigured modals available. You can add a modal to your component. Then, configure the content of the modal, and how it displays on the screen. Add an event handler to the modal to perform an action when a user selects it. The action can alert a user about something, or ask a user to confirm an action. A modal is a way to ensure that a user knows what is happening. For example, a modal may ask a user to confirm a selection before continuing whatever action they are performing on the main page.

Modal Anatomy

Modals in UI Builder can have:

- Modal header
- Modal content
- Close dialog button (no action taken by the modal)
- At least one Modal button (action can be taken by the modal)



Modal types

Different types of modals are available in UI Builder, as shown in the following table.

Modal type	Description
Alert	An Alert modal provides information relating to the component action. For example, when a user presses a delete button, you could have an alert pop-up that lets the user know they cannot undo a delete action.
Confirm	A Confirm modal asks a user to confirm the component action. For example, when a user presses a delete button, the user would have to confirm the deletion of data. You can choose the confirm options from the primary and secondary button label fields, such as Yes/Cancel.
Confirm and destroy	A Confirm and destroy modal is more directive, usually relating to deleting or erasing content. It lets the user know the seriousness of an action, and asks them whether they want to proceed with the action.
Custom	Custom modals address scenarios that are not handled using the standard modals. Custom modals can be thought of as a container component on a modal. You can add a custom layout, components, events, and data resources just like you do on a page. The custom modal uses layouts to let you fully design what information you want in the modal. Layouts also decide where the information sits within the modal screen. You can use Cascading Style Sheets (CSS) styling to change the visual look of the modal.
iframe	Use iframe to bring content into your modal from existing iframe content from a URL and data.

Modal type	Description
Modal viewport	Dynamically pass content into your viewport modal through an event binding using a client script.

Event handlers and modals

Expose events to modals to handle call-to-action events. For example, a primary action, secondary action, and so on. You configure the data by adding an event handler and invoking a data resource. It is as simple as adding a new event handler for the component that has a modal. Or you can add an event handler to the modal itself. You select the event that you want associated with the component or modal and add it. See Add modal to component for detailed instructions.

Add modal to component

Learn how to add a modal in UI Builder. A modal is a window that appears when you click a component. For example, a modal might display when a delete button component is clicked, prompting the user to confirm deleting a record.

Before you begin

Role required: ui_builder_admin

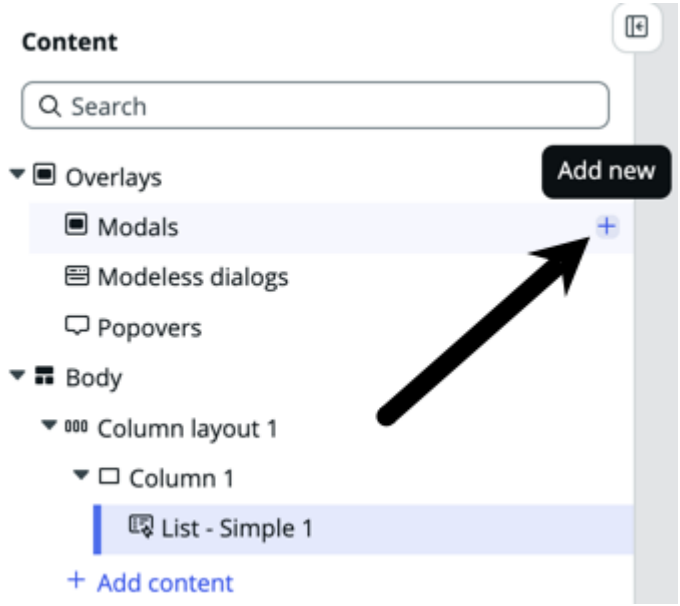
About this task

A modal is a screen that appears when an event handler is triggered by an event such as a button being clicked. The following procedure shows an example of how to add a Confirm modal and an associated event handler to a button component.

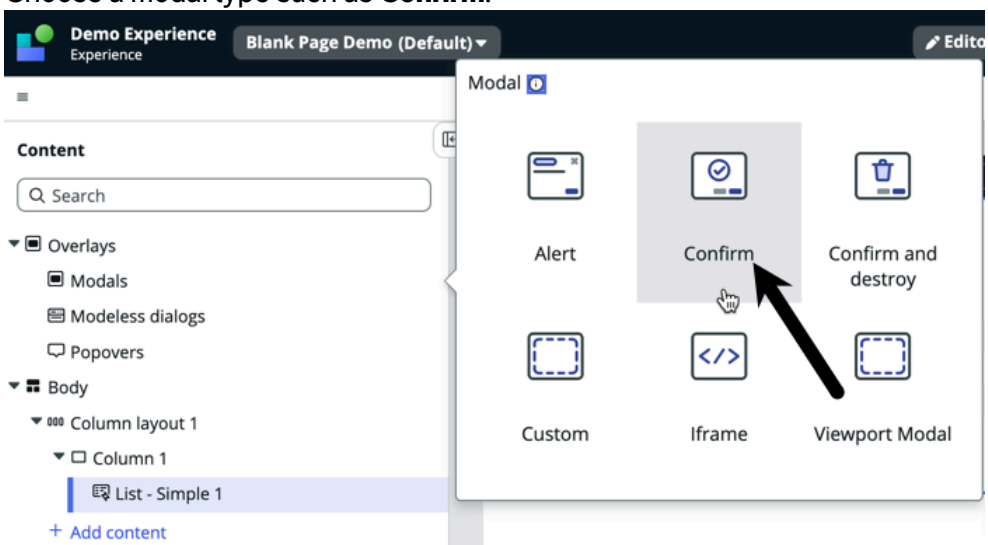
Procedure

1. Navigate to **All** > **Now experience framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Open or create a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Add a modal to the page.

a. Select the **+** icon in the content tree next to **Modals and popovers**.



b. Choose a modal type such as **Confirm**.



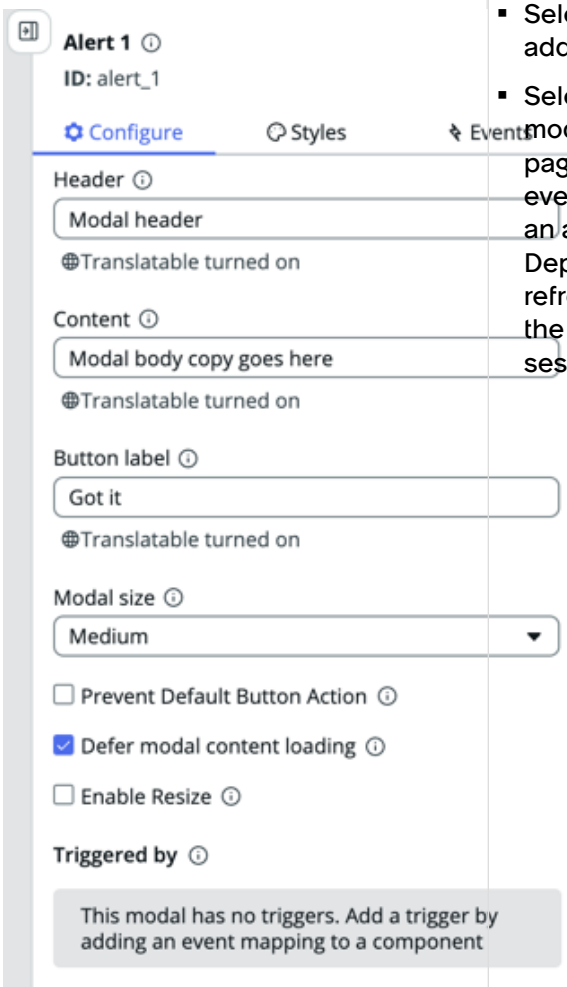
Modal type	Description
Alert	An Alert modal provides information relating to the component action. For example, when a user presses a delete button, you could have an alert pop-up that lets the user know they cannot undo a delete action.
Confirm	A Confirm modal asks a user to confirm the component action. For example, when a user presses a delete button, the user would have to confirm the deletion of data. You can choose the confirm options from the primary and secondary button label fields, such as Yes/Cancel.

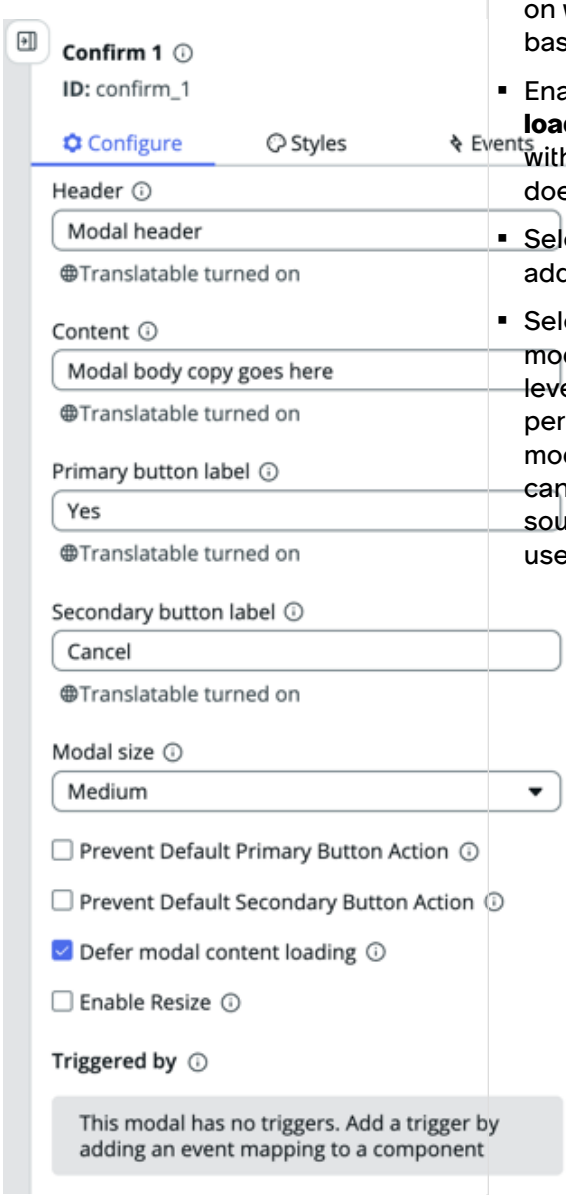
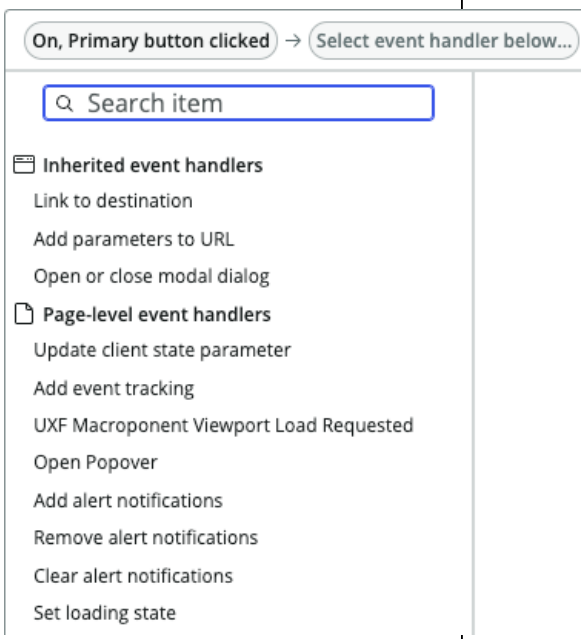
Modal type	Description
Confirm and destroy	A Confirm and destroy modal is more directive, usually relating to deleting or erasing content. It lets the user know the seriousness of an action, and asks them whether they want to proceed with the action.
Custom	The Custom modal uses layouts to let you fully design what information you want in the modal. Layouts also decide where the information sits within the modal screen. You can use Cascading Style Sheets (CSS) styling to change the visual look of the modal, such as background color.
iframe	Use iframe to bring content into your modal using existing iframe content from a URL and data.
Modal viewport	Dynamically pass content into your viewport modal through an event binding using a client script. See Bind an event to a component for more information on binding an event to a component.

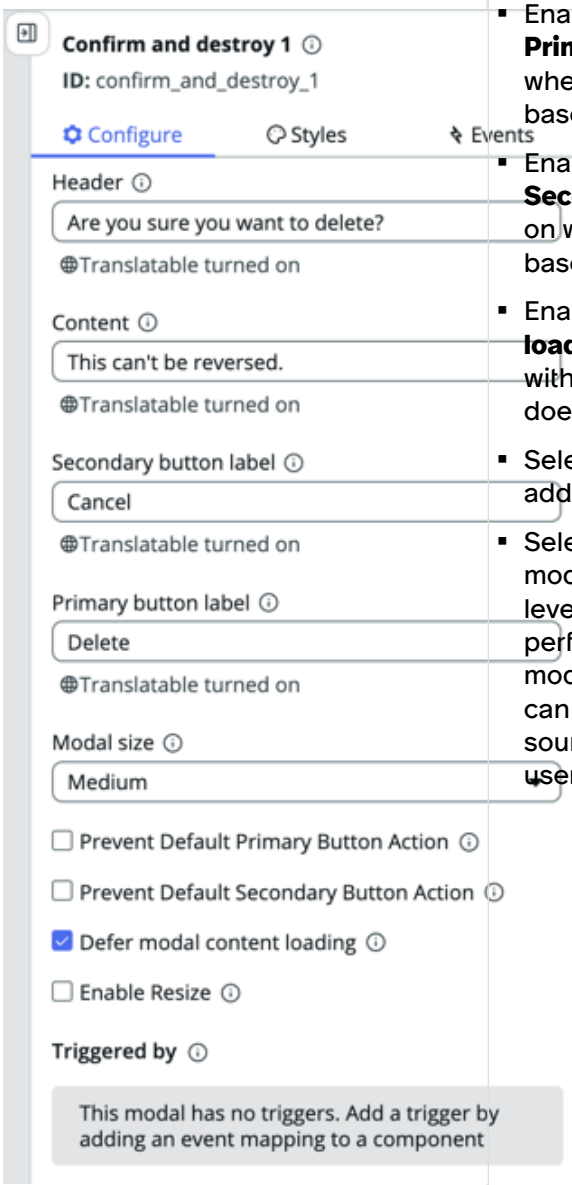
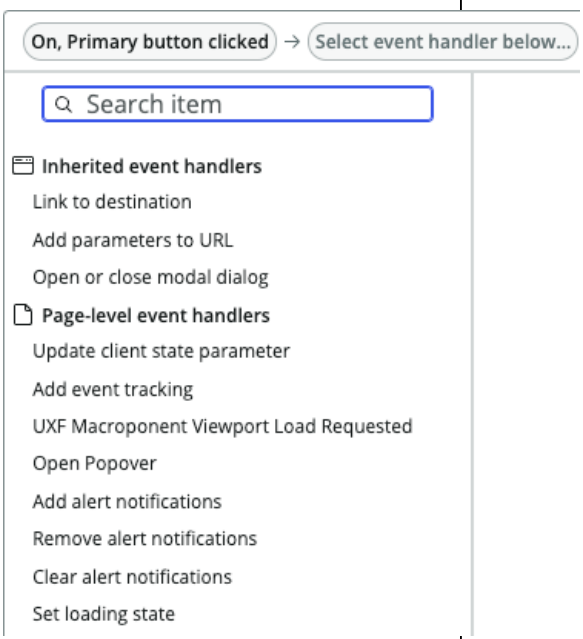
c. Configure the modals as shown in the table.

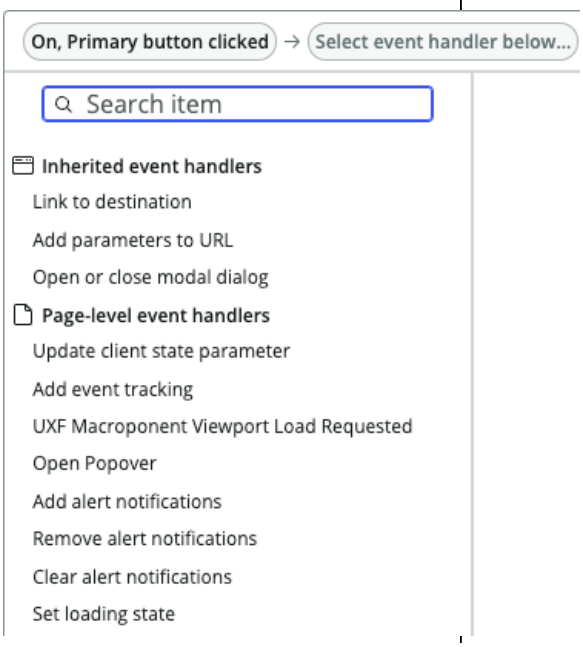
Configure each modal differently, depending on what the modal requires. Change what information goes into the modal, the size of the modal, and how it looks. You can add an event handler to the modal that performs the action for the modal, such as opening or closing the modal.

Modal	Configuration
Alert	▪ Add a header, which is the title of the modal. ▪ Write the content that you want displayed in the modal. The content tells the user what the alert is. ▪ Add text for the button label. It can be anything you want, such as **OK**, **Yes**, and so on. ▪ Choose the size of the modal on the screen. Select **Small**, **Medium**, **Large**, or **Fullscreen**. ▪ Enable or disable **Prevent Default Button Action**, depending on whether you want the modal to close based on the default action. ▪ Enable or disable **Defer modal content loading**. If you disable it, the modal loads with the page. If you enable it, the modal doesn't load when the page loads.

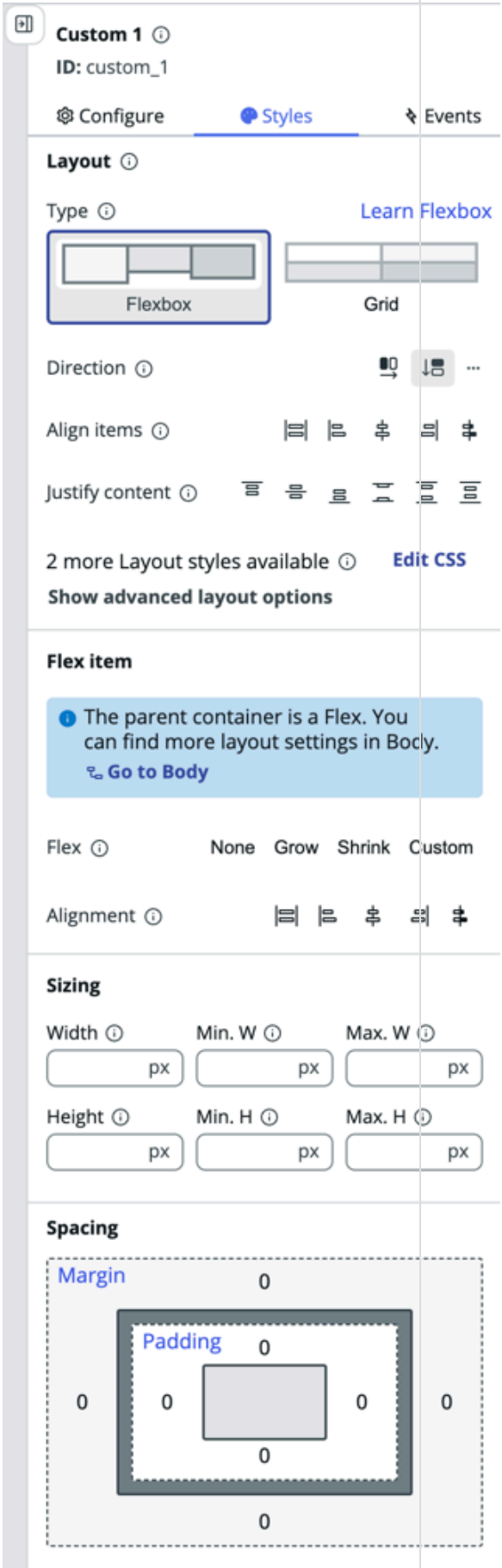
Modal	Configuration
	▪ Select **Events** > **Add event mapping** to add an event handler to the modal. ▪ Select an event handler to apply to the modal, then select **Add** to add it to the page. Choose from inherited or page-level event handlers. Event handlers perform an action such as open or close a modal. Depending on the modal type, you can refresh data for the App Shell data source, the user session for GraphQL, or a user session for Transform.
Confirm	▪ Add a header, which is the title of the modal. ▪ Write the content that you want displayed in the modal. The content tells the user what the alert is. ▪ Add text for the primary button. The primary button is the main action button for users, such as **Yes**, **Add**, and so on. ▪ Add text for the secondary button. The secondary button is usually the no choice for users, such as **Cancel**, **No**, and so on. ▪ Choose the size of the modal on the screen. Select **Small**, **Medium**, **Large**, or **Fullscreen**. ▪ Enable or disable **Prevent Default Primary Button Action**, depending on whether you want the modal to close based on the default action. ▪ Enable or disable **Prevent Default Secondary Button Action**, depending

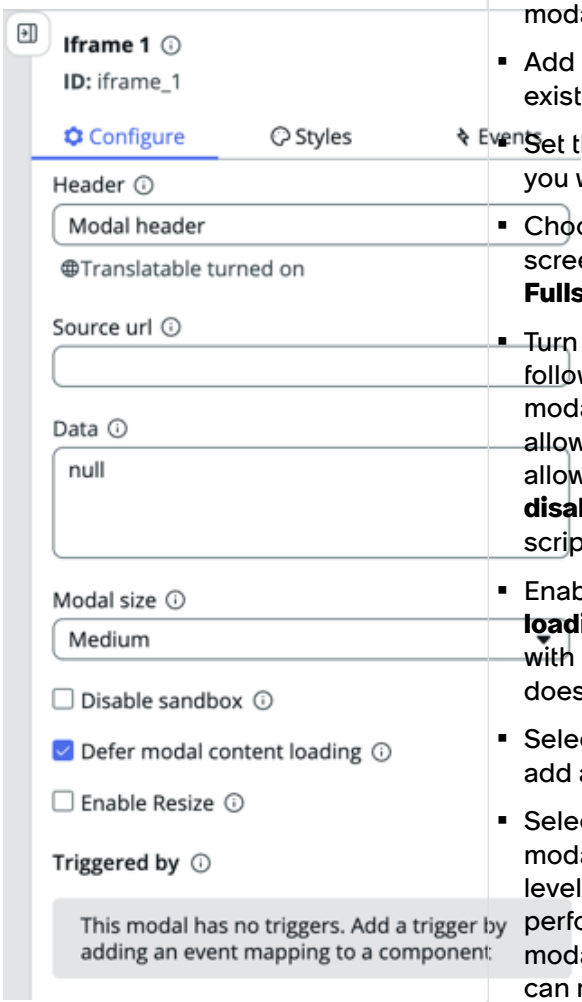
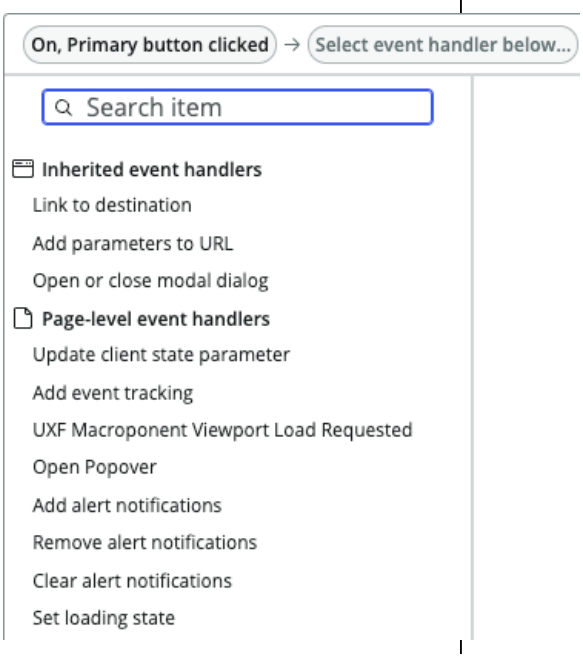
Modal	Configuration
 	on whether you want the modal to close based on the default action. ▪ Enable or disable **Defer modal content loading**. If you disable it, the modal loads with the page. If you enable it, the modal doesn't load when the page loads. ▪ Select **Events** > **Add event mapping** to add an event handler to the modal. ▪ Select an event handler to apply to the modal. Choose from inherited or page-level event handlers. Event handlers perform an action such as open or close a modal. Depending on the modal type, you can refresh data for the App Shell data source, the user session for GraphQL, or a user session for Transform.
Confirm or destroy	▪ Add a header, which is the title of the modal. ▪ Write the content that you want displayed in the modal. The content tells the user what the alert is. ▪ Add text for the primary button. Primary is the main action button for users, such as **Delete** or **Erase**. ▪ Add text for the secondary button. The secondary button is usually the no choice for users, such as **Cancel**, **No**, and so on. ▪ Choose the size of the modal on the screen. Select **Small**, **Medium**, **Large**, or **Fullscreen**.

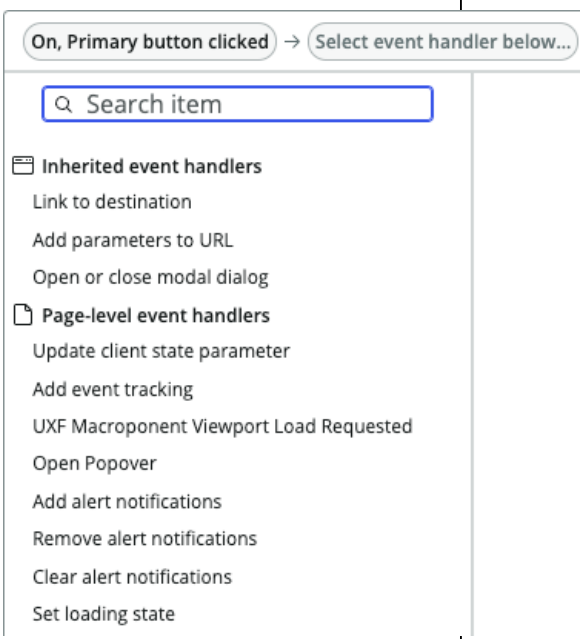
Modal	Configuration
 	▪ Enable or disable **Prevent Default Primary Button Action**, depending on whether you want the modal to close based on the default action. ▪ Enable or disable **Prevent Default Secondary Button Action**, depending on whether you want the modal to close based on the default action. ▪ Enable or disable **Defer modal content loading**. If you disable it, the modal loads with the page. If you enable it, the modal doesn't load when the page loads. ▪ Select **Events** > **Add event mapping** to add an event handler to the modal. ▪ Select an event handler to apply to the modal. Choose from inherited or page-level event handlers. Event handlers perform an action such as open or close a modal. Depending on the modal type, you can refresh data for the App Shell data source, the user session for GraphQL, or a user session for Transform.
Custom	▪ Choose a layout for your modal. You can use a flexbox or CSS grid layout. These layouts let you add content in your modal however you want. ▪ Use styling options to change how your modal looks. You can apply any standard CSS styling to your modal, such as background color and padding. For more information about styling, see Change the default appearance of components.

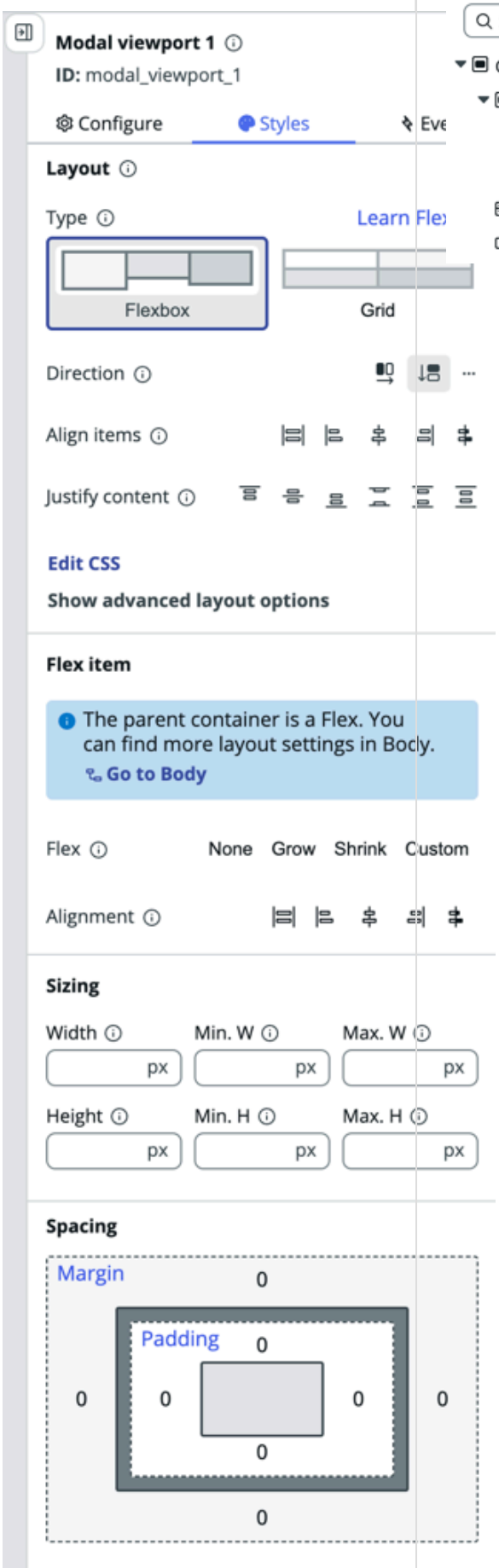
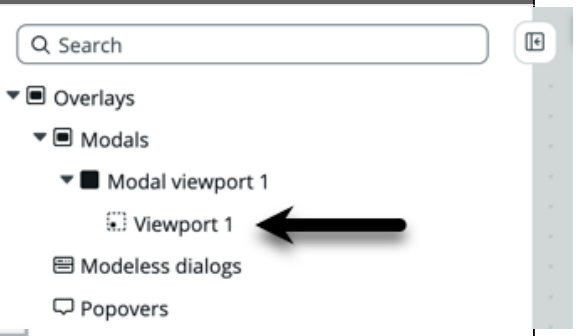
Modal	Configuration
	▪ Add components to the containers within the custom modal. ▪ Select **Events** > **Add event mapping** to add an event handler to the modal. ▪ Select an event handler to apply to the modal. Choose from inherited or page-level event handlers. Event handlers perform an action such as open or close a modal. Depending on the modal type, you can refresh data for the App Shell data source, the user session for GraphQL, or a user session for Transform. 

Modal	Configuration



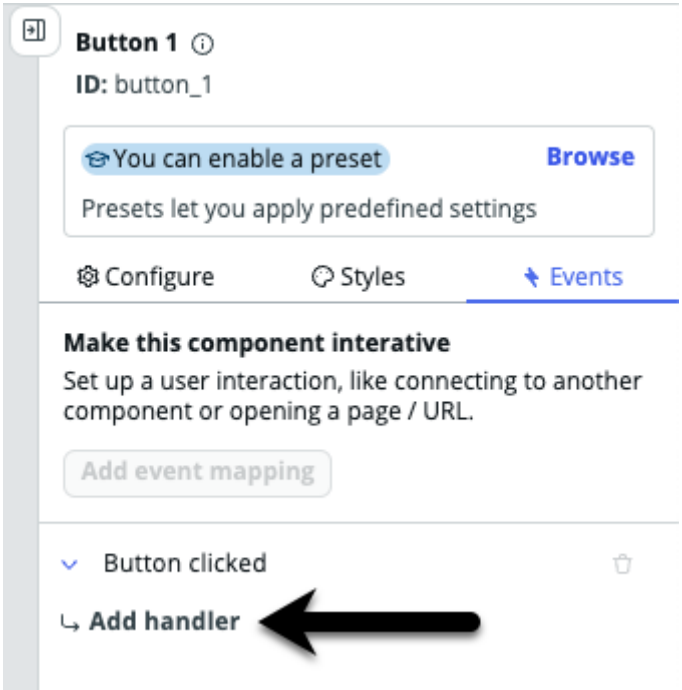
Modal	Configuration
iframe 	▪ Add a header, which is the title of the modal. ▪ Add a source URL that points to your existing iframe content. ▪ Set the parameters and initial data that you want to iframe. ▪ Choose the size of the modal on the screen. Select **Small**, **Medium**, **Large**, or **Fullscreen**. ▪ Turn on **disable sandbox** to lift the following restrictions: allow-forms, allow-modals, allow-popups, allow-presentation, allow-same-origin, allow-scripts, and allow-downloads options. Turn off the **disable sandbox** to only lift the allow-scripts option. ▪ Enable or disable **Defer modal content loading**. If you disable it, the modal loads with the page. If you enable it, the modal doesn't load when the page loads. ▪ Select **Events** > **Add event mapping** to add an event handler to the modal. ▪ Select an event handler to apply to the modal. Choose from inherited or page-level event handlers. Event handlers perform an action such as open or close a modal. Depending on the modal type, you can refresh data for the App Shell data source, the user session for GraphQL, or a user session for Transform. 

Modal	Configuration
Modal viewport	▪ Choose the size of the modal on the screen. Select **Small**, **Medium**, **Large**, or **Fullscreen**. ▪ Enable or disable **Hide Padding**. ▪ Enable or disable **Hide Close Button**. If you enable it, the close button does not display in the modal. ▪ Enable or disable **Defer modal content loading**. If you disable it, the modal loads with the page. If you enable it, the modal doesn't load when the page loads. ▪ Select **Events** > **Add event mapping** to add an event handler to the modal. ▪ Select an event handler to apply to the modal. Choose from inherited or page-level event handlers. Event handlers perform an action such as open or close a modal. Depending on the modal type, you can refresh data for the App Shell data source, the user session for GraphQL, or a user session for Transform.  ▪ Select the viewport component within the modal viewport.

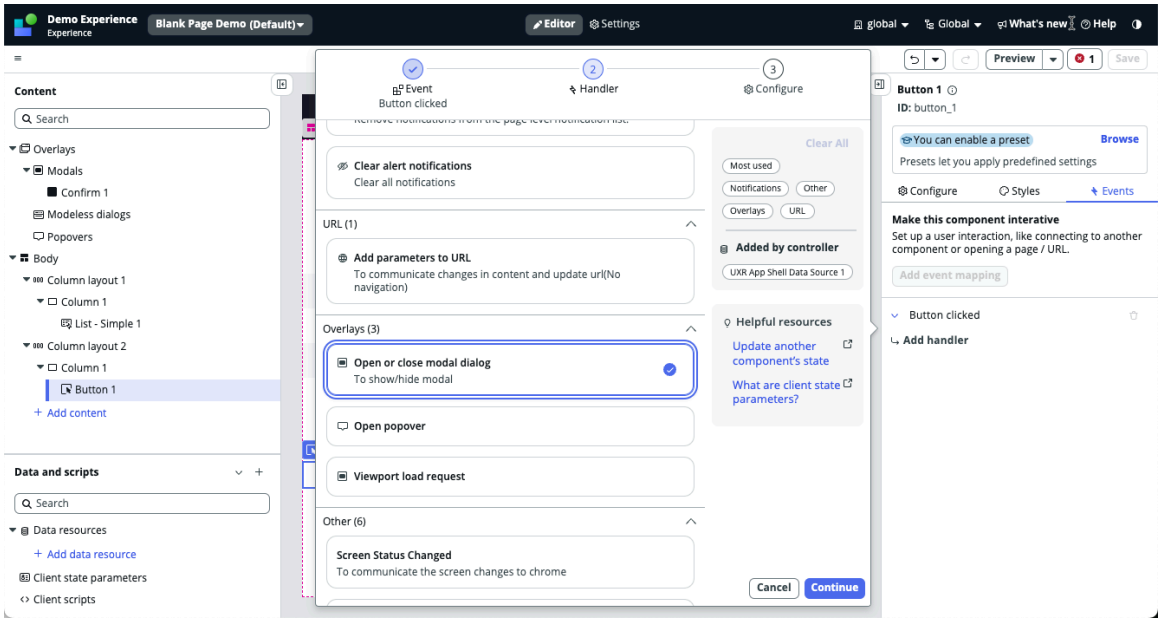
Modal	Configuration
	

Modal	Configuration
	▪ Select **+ Add** in the Configure tab to add a page collection to the viewport. ▪ Select a page collection from the list and click **Add**.

5. Add a component to your page to trigger the modal you just added, such as a button component.
   See Add and configure components for more information.
6. Select the **Events** tab in the configuration panel.
7. Select **+ Add event handler**.

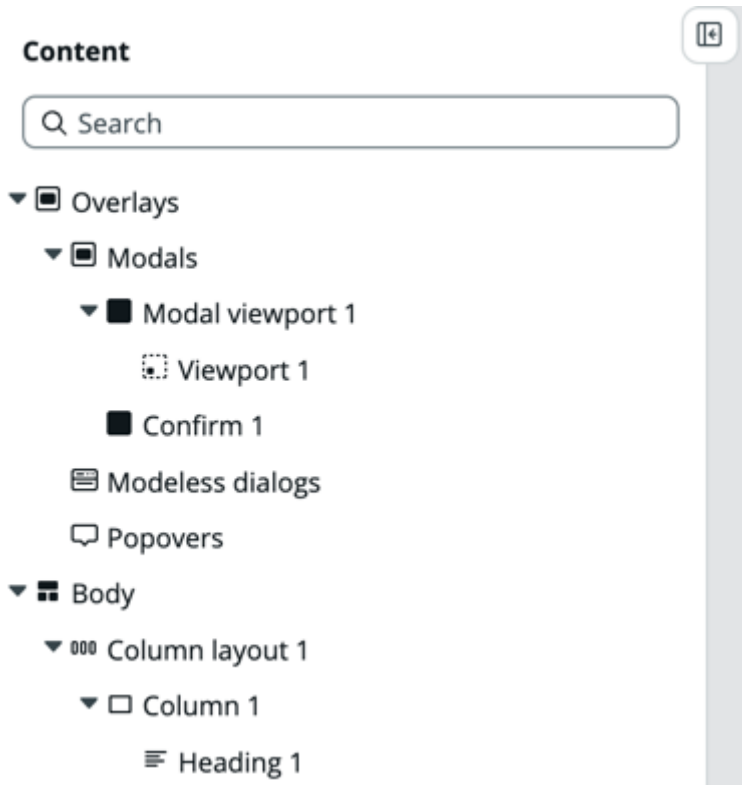


8. Select **Open or close modal dialog**.
9. Select the modal you created in the previous steps using the **Modal** drop-down.
10. Click **Add**.



11. When you finish configuring the modal, close it.

Notice in the content tree that the modals you create sit above the body of your page structure.



12. Click **Save**.

Extend your UI experience with viewport components

Viewports are specialized components that enable you to extend your experience without needing to own the parent page in UI Builder. You can work with viewports in three ways. You can add a viewport component or a viewport-enabled tab to a page, or add a viewport to the Contextual sidebar component.

Add a viewport component to your page to create separate content even if you do not own the content of the page itself. Create subpages, specify audiences, or customize content with

different data, routes, and screens. For more information about viewport components, see Add a viewport component to your page.

You can replace Tabs components with Viewport-enabled tabs as a way to display third-party custom data, assign audiences, and create variants. For more information about viewport-enabled tabs, see Replace a tab with a viewport-enabled tab.

Use the **Edit content** mode of UI Builder to add viewports or tab sets to a page or Contextual sidebar component. Edit tabs in a Contextual sidebar component as you would on a page using the **Config** panel. The main difference is, when you are in the **Edit content** mode of a tab, the page management area changes to **Tab management**.

Select the menu icon ⋮ to edit your tab settings, or change the required or optional parameters.

Add viewport modals to your experience to embed subpages or other experiences within a modal in your parent page/experience. For more information about adding viewport modals to your experience, see Add a viewport modal to your experience.

Add a viewport component to your page

Add a viewport component to your page and create a subpage to create separate content on the page.

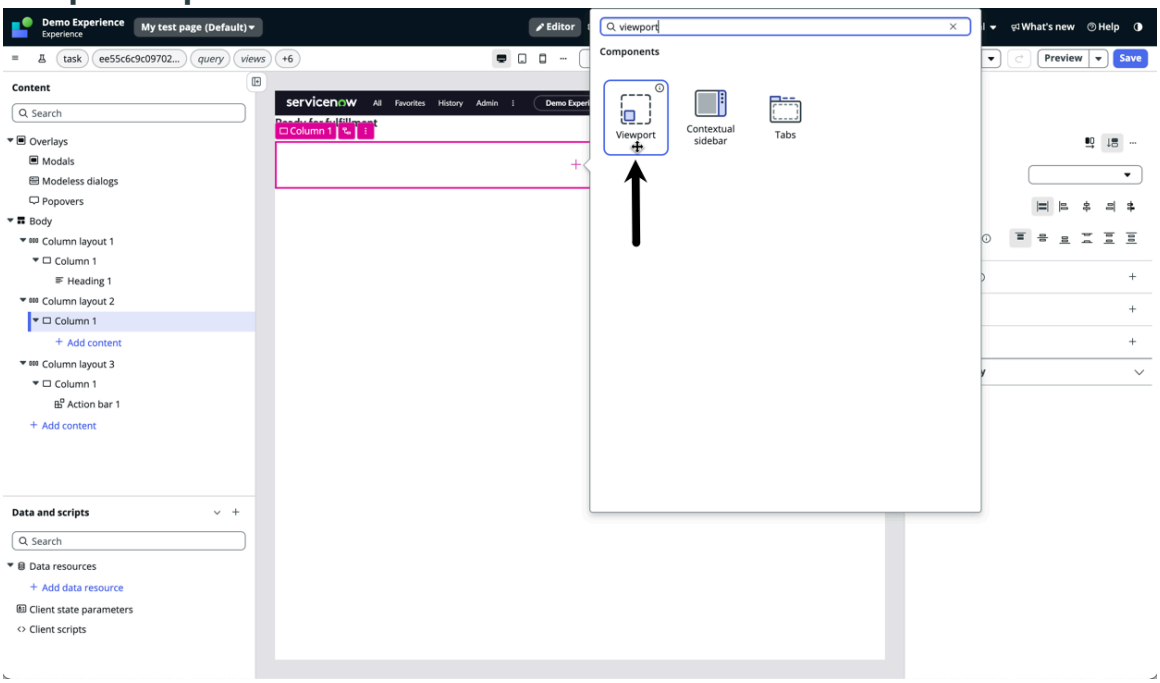
Before you begin

Role required: ui_builder_admin

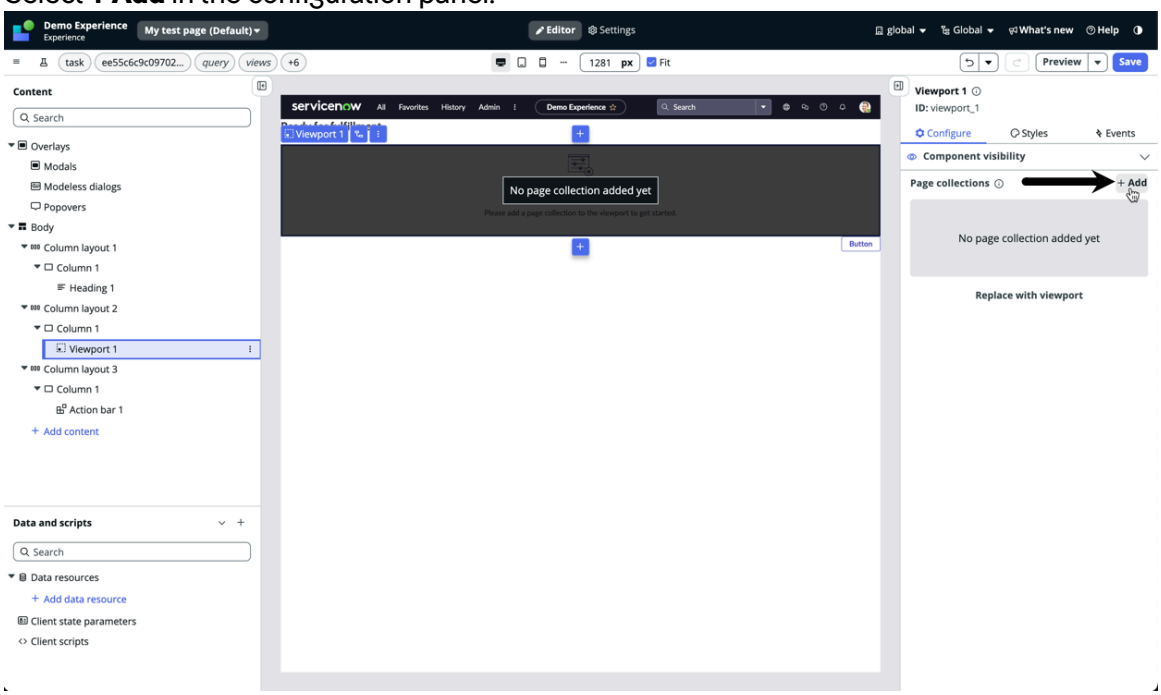
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Open the editor for the page variant that you want to add the viewport to. If you haven't created a page for your experience, follow the steps to Create a page in UI Builder.
4. Select **+ Add content** in the content tree.
5. Select a **Single column** layout.
6. Next, select the **+** icon in the column to open the toolbox.
7. Enter `viewport` in the search box.
8. Select **Viewport**.

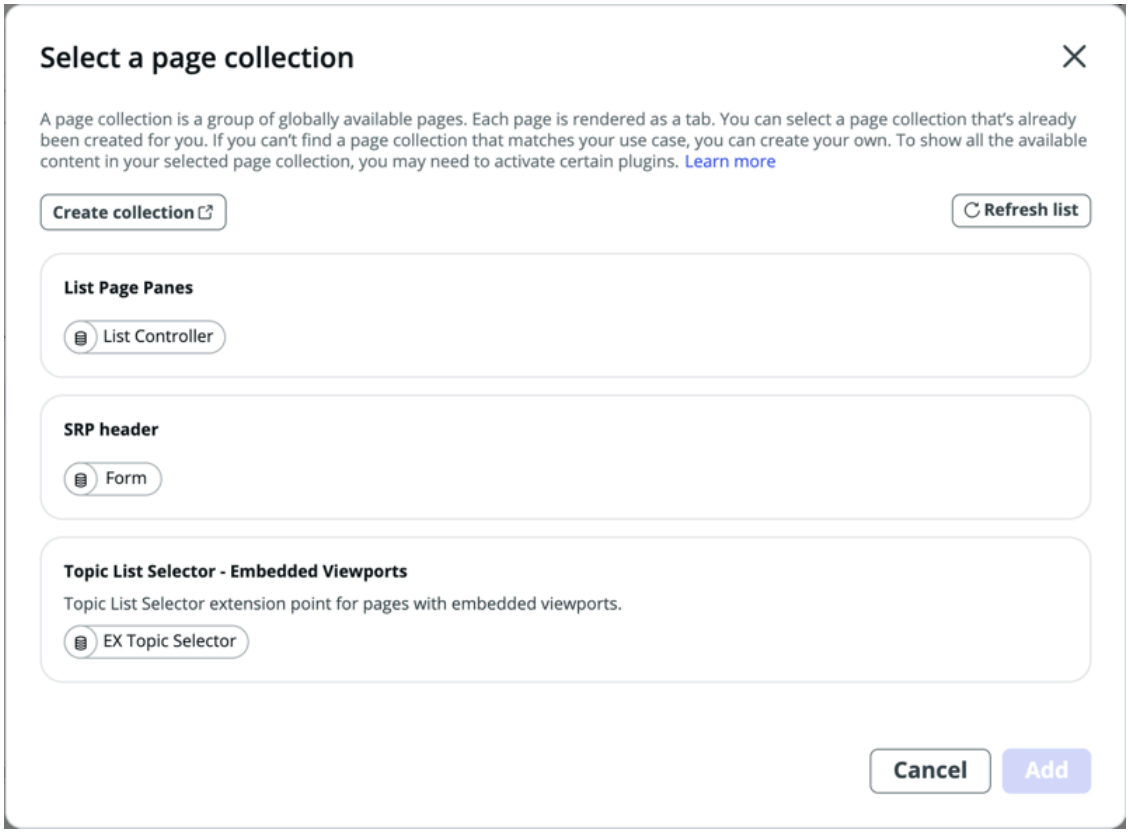
Viewport component



9. Select the viewport component in the content tree.
10. Select **+ Add** in the configuration panel.



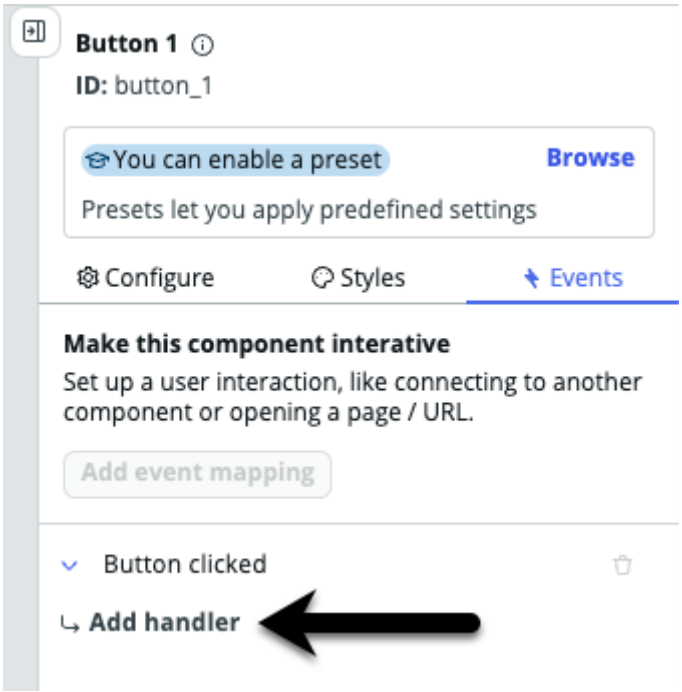
11. Select a page collection from the list or create a collection by selecting **+ Create collection**. For more information on creating your own page collection, see Create a page collection across multiple UI pages.



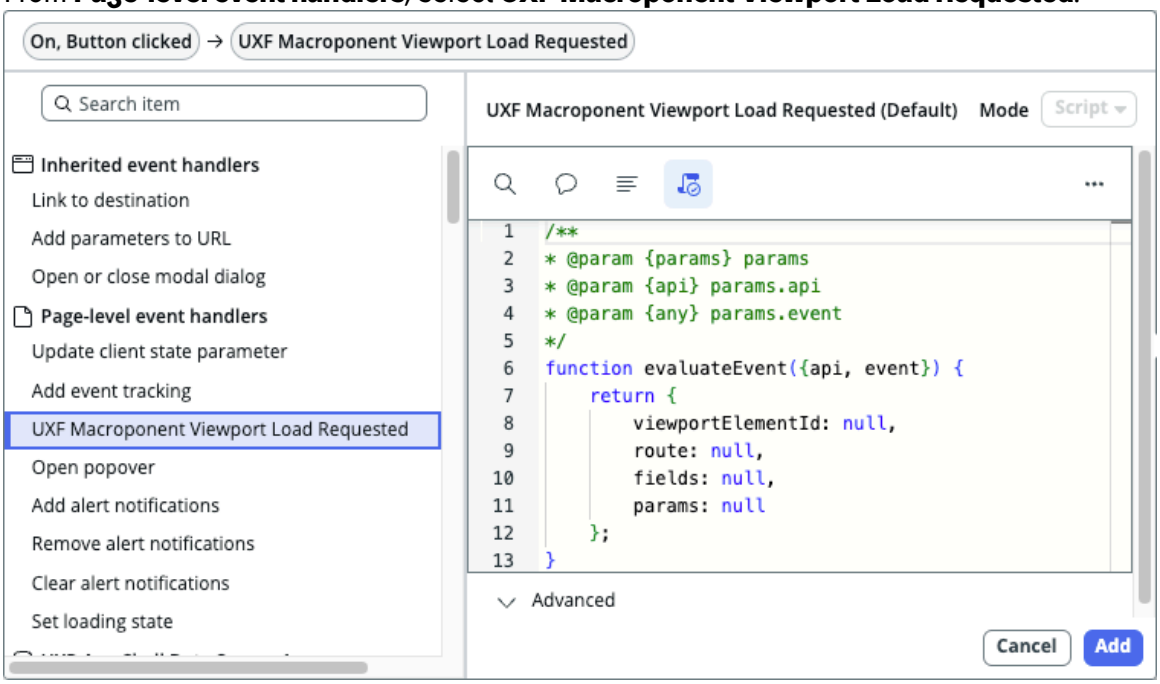
12. Select **Add**.
    The page collection displays in the **Page collections** section of the configuration panel. Only one page of the page collection displays in the staging area.
13. Select **Save**.
14. Locate the viewport ID and route.

    You can see the ID and route in the Config panel as shown in the following figure.

15. Add a component to your page to open the viewport you just added, such as a button component.
    For more information, see Add and configure components.
16. Select the **Events** tab in the configuration panel.
17. Select **+ Add event handler** to view the viewport.

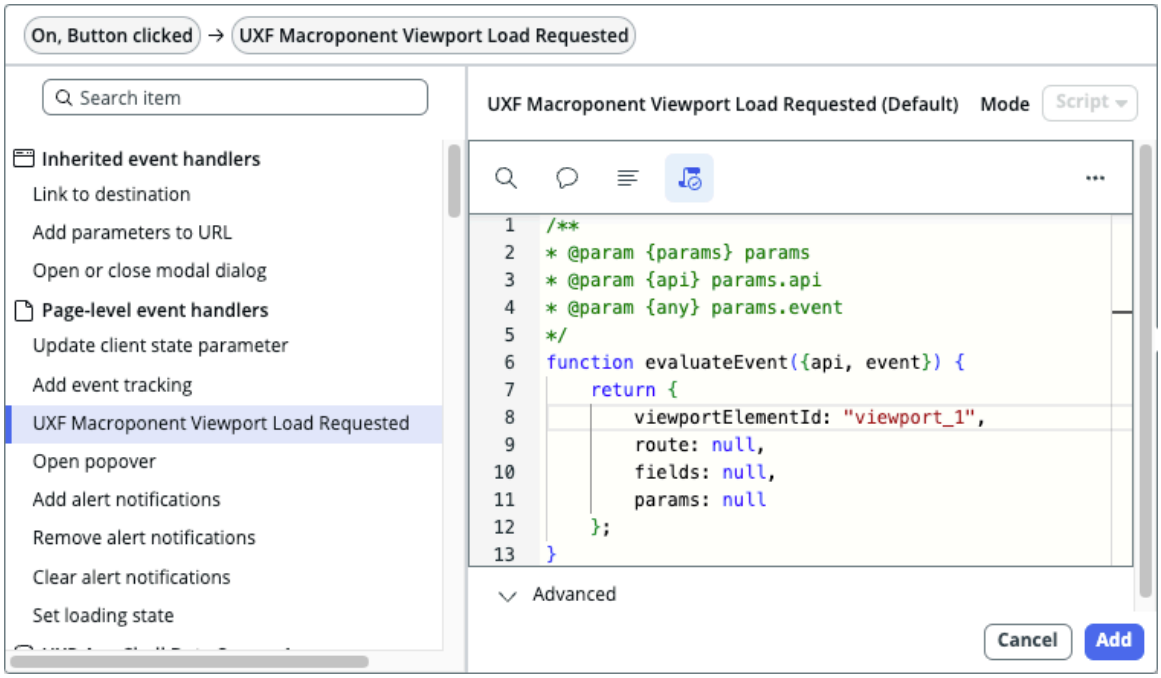


18. From **Page-level event handlers**, select **UXF Macroponent Viewport Load Requested**.



19. In the **viewportElementID** element, replace the `null` value with the viewport ID that you located in a previous step.
In this example, it is, `"viewport_1"`.

20. Select **Add**.



21. Select **Save**.
22. View and test your page by selecting Preview.

Add a viewport modal to your experience

Add a viewport within a modal in your experience.

Before you begin

Role required: ui_builder_admin

About this task

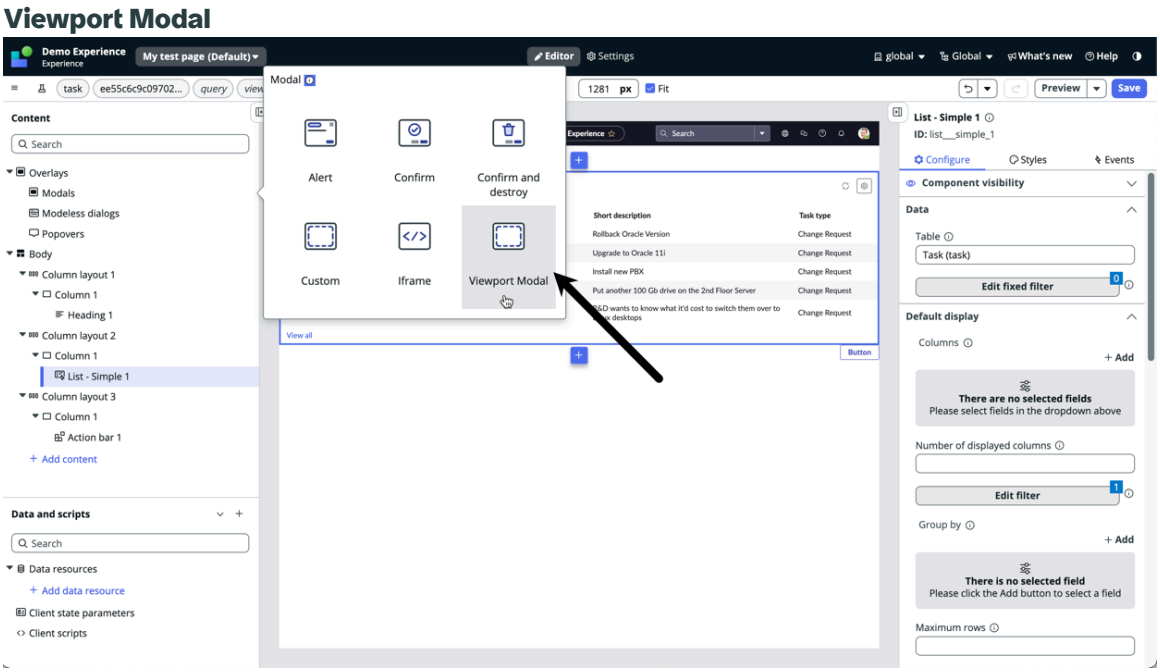
Use viewport modals to embed subpages or other experiences within a modal in your parent page or experience. Viewport modals can be opened via events or scripts. Viewport modals are limited to one subroute per viewport modal. You can create additional viewport modals for extra routes.

Note:
Legacy viewport modals cannot be upgraded and must be recreated to take advantage of the new functionality.

Procedure

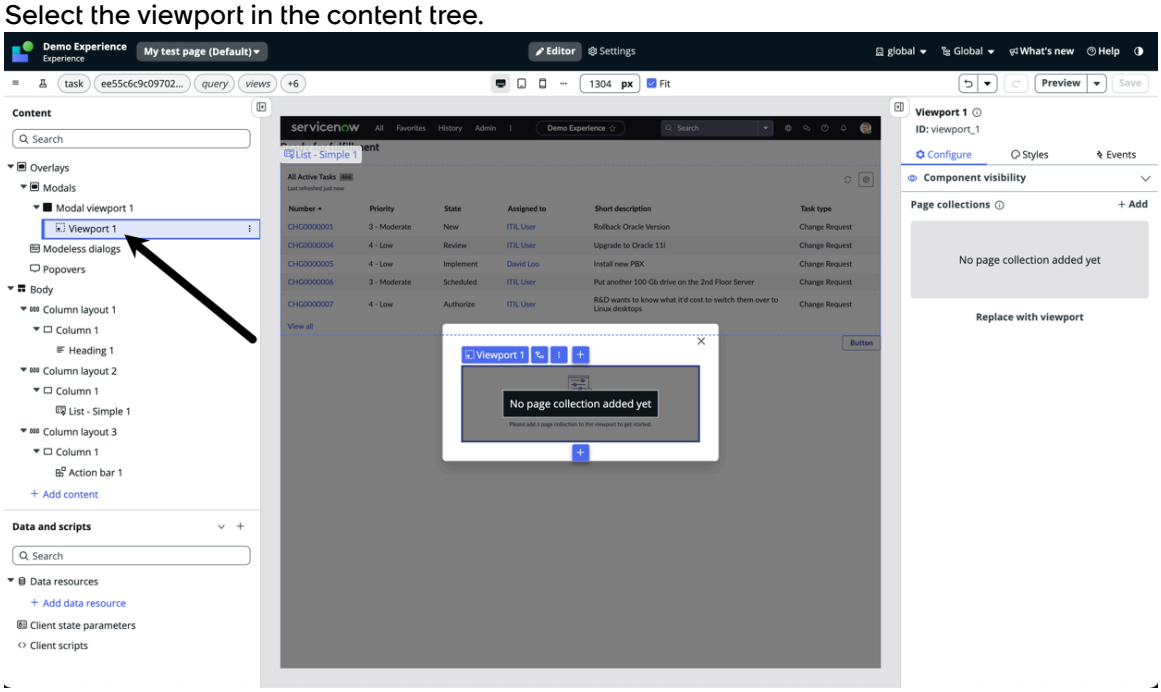
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information.
3. Open or create a page.
   If you open an existing page, ensure you are in the same scope as the original page. If not, change the scope before you start editing the page. Application scoping protects applications by identifying and restricting access to application files and data. Administrators set the scope to specify what parts of an application are accessible to other applications. Application scope protects data and application files. See Learn about security and roles for more information on application scope.
4. Select **+ Add content** in the content tree.
5. Select a **Single column** layout.

6. Select **+** next to **Modals** in the content tree.
7. Select **Viewport Modal** in the list.

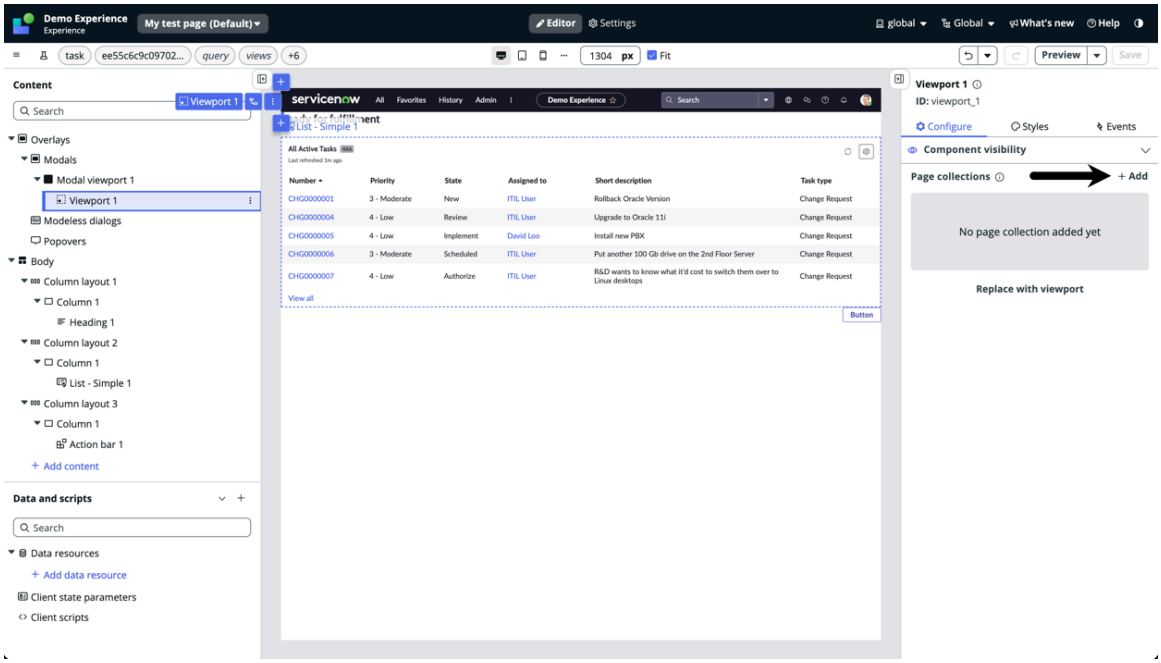


A viewport model appears above your page.

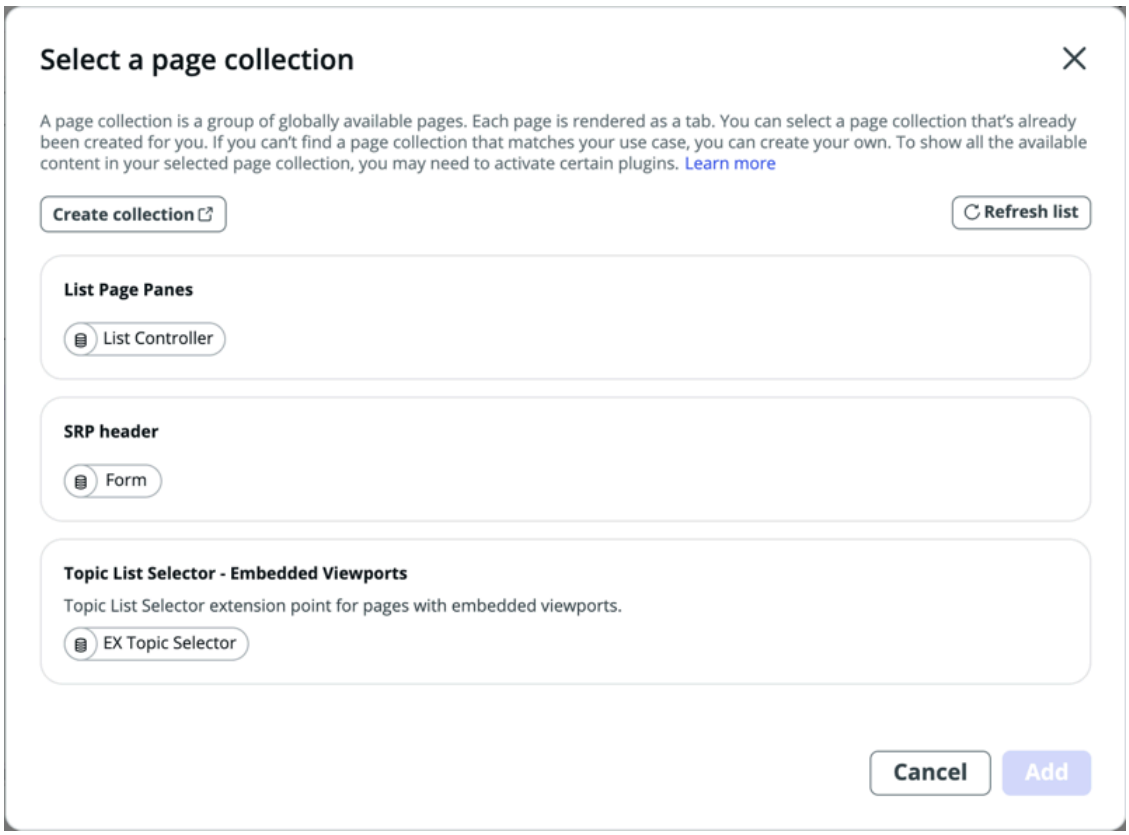
8. Click **Save**.
9. Select the viewport in the content tree.



10. Select **+ Add** next to **Page collections** in the configure tab.

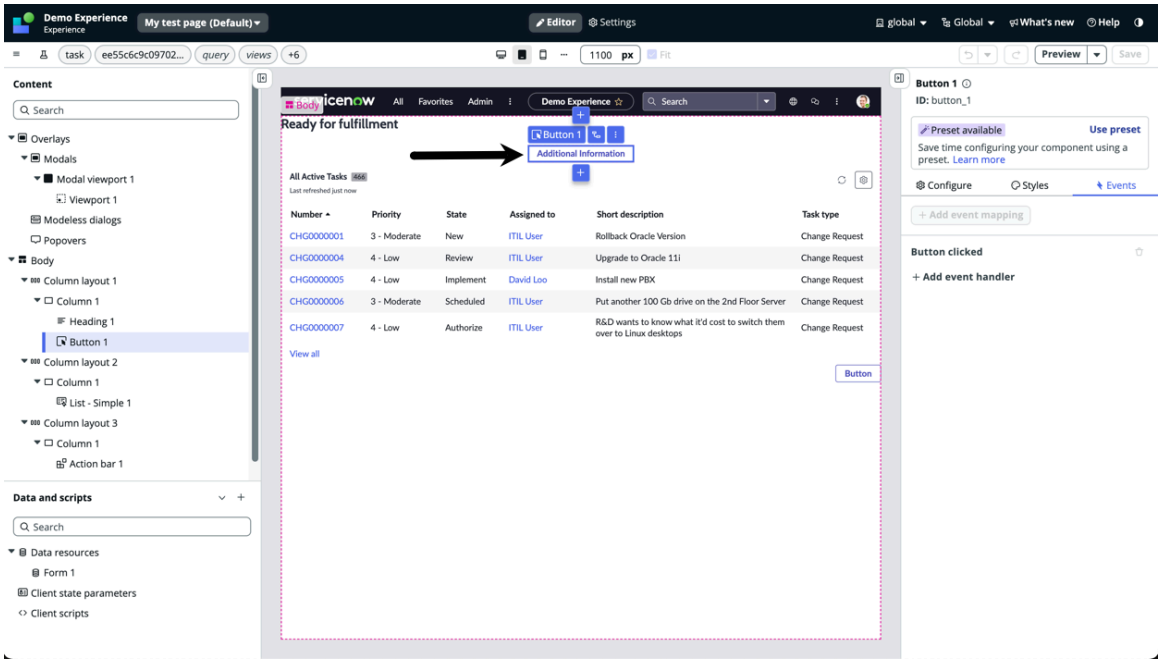


11. Select a page collection or create a new one.
    For more information, see Create a page collection across multiple UI pages.

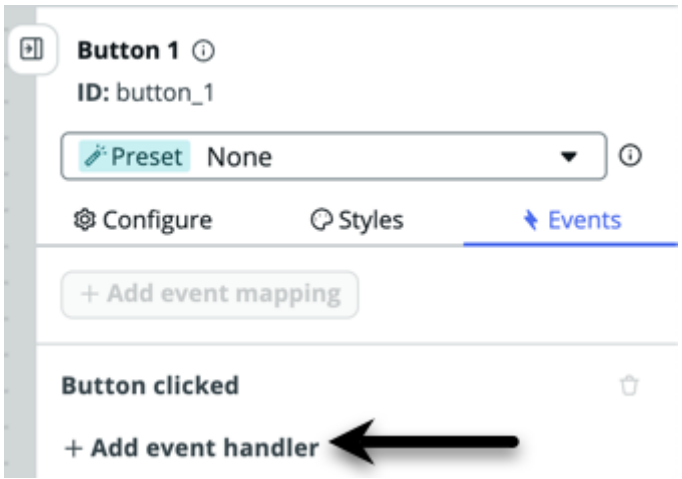


12. Click **Add**.
13. Click **Save**.
14. Add a component that opens the viewport modal.

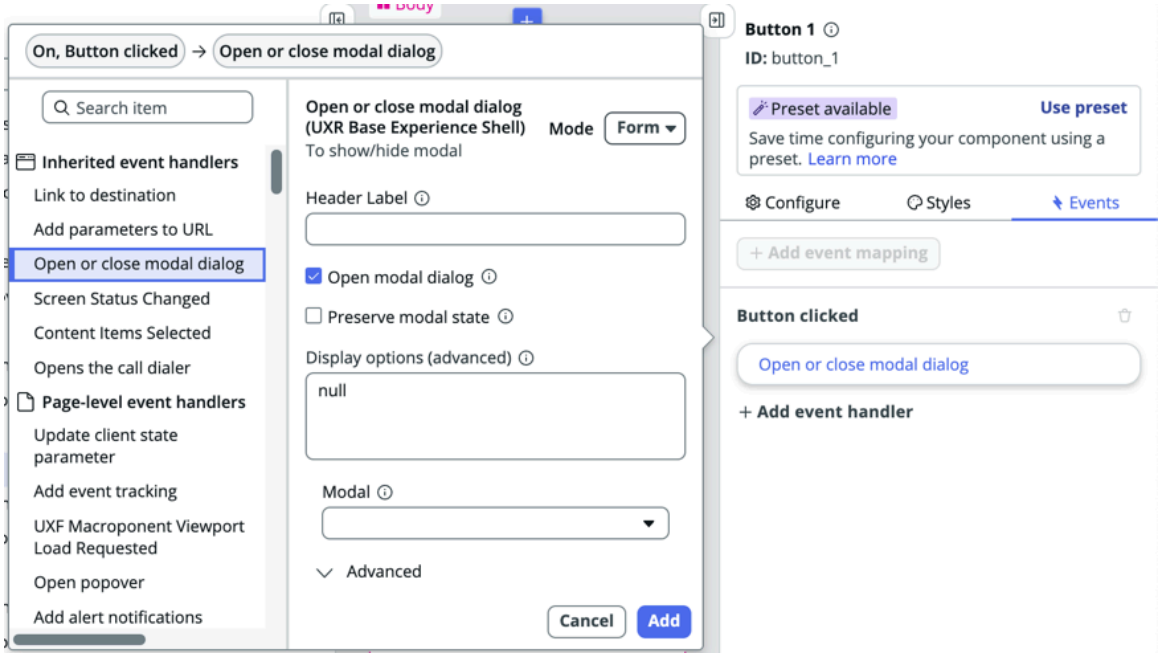
    The following example uses a button to open the viewport modal.



15. Select the **Events** tab.
16. Select **+ Add event handler**.



17. Select **Open or close modal dialog**.
18. Enable **Open modal dialog**.
19. Select the viewport modal that you created in the **Modal** dropdown.



The **Viewport ID** auto populates.

20. Select **Add**.
21. Select **Save**.
22. View and test your page by selecting Preview.

Replace a tab with a viewport-enabled tab

Convert a tab on a page to a viewport-enabled tab. Use viewport-enabled tabs to display third-party custom data, assign audiences, and create variants.

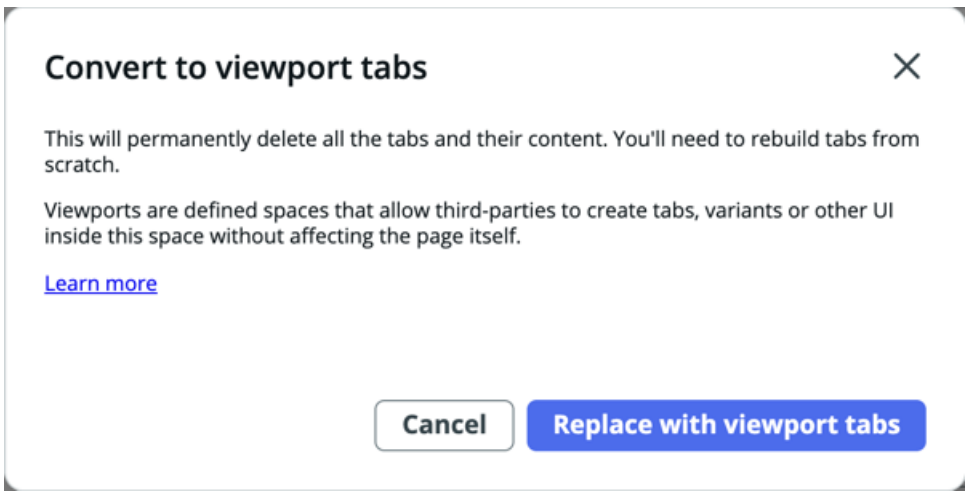
Before you begin

Role required: admin

About this task

Replace your tab or tabs with viewport-enabled tabs.

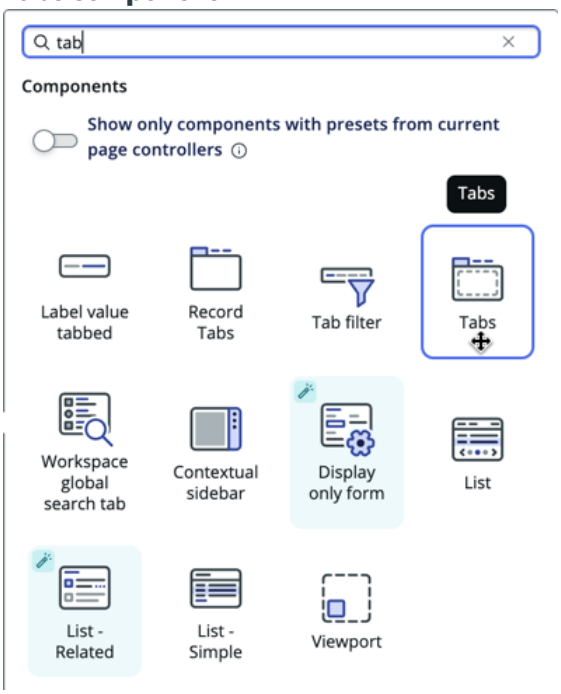
Note:
Replacing a tab with a viewport-enabled tab is a permanent and one-way process. You lose any existing tab content and must recreate the viewport tab content.



Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page with existing tabs, or add a **Tabs** component to your page.

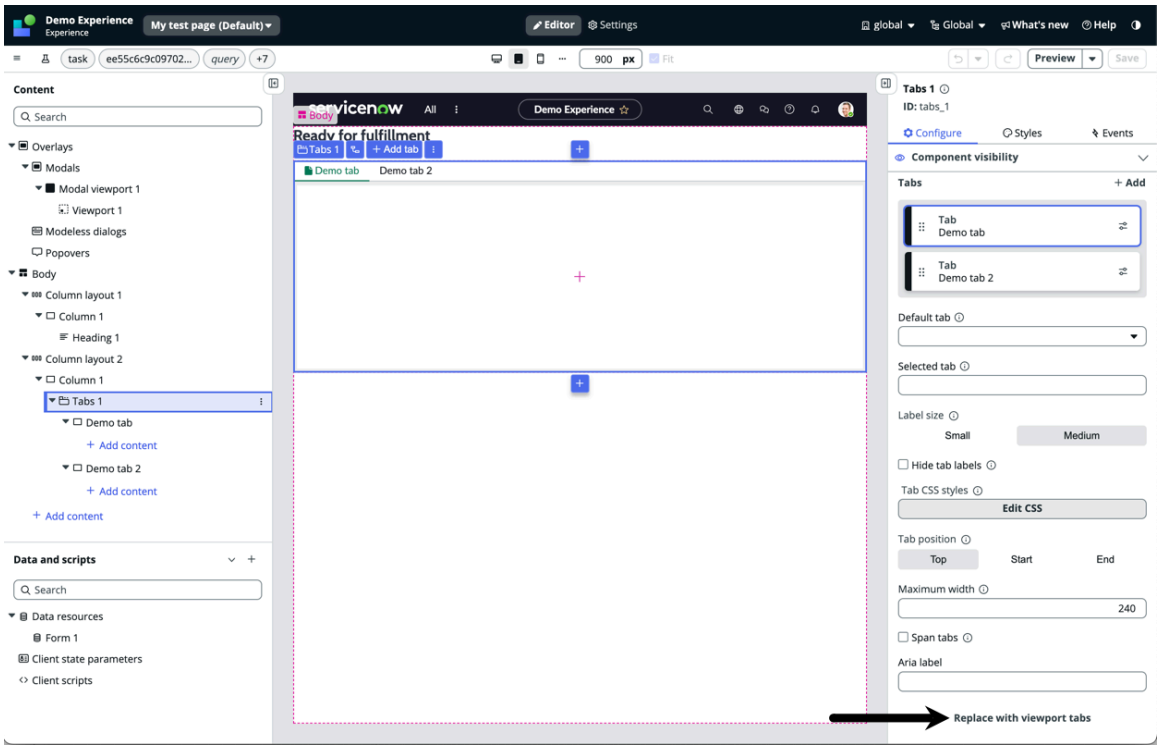
Tabs component



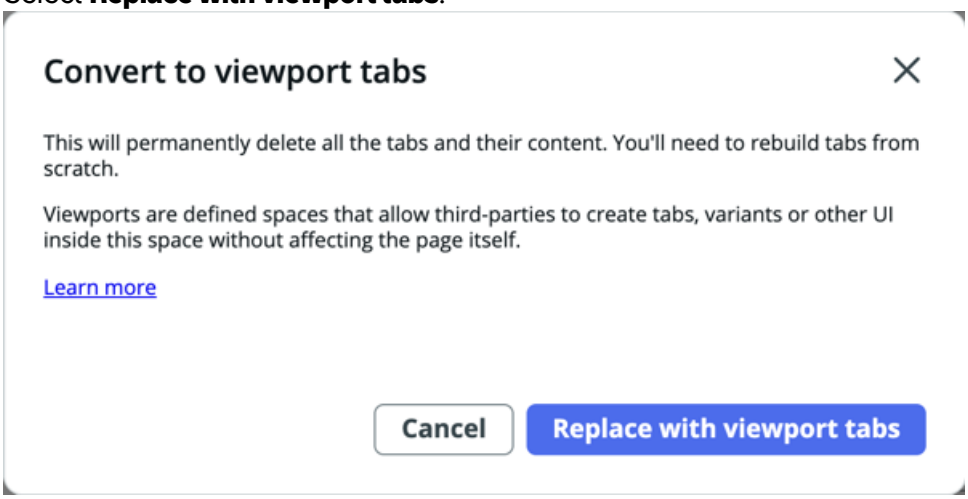
For more information on how to add a Tabs component to a page, see Add tabbed content to UI Builder pages.

4. Select the tabs component that you want to replace with viewport tabs in the content tree.
5. Select **Replace with viewport tabs** in the configuration panel.

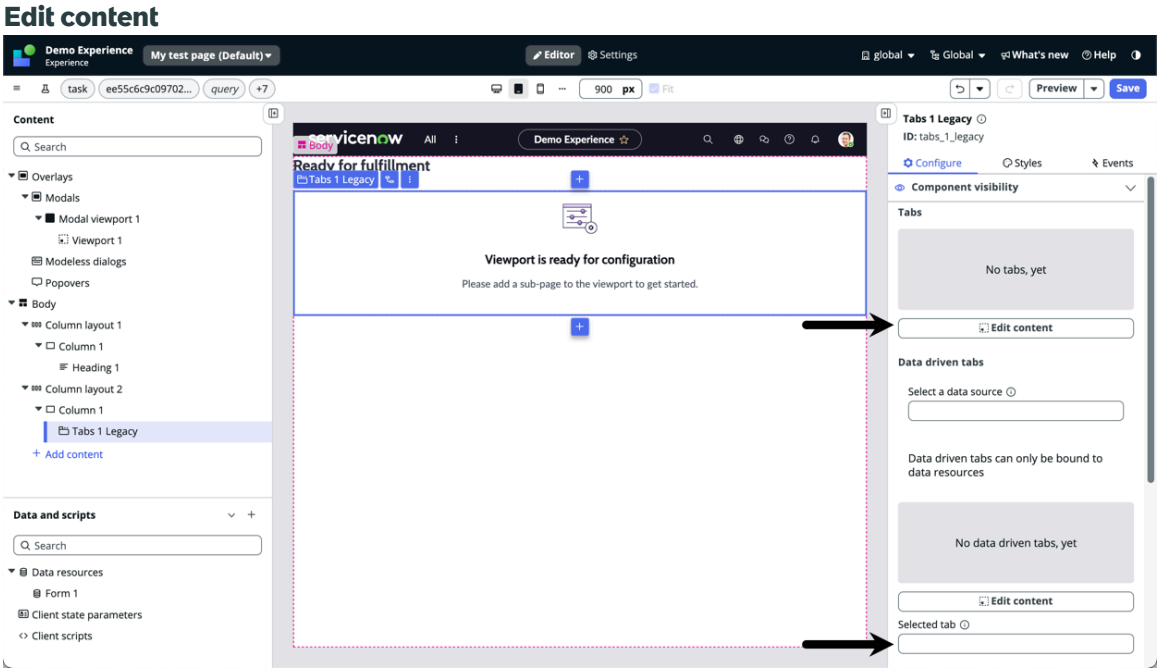
Note:
Replacing a tab with a viewport-enabled tab is a permanent and one-way process. You lose any existing tab content and must recreate the viewport tab content.



6. Select **Replace with viewport tabs**.



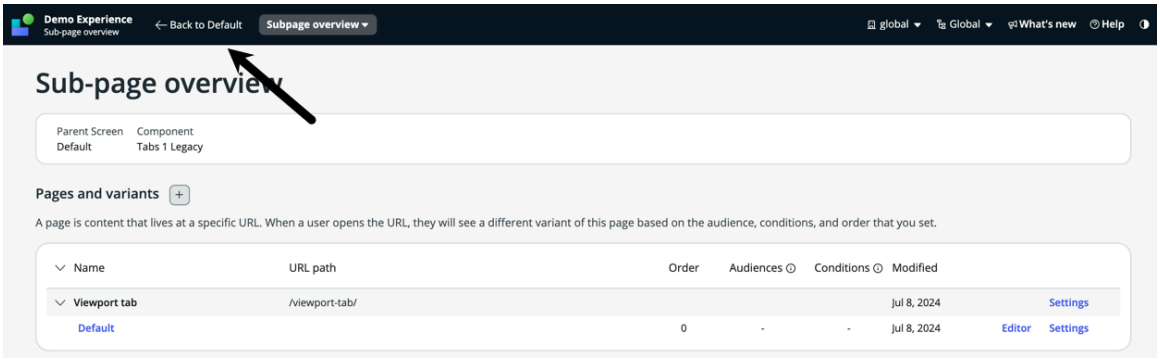
7. Click **Edit content** to add a viewport-enabled tab.



For more information on viewport components, see Add a viewport component to your page.

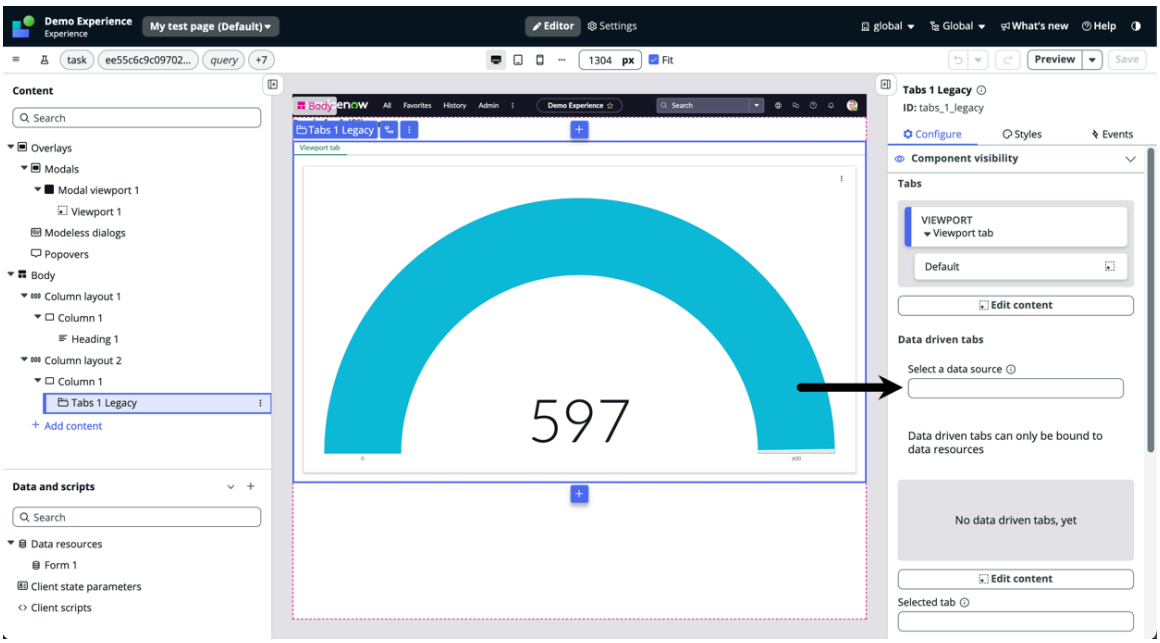
8. Select **Save and continue**.
   The sub-page overview page appears.
9. Select **Create new page.**
10. In the **Name** field, enter a name for your viewport-enabled tab.
11. Keep the default path, or change it to your preference.
12. Select **Continue**.
13. In the **Name** field, enter a name for the page variant or leave it as **Default**.
14. Add one or more audiences for this page.
    If an audience you need is not listed, you can choose the **View all available audiences** to create one.
15. **Optional:** Declare conditions for when to display the page or tab by either using the provided dropdown menus or writing an encoded query string.

    (Optional) For more information on writing encoded queries, see Encoded query strings.
16. Select **Open in editor**.
17. Add components to your tab as you would to a page.
18. Add an event handler to any components to make them perform actions, such as loading page content.
    For more information about event handlers, see Manage actions in UI Builder pages.
19. Select **← Back to (name of page)** to return to your original page.

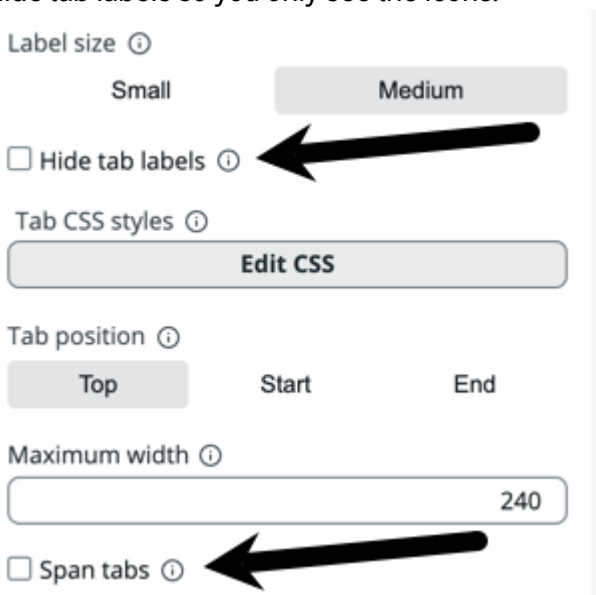


20. In **Data driven tabs**, select a data source to bind data to your viewport-enabled tab using data resources to dynamically expose your data from tables and records.

You then bind these data properties to components in your tab. For more information about using data resources, see Dynamically expose data in UI Builder pages (advanced feature).



21. Choose where the viewport-enabled tab labels appear on the page and decide whether to hide tab labels so you only see the icons.



Create popovers in UI Builder

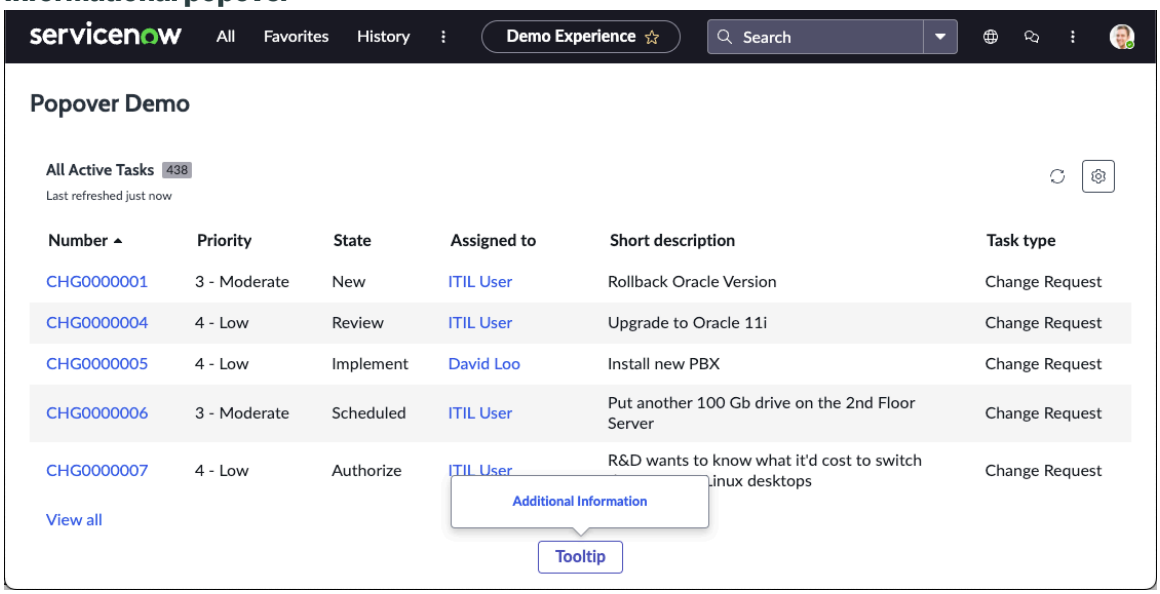
Use popovers on a UI Builder page to overlay contextual information or functionality to help users complete tasks.

A popover is a small window or dialog box that appears above a UI Builder page and contains additional information, options, or actions related to the content or task at hand. Add components to a popover in the same way you add modals to a page.

You can place popovers anywhere on a UI Builder page where you think additional information helps the users. Popovers are intended to provide small pieces of information or links to related content, so you should limit the amount of information or functionality within a popover because the popover only displays when a user is interacting with it.

You can make popovers visible or hidden with event mapping, such as triggering a popover to appear when selecting a button or pointing to a part of the page. For more information, see Define map events.

Informational popover



Add popover to a UI Builder page

Learn how to add a popover in UI Builder. A popover is a container that appears above a page when you click a component. For example, a popover might display contact information when selecting a persons name in a list.

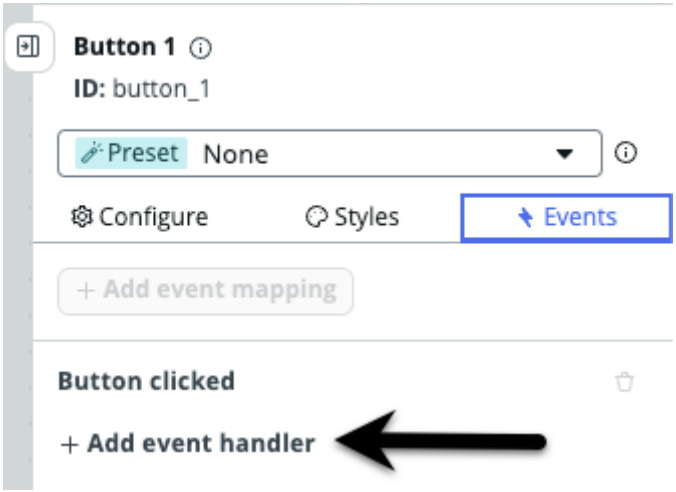
Before you begin

Role required: admin

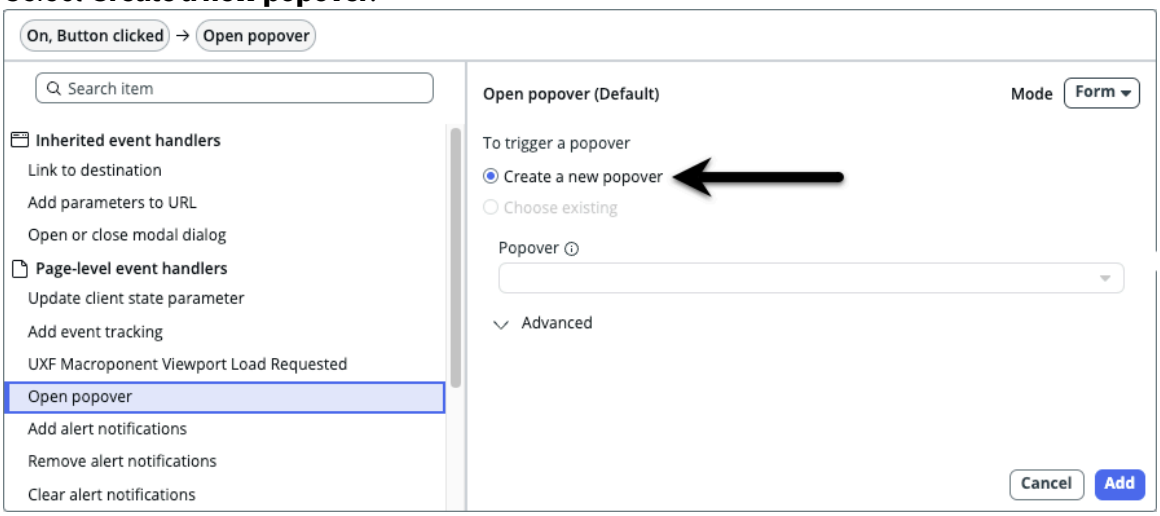
Procedure

1. Navigate to **All** > **Now experience framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Open or create a page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Add a component to your page that you want to trigger a popover, such as a button component.
   See Add and configure components for more information.

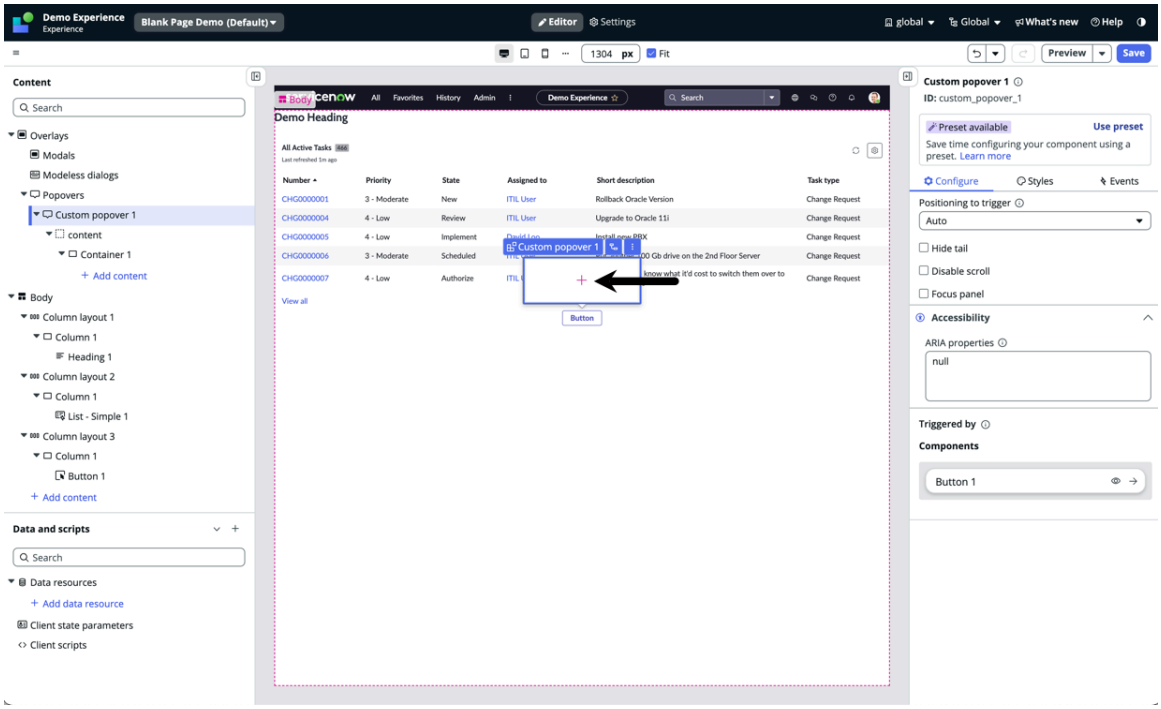
5. Select the **Events** tab in the configuration panel.
6. Select **+ Add event handler**.



7. Select **Open Popover**.
8. Select **Create a new popover**.

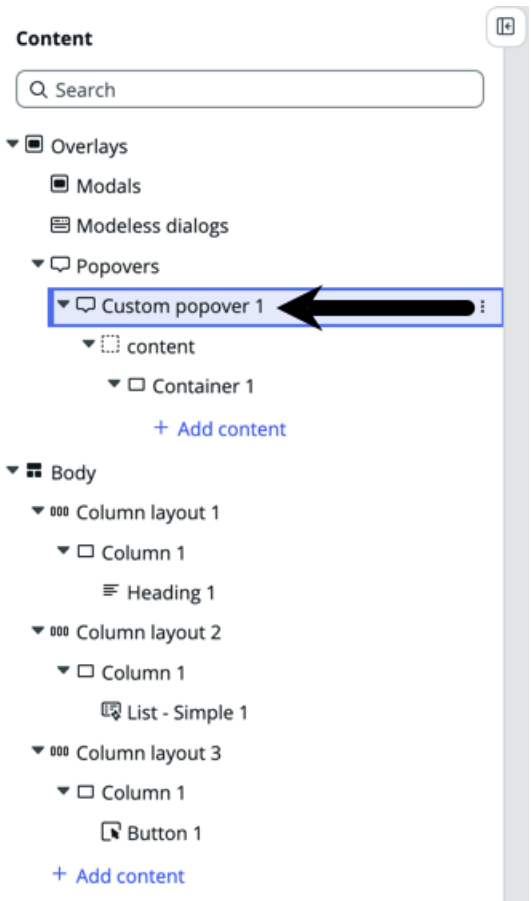


9. Select **Add**.
   The popover appears above the stage.
10. Add components to the popover by selecting the **+** icon.



11. When you finish configuring the popover, close it.

Notice in the content tree that the popovers you create sit above the body of your page structure.



12. Click **Save**.

13. Select **Preview** in the UI Builder header.
14. Click on the button to test the popover.

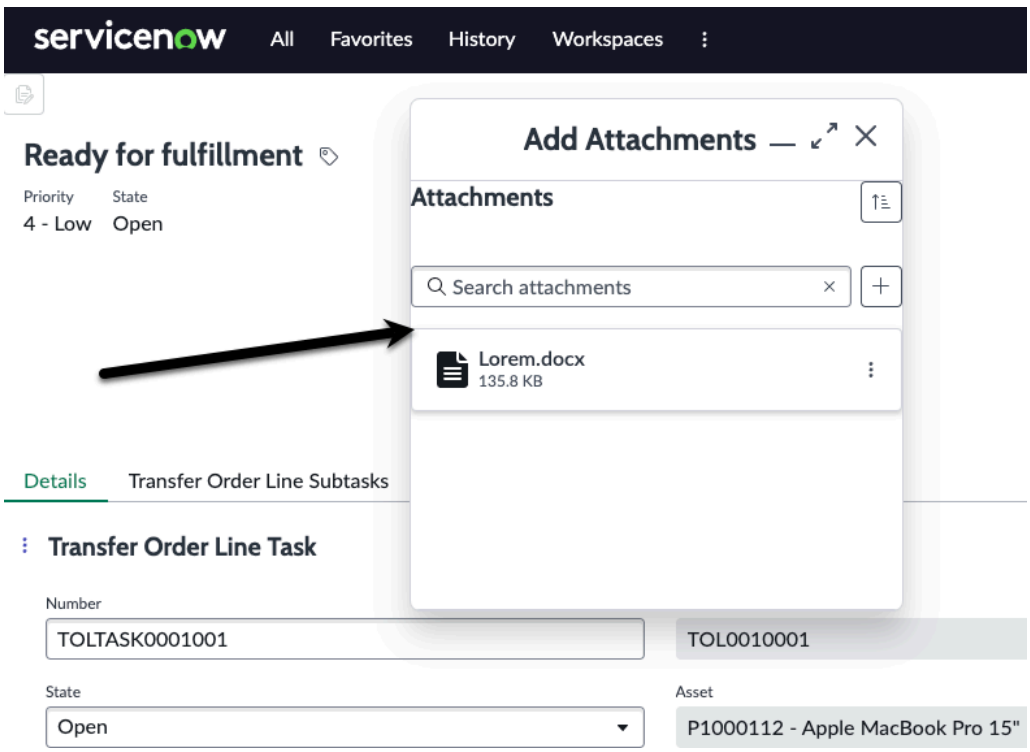
Create modeless dialogs in UI Builder

Use modeless dialogs on a UI Builder page to add a floating window that enables you to interact with both the window content and the page content below.

The Modeless Dialog component is a floating window containing components and content that appears over a UI Builder page. You can do actions in the modeless dialog while still viewing and interacting with the page below. Multiple modeless dialog windows can be added to a single page. Modeless dialogs can be resized, moved to a new position on a page, and minimized.

Modeless dialogs are different than modals, which pop up but block interaction with the page until the modal is closed. Modeless dialogs are also different than popovers, which close after you click outside the popover.

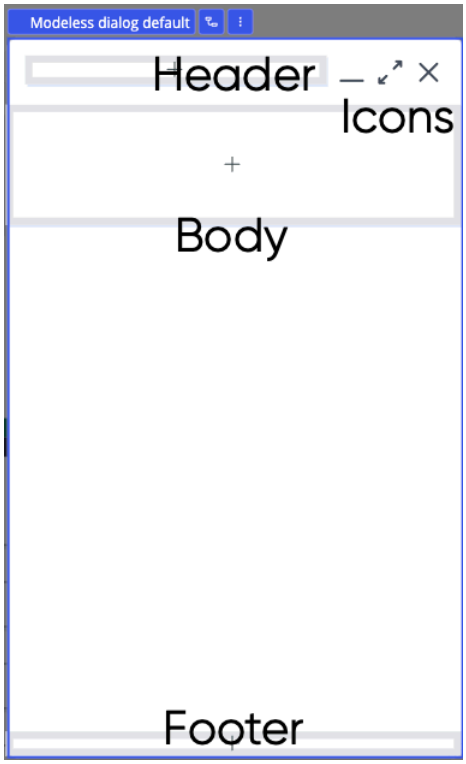
Common modeless dialog use cases include creating a record, composing an email, using chat, adding work notes or comments, reading help information, or adding/viewing attachments in the modeless dialog.



Anatomy of a modeless dialog

A modeless dialog consists of three parts:

- When you add a modeless dialog to a page, the header includes a preconfigured minimize icon, expand icon, and close icon. Other actions can be configured if needed, for example, configure an action in a button to open the modeless dialog in a new tab.
- Add any component from the UI Builder toolbox, for example, Email composer, Attachments, or Agent chat, into the body. You can apply layouts and configure the component as if you were adding the component to a page.
- If needed, provide additional buttons or content in the footer at the bottom of the modeless dialog.



Add modeless dialog to a UI Builder page

Learn how to add a modeless dialog in UI Builder. A modeless dialog is a floating window containing content above a page.

Before you begin

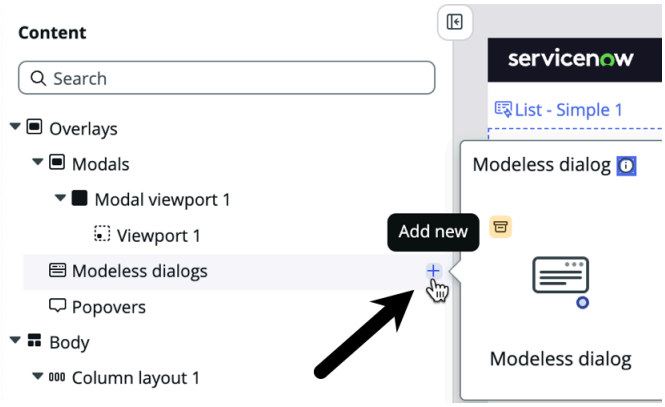
Role required: ui_builder_admin

The following steps walk you through the process of configuring a button to open a modeless dialog containing the Attachments component.

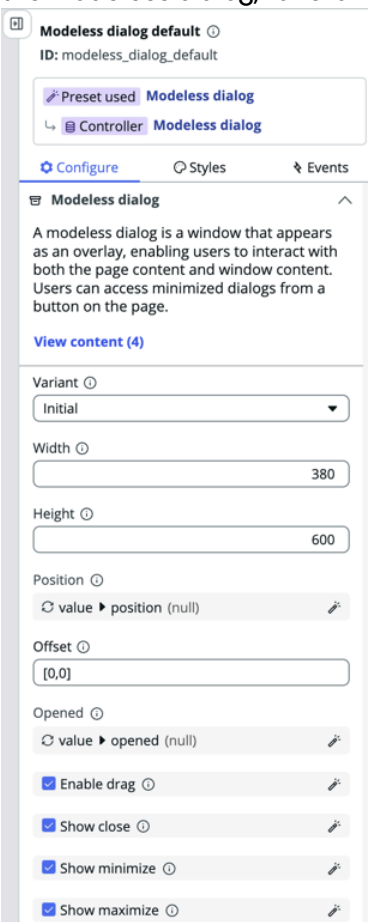
Note:
The procedure outlined here is just one example of how to use modeless dialogs. There are infinite possibilities. Add and configure modeless dialogs to meet your business needs.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Open or create a page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. In the content tree, move your mouse to **Modeless dialogs**, select the + icon, and select the **Modeless dialog** component.



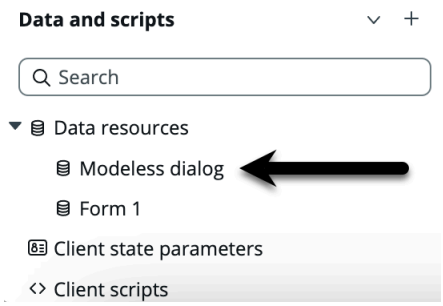
5. If the modeless dialog doesn't appear on the stage automatically, select the new modeless dialog named **Modeless dialog default** in the content tree.
6. In the configuration panel **Configure** tab, view the presets that were added automatically with the modeless dialog, for example, the width, height, position, and enable drag setting.



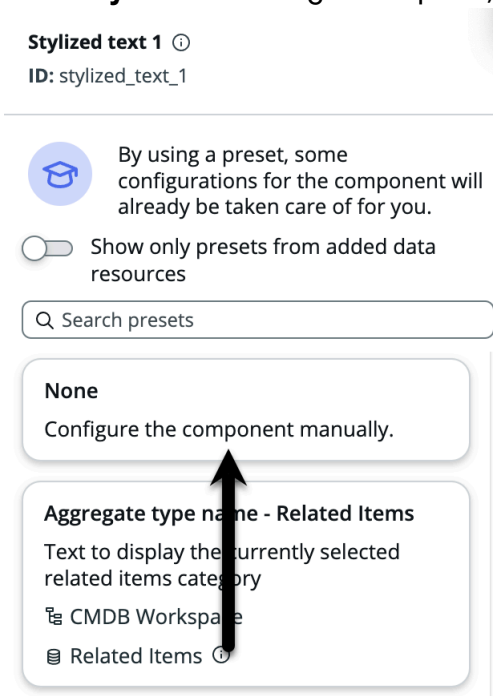
All of these preset properties can be edited, if necessary.

Note:
For more information about Modeless Dialogs advanced properties, see the ServiceNow Developer site.

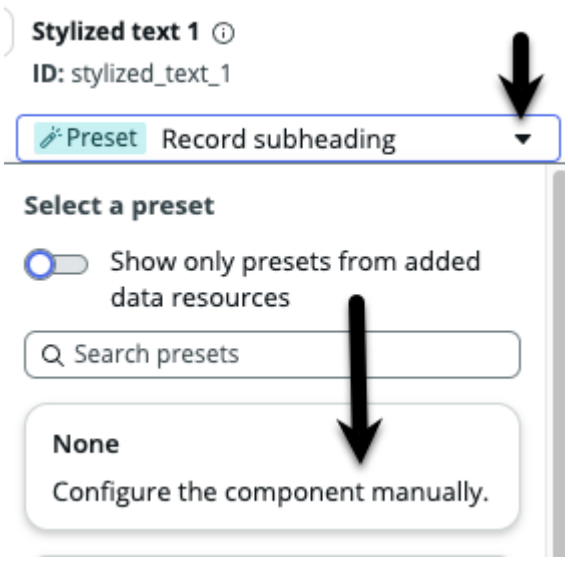
7. In the data resources drawer, the **Modeless dialog** controller was added automatically.



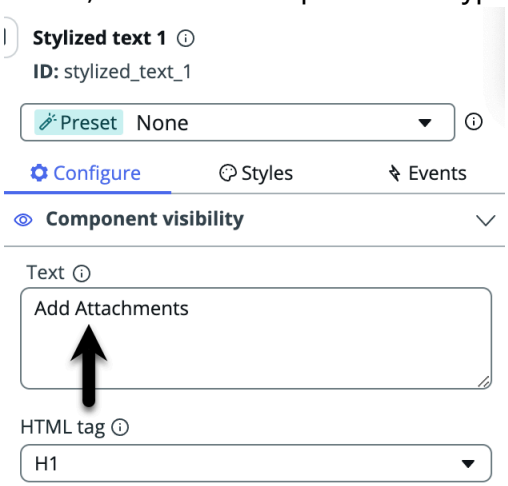
8. In the content tree, select **+ Add content** under **Modeless dialog default > actions** to add content to the modeless dialog header.
9. Search for and select the **Stylized text** component.
10. In the **Stylized text** configuration panel, select **None** to configure the component manually.



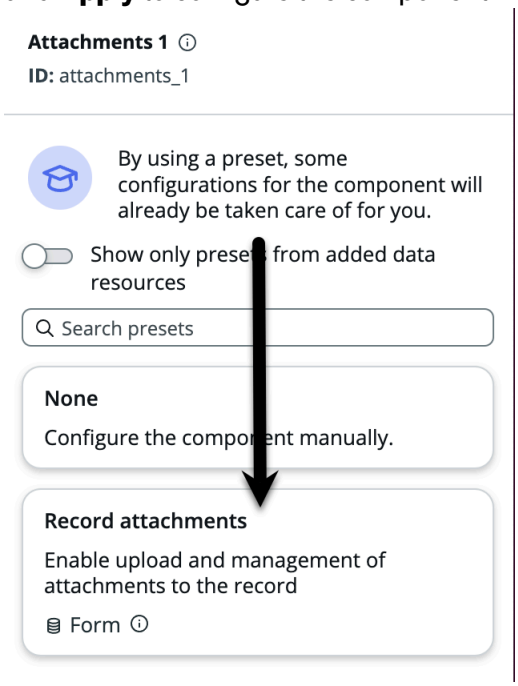
If the **Record subheading** preset was automatically added, select the drop-down arrow, select **None**, and select **Remove** on the **Configure** tab.



11. If not open already, select **Configure** to open the configure tab.
12. In **Text**, remove the sample text and type `Add Attachments`.



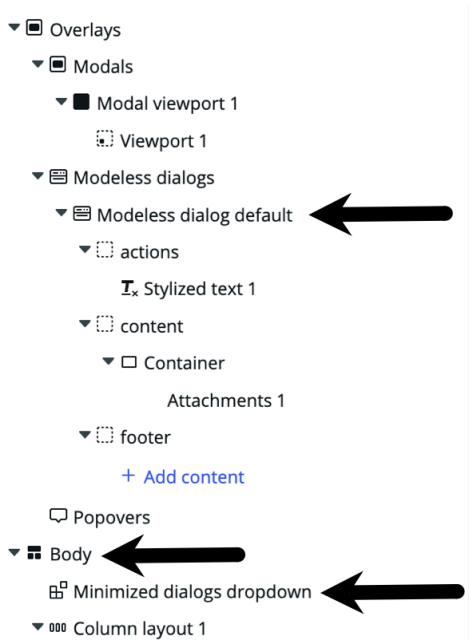
13. Select **Save** in the UI Builder header.
14. In the content tree, select **+ Add content** under **Modeless dialog default > content > Container** to add content to the modeless dialog body.
15. Select the **Attachments** component.
16. In the **Attachments** configuration panel, if not selected already, select **Record attachments** and **Apply** to configure the component with a preset.



For more information about configuring the Attachments component, see the ServiceNow Developer site.

17. You could add content to the modeless dialog footer, but in this example leave the footer empty.
18. Select **Save**.
Notice in the content tree that the modeless dialog and all of its components are listed above the **Body** of your page structure. Also, the **Minimized dialogs dropdown** component is added

to the page automatically (and is listed under the **Body** in the content tree) to provide the functionality for the minimize icon in the modeless dialog header.

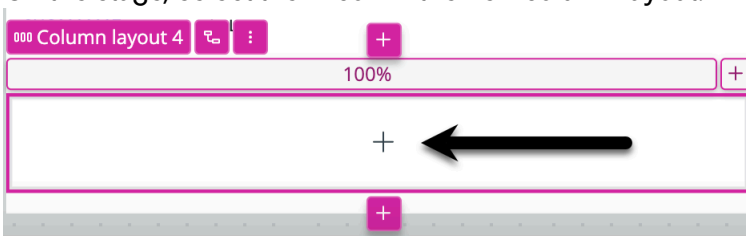


19. In the content tree, select **Body**.
20. Add a button component and configure it to open the modeless dialog.

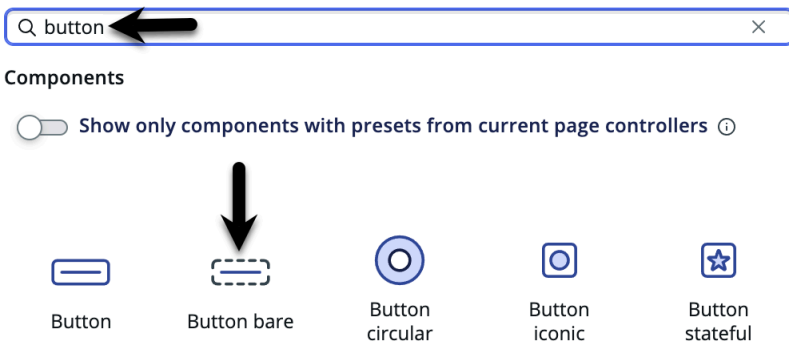
    a. In the content tree, select the **Menu** icon next to **Body** and select **Add content**.

    b. On the **Layouts** tab, select **Single column**.
    A new column layout containing a single column is added to the bottom of the page.

    c. On the stage, select the + icon in the new column layout.



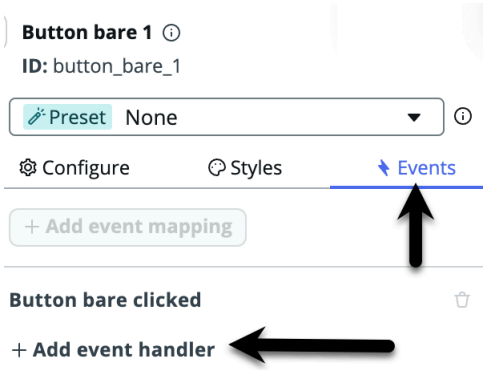
    d. Search for and select the **Button bare** component.



e. In the configuration panel, on the **Configure** tab, select **None** to configure the component manually.

f. Select the **Events** tab in the configuration panel.

g. Select **+ Add event handler**.

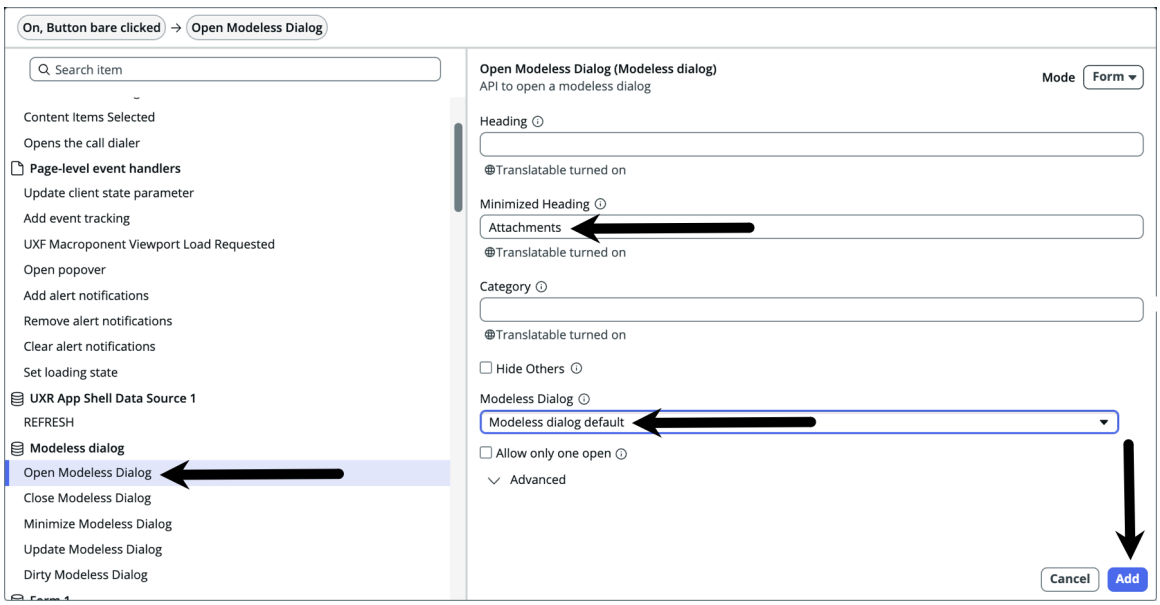


h. In the list at left, select **Open Modeless dialog** (you may need to scroll down in the list).

i. Type `Attachments` in **Minimized Heading**.

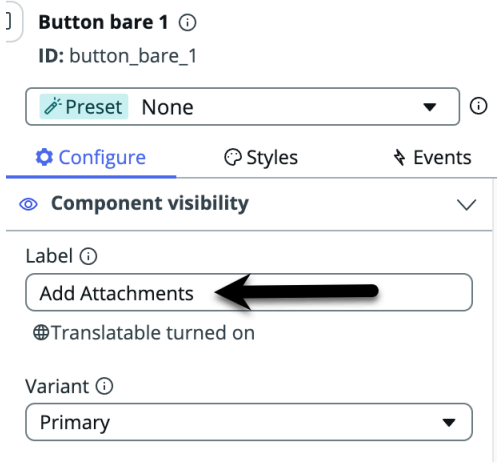
j. Select **Modeless dialog default** in **Modeless dialog**.
There are other options that can be configured here, including a heading and a category.

k. Select **Add**.



l. Select the **Configure** tab.

m. Type `Add Attachments` in **Label**.



21. Select **Save**.
22. Select the drop-down arrow next to **Preview** in the UI Builder header and select **Open URL path** in the list.
23. Select the **Add Attachments** button to test the modeless dialog.
    The modeless dialog opens above the main page with the heading at the top and the Attachments component below.
24. To test the minimize functionality, select the minimize icon on the modeless dialog.

    The modeless dialog is minimized and can be accessed from the Minimized dialogs drop down.
25. Select the minimized dialogs icon and then select **Attachments** to open the modeless dialog window again.

Add modeless dialog event to a UI Builder page

Learn how to add and configure a modeless dialog event, such as open, close, or minimize in UI Builder. A modeless dialog is a floating window containing content above a page.

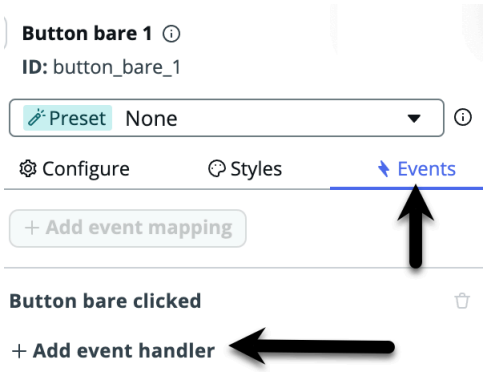
Before you begin

Role required: ui_builder_admin

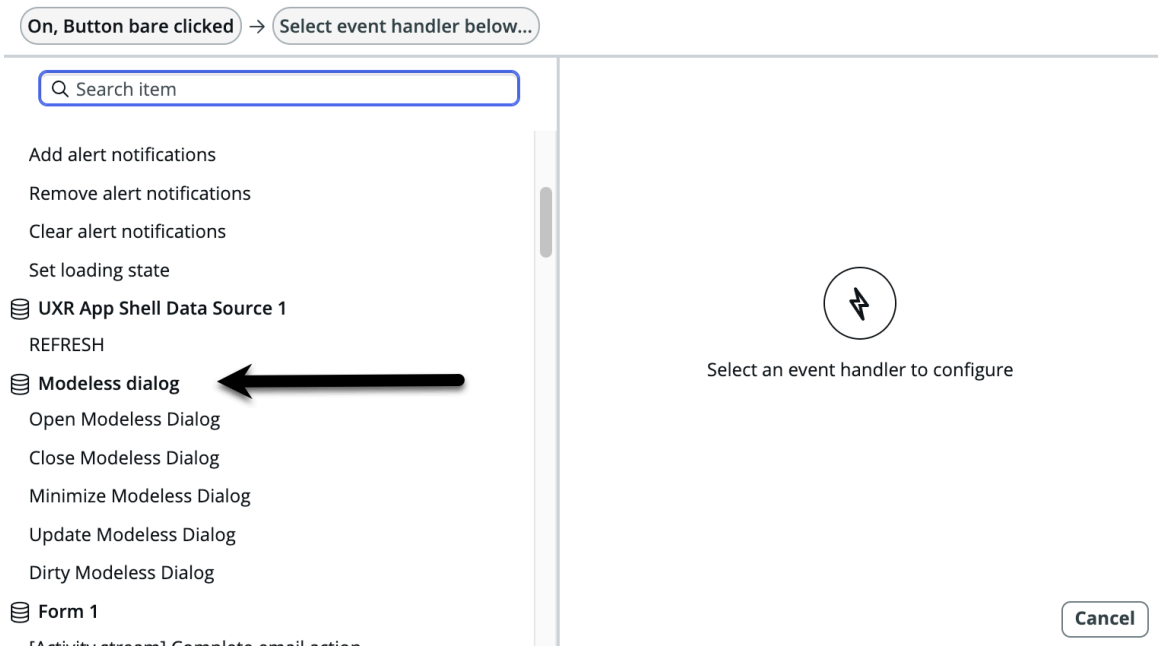
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.

3. Open or create a page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Add a modeless dialog to the page.
   For more information about how to create a modeless dialog, see Add modeless dialog to a UI Builder page.
5. Add a component to the page, such as a button, to take action on the modeless dialog.
   For more information about how to create a button that interacts with a modeless dialog, see Add modeless dialog to a UI Builder page.
6. Select the button component in the content tree or on the stage.
7. In the configuration panel, select the **Events** tab.
8. Select **+ Add event handler**.



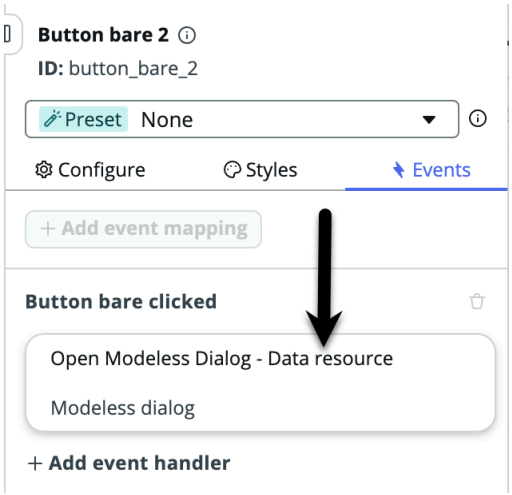
9. From the Event handler preview window, select an action to assign to the button.
   There are five modeless dialog event handlers available.



Note:
For any of the event handlers, select **Script** in the **Mode** drop-down if you prefer to work in the script editor.

10. Select **Add**.

Result
The configured event handler displays in the configuration panel **Events** tab for the component.



Manage actions in UI Builder pages

Learn how to work with events so that you can add actions to components, pages, data resources, and declarative actions in UI Builder.

Actions in UI Builder

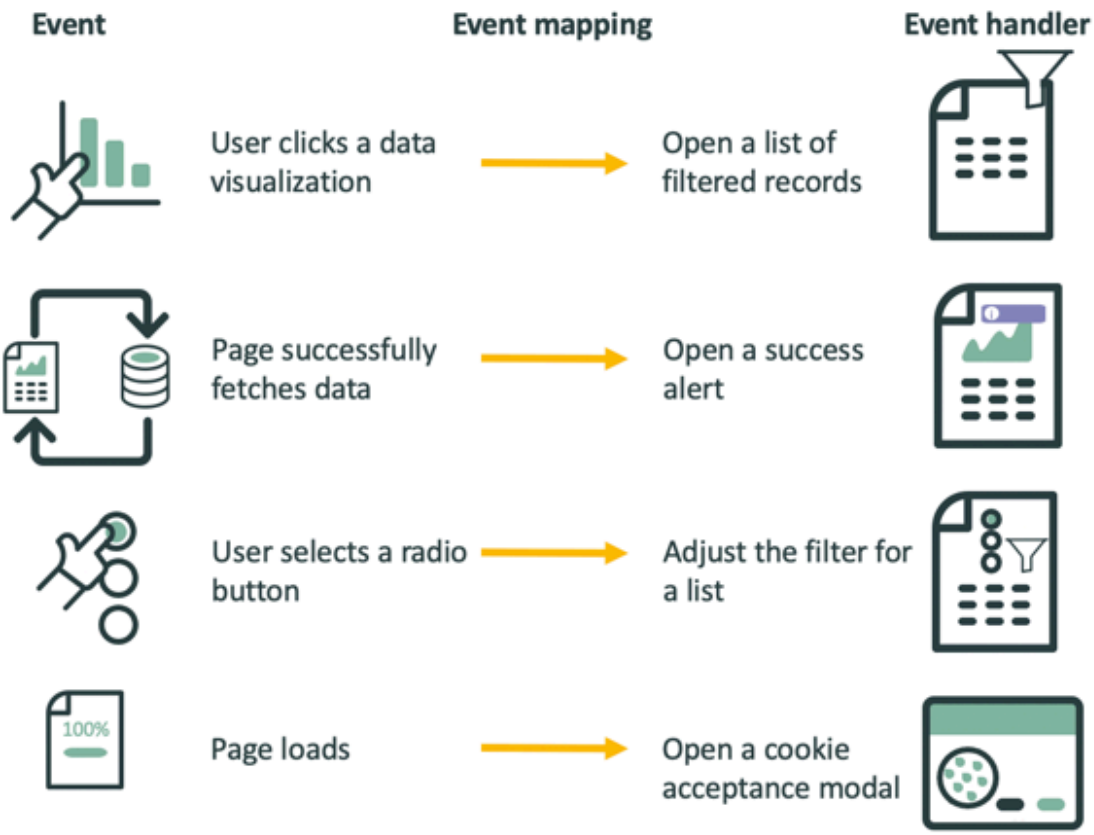
UI actions tell UI Builder what to do when an event is triggered. An event is an action a user takes or occurrence that happens on a page. Use UI actions to create interactive, user-friendly interfaces that help your users complete tasks. Each component has its own events associated with it. Events include:

- User clicks a data visualization
- Page successfully fetches data
- User selects a radio button
- Page loads

Events in UI Builder

Use an event to add actions to your components, pages, and data resources.

- A component event is an action that you set up for a component. You set up an event handler to configure that component action. For example, you can add a button component to your UI Builder page. Then, you can add an event handler to apply an action for that button, such as going to a web page.
- Page events perform actions for the entire page. You can configure the following page events:
  - Page event mappings. Add, remove, or clear alert notifications on your page.
  - Variant event mappings. Add, remove, or clear alert notifications on your page variant.
  - Dispatched events. Create dispatched events for your page to create relayed event mappings that model events after a parent event handler. Select a target parent event handler to model the payload fields after it.
  - Handled events. A handled event is an event that is exposed and available for use by other users. After you create a handled event, it is available under *`Page event mappings`* for other users to use. You can also set up payload fields that you create manually or choose a template for your handled event, such as an open or close a modal dialog.
- Data resources events map data resources to notify information about when data is fetched.
- Events for a page or component do nothing until the event is mapped to one or more event handlers.



Event mapping in UI Builder

Map actions to events such as clicking a button or filling in a field. An event mapping is the process that enables you to map an event's payload or contextual values to the object or handler that acts on that event.

For more information about event mapping, see Define map events.

Event handlers in UI Builder

An event handler is an action performed when an event occurs. By mapping an event handler to an event, you are specifying which action or actions to take when the event occurs. Use an event handler to configure an action for your UI Builder page or the components on the page. For example:

- Clicking a data visualization opens a list of records represented in the visualization
- Fetching data successfully for a list opens an alert that indicates the data fetch was successful
- Selecting a radio button adjusts the filter for a list on the page
- Loading a page opens a modal to confirm acceptance of cookies before proceeding

When you add an event handler to a UI Builder page or component, you can choose different types of event handlers. For example, a Button component can have the following types of event handlers:

- Inherited event handlers. An inherited event handler is exposed from the page that you are working in. If you are in the parent UI Builder page, an inherited event handler could be exposed from the app shell. The following table lists the different types of inherited event handlers that you can use and what you can do with them.

Inherited event handlers

Event handler	Description
Breadcrumb URL changed	
Link to destination	Navigate to a destination. ○ App routes: Link to another page within an app, like a home screen. ○ External URL: Link to a website or any external URL. Sample script `return { route: null, /* Page route, e.g. 'record' */ fields: null, /* Required params, e.g. {"table":"incident","sysId":"X"} */ params: null, /* Optional params, e.g. {"selectedIndex" : 1} */ ##redirect: null, /* ??? True/false? */ ##passiveNavigation: null, /* Load in background, e.g. 'false' */ title: null, multiInstField: null, ##targetRoute: null, /* ??? */ ##external: null /* ??? True/false? */ };`
Add parameters to URL	Add additional parameters to a URL. Sample script `{ “selectedTabIndex” : 0 }`
Open or close modal dialog	After you create a modal, use *Open or close modal dialog* to trigger the modal.

- Page-level event handlers. This type of event handler is common to all pages, and you would use this handler type when you want to add or clear page-level alert notifications. The following table lists the different types of page-level event handlers that you can use and what you can do with them.

Page-level event handlers

Event handler	Description
Add alert notifications	Add a code snippet to send an alert notification. For example: `return { items: [{"type" : "info", "message" : "Info message", "id" : "optionalID"}] /* Types: info, warning, error */ };`
Remove alert notification	Add code to call alert notification IDs that you want to dismiss. For example, click a button to remove a page load alert notification.
Clear alert notification	Add code to call all alert notification IDs that you want to dismiss. For example, click a button to remove all alert notifications.
Set loading state	Toggle loading on or off. For example, you can toggle loading on to load the page when you click a button or toggle loading off to not load the page when a button is clicked.
Update client state parameter	Declaratively set the client state parameter. Let's say that you have a client state parameter that you set up with a value. You can configure the *`Update client state parameter`* event handler to update the client state parameter with a new value. For example, you have a client state parameter that is called **Greeting** that is set up with `Hello` as the initial value. You can add an *`Update client state parameter`* event handler, select the **Greeting** client state parameter, and then enter a new value like `Goodbye`. When you click the button, **Goodbye** replaces **Hello** on the page.
UXF macroponent viewport load requested	Add to a component, such as a button component, to open a viewport. For more information, see Add a viewport component to your page.

- Data resource handlers. This type of event handler triggers the fetching or writing of data to the server.

  You can refresh the app shell data source data on your UI Builder page by clicking a button. For example, with a data resource handler, you can do the following actions:

  - Bind data to the description of an incident record.
  - Change the value of the incident description.

- Add a button component to your page.
- Label the button as `Refresh incident`.
- Add a *`Look Up Record`* event handler for the button.
- Save your page.
- If the description of the incident record changes, click **Refresh** to update the description on your page.

- Client scripts. Scripts that execute when an event is triggered on a component. You create these scripts in the Client scripts section in UI Builder. For more information, see Define and bind client scripts to components.

Binding events to components in UI Builder

Bind event handlers to the events on a component. When you add components to your UI Builder page, these components are not configured to perform any action on your page. For example, a button component is static and does not do anything until you bind an event action to it, such as deleting a record. Some components have several events where event handlers can be assigned. For example, on the list component, you can assign a navigation handler to the *`Row clicked`* event. You can also assign an open modal to the *`Data fetch failed`* event. For more information, see Bind an event to a component.

Binding events to UI Builder pages

Bind a page-level event to execute event handlers on the page. For example, use page-level events for page notifications, page load, or when a page property changes. The assignment of the event handler to the page-level event is similar to assigning handlers to events from components.

You can bind event mappings using the following methods:

- Page event mappings. Add, remove, or clear alert notifications on your page.
- Variant event mappings. Add, remove, or clear alert notifications on your page variant.
- Dispatched events. Create dispatched events for your page to create relayed event mappings that model events after a parent event handler. Select a target parent event handler to model the payload fields after it.
- Handled events. Add a handled event for an event that is exposed and available for use by other users.

For more information, see Bind an event to a page.

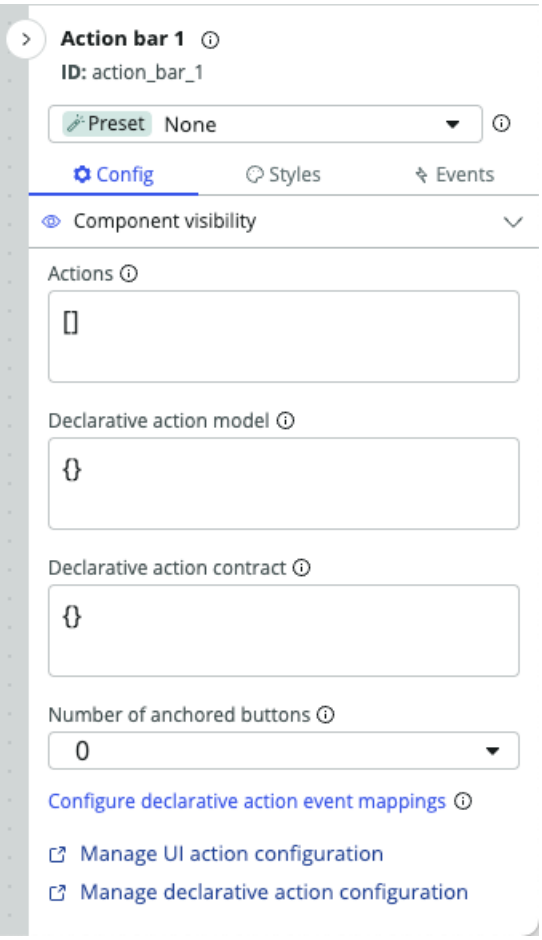
Binding events to data resources in UI Builder

Bind event handlers to individual data resources on your UI Builder page. For example, when a data resource successfully fetches new data, it executes an event handler, like navigation, to take a user to the next step in a flow. When a data resource successfully adds a record to a table, it shows a success modal that uses the show modal event handler. For more information, see Bind an event to a data resource.

Binding events to declarative actions in UI Builder

Bind data elements to add event actions to a declarative action definition in **Actions & Components** in the ServiceNow AI Platform®. For example, you could bind a data element to add an event action to complete work on a table.

If you add a component to your UI Builder page that has a declarative action, you must bind it to a handled event. The handled event creates an action that is performed when a user selects the component. By selecting **Configure declarative action event mapping**, you add a new event handler to define what the declarative action does on the page.



For more information, see Bind an event to a declarative action.

Define map events

An event mapping in UI Builder is the process that enables you to map an event's payload or contextual values to the object or handler that acts on that event. The four event types are: component, page, data resource, and declarative action.

Event mapping is an important process within UI Builder. When you build pages with components, you need those components to perform actions for users. For example, if you add a button component to the page, a button-clicked event must be mapped to an event handler. The event handler performs a button-clicked action when it is selected by a user. An example is when you add a data resource, such as a form, and have an event handler notify the user when the form successfully loads.

Event types

The event types that are available are based on the component. For example, declarative action events are available for specific components, such as the Action bar and List components.

You choose a type of event based on what action you want to perform on your page. For example, if you want to bind an action to a component, such as a button loading a web page, you would use a component event. If you want an event to apply to your whole page, such as adding

an alert notification to a page, you use a page event. The following table describes each event type that is available in UI Builder and provides some examples on how you can use the events.

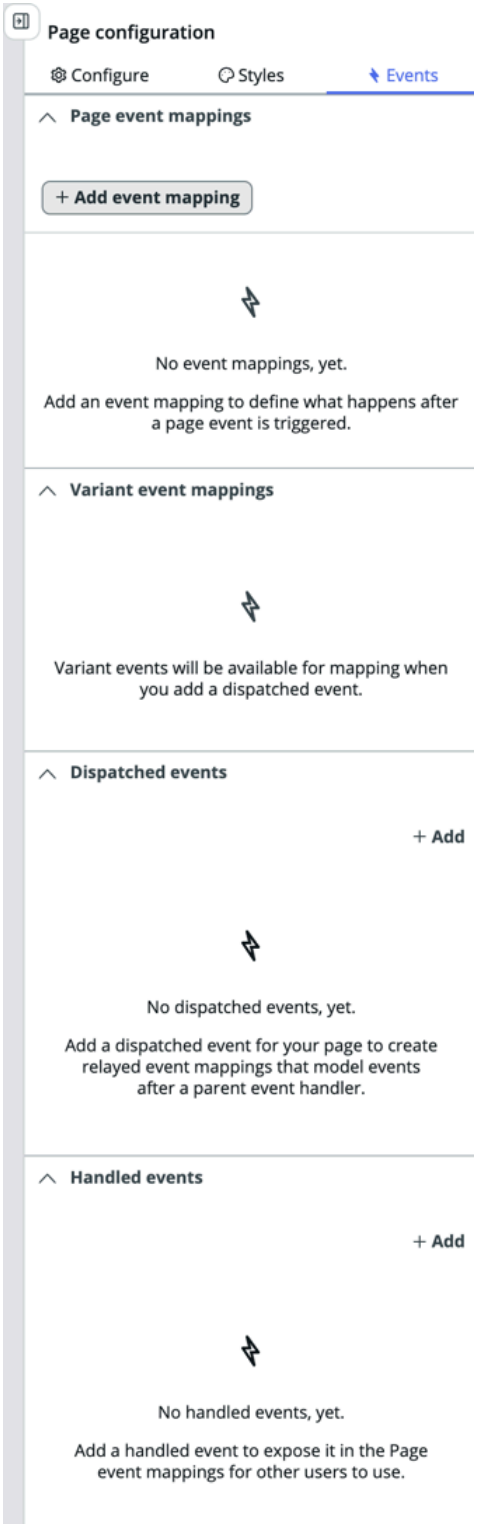
Event types and descriptions

Event type	Description
Component events	Action that you set up for a component. You set up an event handler to configure that component action. For example, add an event handler to apply an action for a button, such as going to a web page. For more information on binding events to components, see Bind an event to a component.
Page events	Page event that performs an action for the entire page. You can configure the following page events: • Page event mappings. ○ These event mappings are saved on the page definition record, which can be found in the [sys_ux_macroponent] table. ○ The source events for these event mappings are Page ready and Page property. They are defined for your page. ○ The available handlers for these event mappings are: ▪ User session events ▪ Page-level events ▪ UXR App Shell Data Source ▪ Client scripts that are defined on the page ▪ Operations from local and inherited data resources • Variant event mappings. ○ These event mappings are saved on the variant record, which you can find in the [sys_ux_screen] table. ○ The source events for these event mappings are any dispatched events that are defined for your page. ○ The available handlers for these event mappings are canvas-level events, the UXR App Shell Data Source, and operations from the inherited data resources. ○ Variant event mappings are used as relays so that the events that are dispatched from components on your page can be relayed up to user session event handlers.

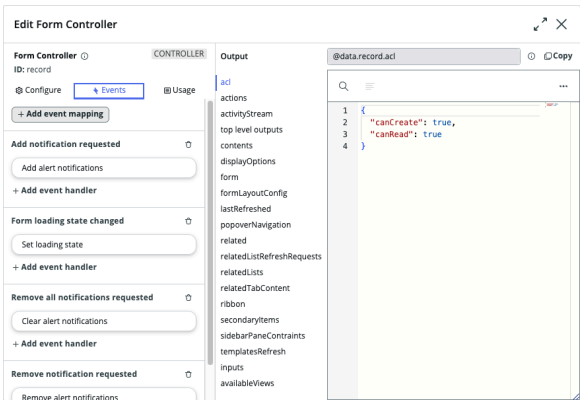
Event types and descriptions (continued)

Event type	Description
	○ Mappings are created automatically on a page save when you have mapped a component's dispatched event to a user session handler. For example, by mapping a **Button clicked** event to the canvas-level **Link to destination** event, the event creates a dispatched relay event and a variant event mapping when the page is saved. • Dispatched events. ○ These events are saved in the [sys_ux_event] table. The page definition record contains references to the [sys_ux_event] record. ○ These events serve as source events for variant event mappings. ○ You can select **+Add** to create new dispatched events for your page. Configure the event label to auto-populate the event name and payload fields for your new `sys_ux_event`. ○ These events are used to create relay event mappings. You can select a **Target parent event handler** to model the payload fields after the selected event. You can also manually create payload fields for your dispatched event. • Handled events. ○ These events are saved in the [sys_ux_event] table. The page definition record contains references to the [sys_ux_event] record. ○ These events serve as source events for page event mappings. ○ You can select **+Add** to create new dispatched events for your page. Handled events are not modeled after parent event handlers. Payload fields for handled events are created manually. To use an

Event types and descriptions (continued)

Event type	Description
	existing handler's payload fields as a template, select a template and edit the fields as necessary.  For more information on binding an event to a page, see Bind an event to a page.

Event types and descriptions (continued)

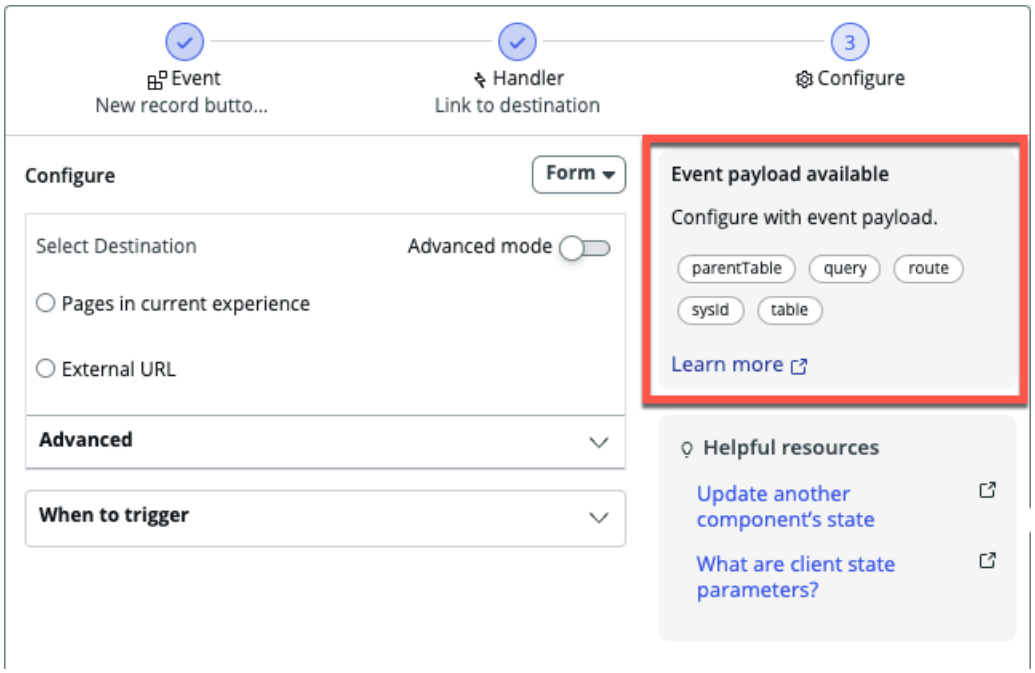
Event type	Description
Data resource events	Events that are mapped to data resources to provide notifications about when data is fetched. • Data Fetch Initiated. When a data resource event is triggered, the event handler executes the data fetch process. • Data Fetch Succeeded. When a data resource event is triggered, the event handler executes the process to notify a user when the data fetch completed successfully. • Data Fetch Failed. When a data resource event is triggered, the event handler executes the process to notify a user if the data fetch was unsuccessful.  For more information on binding an event to a data resource, see Bind an event to a data resource.
Declarative action events	Bind data elements within UI Builder to add event actions to a declarative action. You configure a declarative action event mapping in the ServiceNow AI Platform® declarative action assignment table. For an example, navigate to **Filter navigator** > **sys_declarative_action_assignment.list** and then search for and open an existing declarative action. In UI Builder, you bind an event to the declarative action. For more information on how to use declarative action events, see Bind an event to a declarative action.

Event payloads in UI Builder

Use event payloads to link additional data to an action.

Event payloads are pieces of data sent by a component when a selected event is triggered. The data sent by an event can include the type of event, timestamps, user actions, or resource data such as a SysID.

You can use this payload data when configuring an event handler so that the resulting interaction can be linked to the emitted data. For example, a SysID can be passed to other components on a page to display information related to a specified record.



Each component and event has a unique set of payload options. Event payloads may not be properly defined for each component. If that is the case, define a client script so that the console logs payloads such as `console.log(event.payload)`.

Configure an event handler

Add an event handler to a page, component, data resource, or declarative action within UI Builder so that your user can trigger an action.

Before you begin

Role required: ui_builder_admin

About this task

An event handler lets you configure an action, components, data resource, or declarative action on your page. For example, you can map an event to your page to add an alert notification when the page successfully loads or you can add an event handler for a button component to perform an action when a user clicks it. The event handler could also be a modal on your page that asks a user to verify that the user wants to delete the record. For more information, see Manage actions in UI Builder pages.

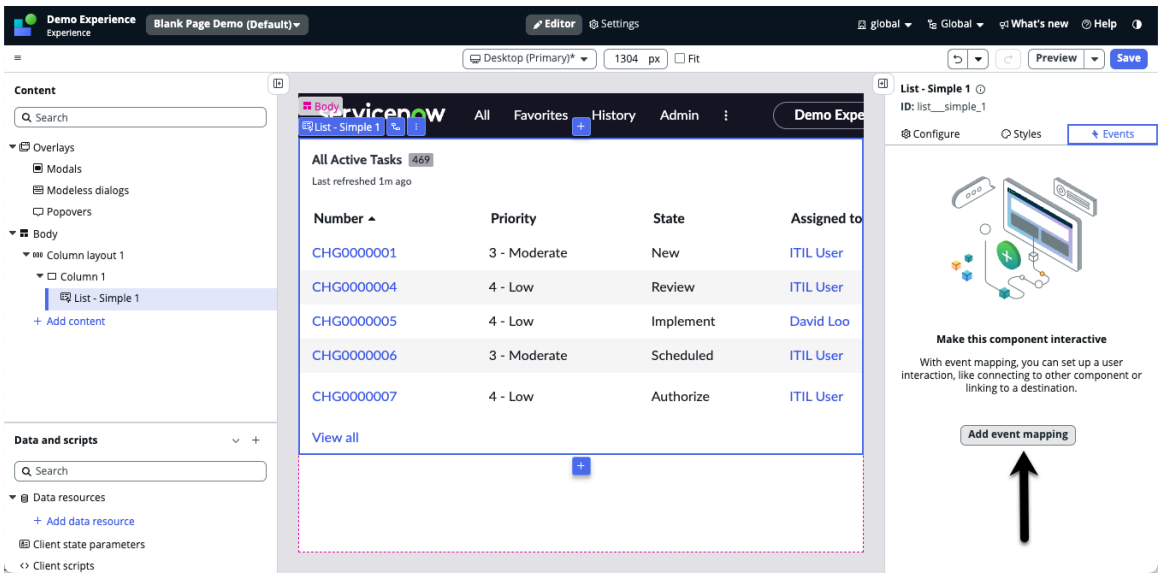
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Open or create a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Add a component to your page, such as a button.

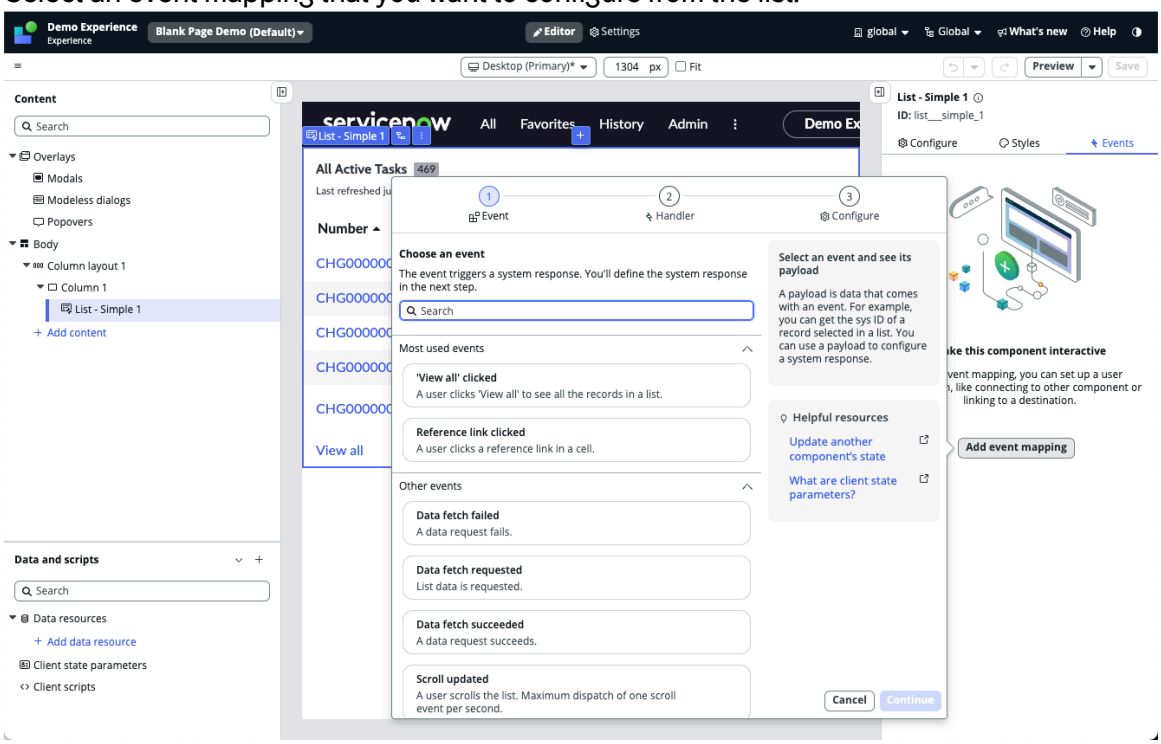
For more information about adding components to a page, see Add and configure components.

5. To add an event handler to your component's event, go to the configuration panel and select **Events**.
   An event handler lets you assign an event to a component. For example, if you add a button component to your page, you want it to perform an action when a user clicks it.
6. To start the process of assigning an event to a component select **Add event mapping**.

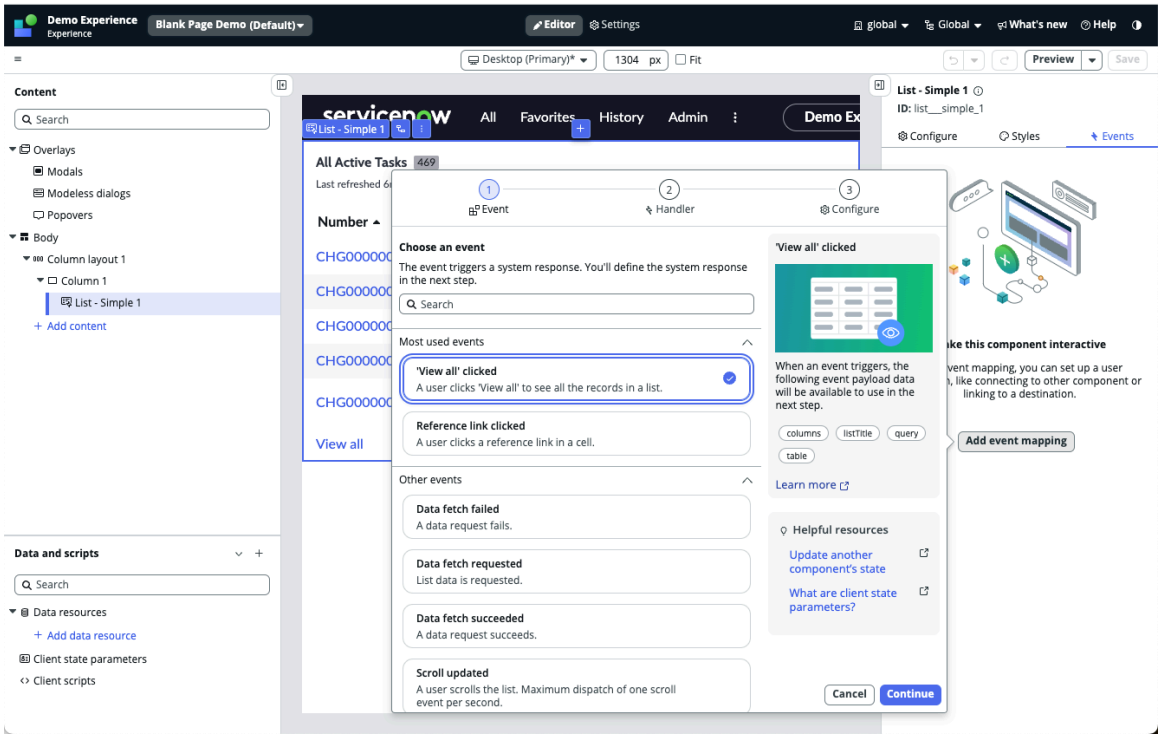
   Some components, such as the button component, only have one event mapping. Other components can have many events.



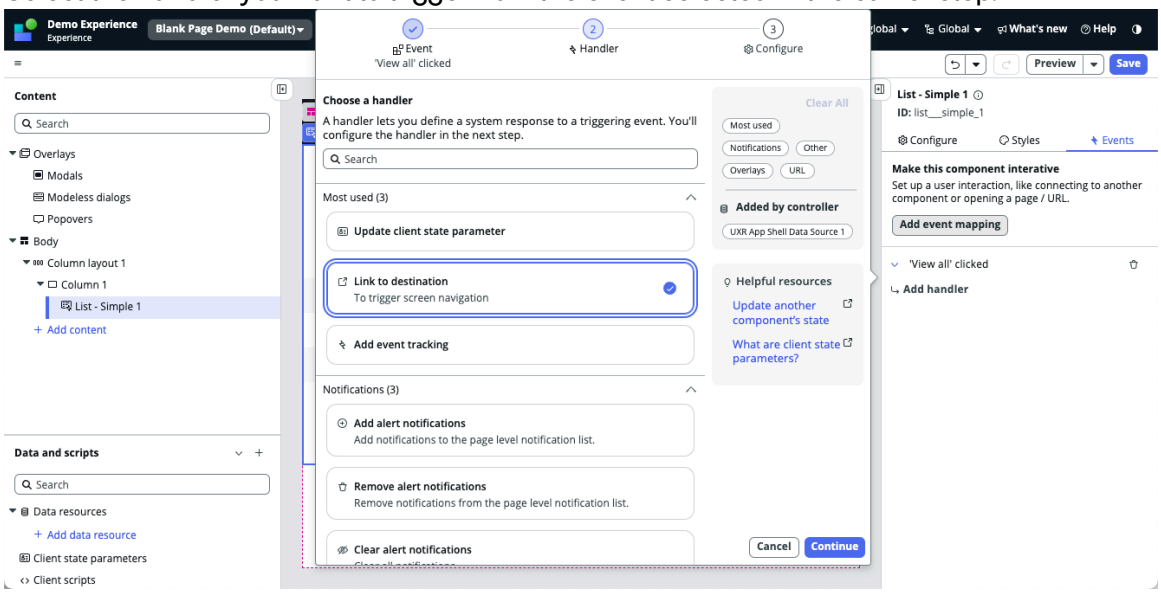
7. Select an event mapping that you want to configure from the list.



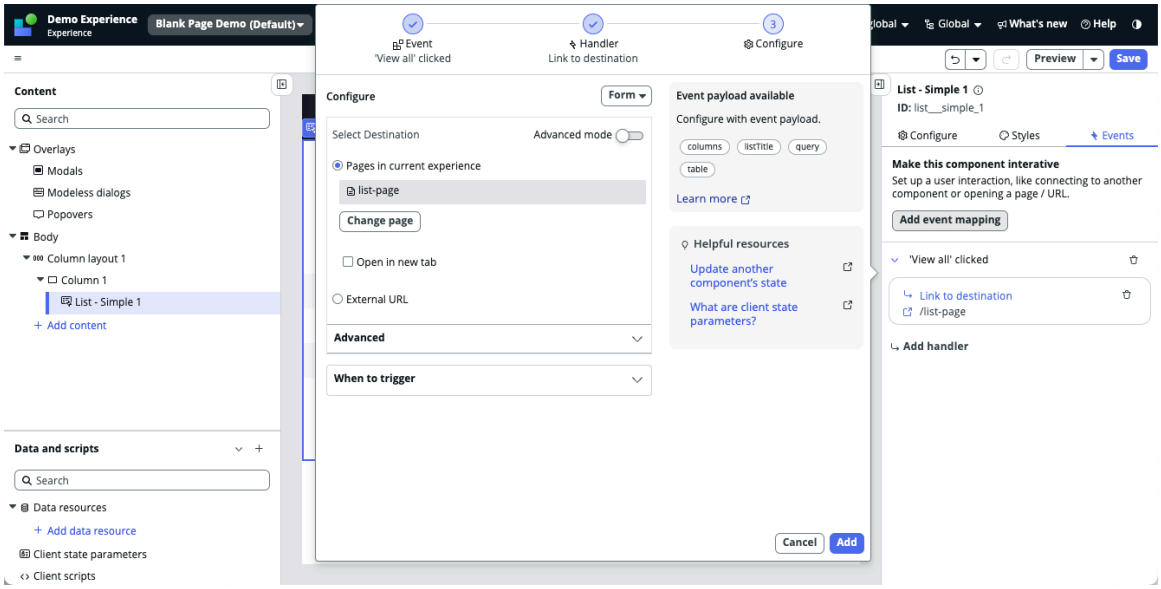
8. From the Event handler preview screen, select an action that you want assigned to the component.



9. Select **Continue**.
10. Select the handler you want to trigger from the event selected in the earlier step.



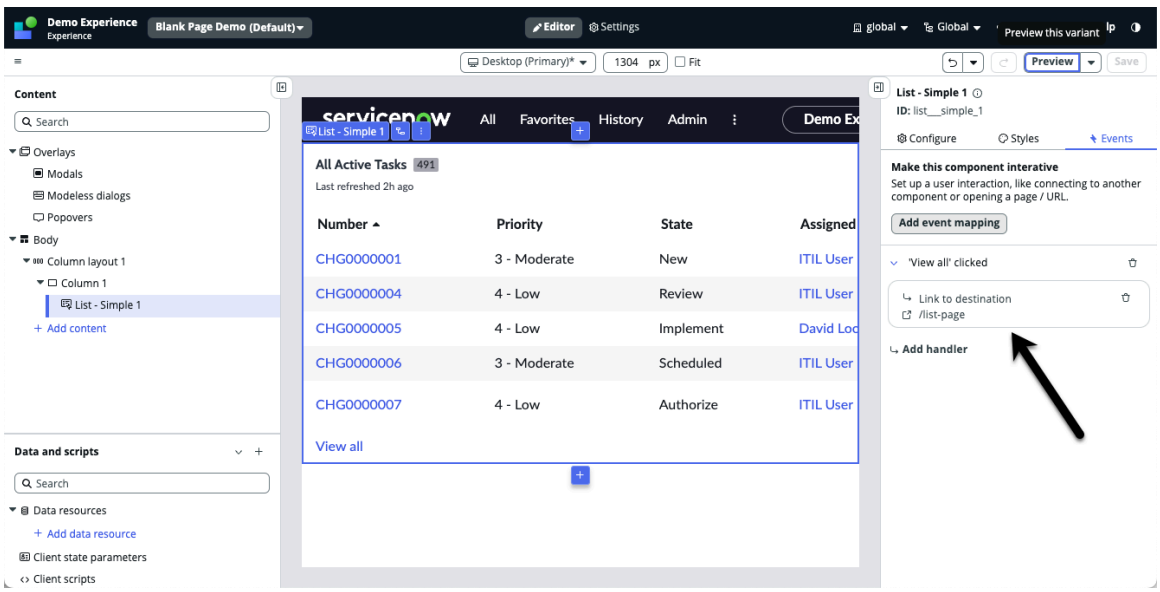
11. Select **Continue**.
12. Configure the payload for the event.



13. Select **Add**.
14. Select **Save**.
15. Test the event by selecting **Preview** in the header and trigger the action.

Result

The configured event handler displays in the events tab of the configuration panel.



Bind events to add actions

After you have created events, you can bind them to actions on your components, pages, data resources, and declarative actions.

Bind an event to a component

Bind data elements within UI Builder so that you can add event actions to your components.

Before you begin

Role required: ui_builder_admin

About this task

Each component has specific events that it binds to. For example, a button component has only a button-clicked event, while other components can have multiple events that are associated with them.

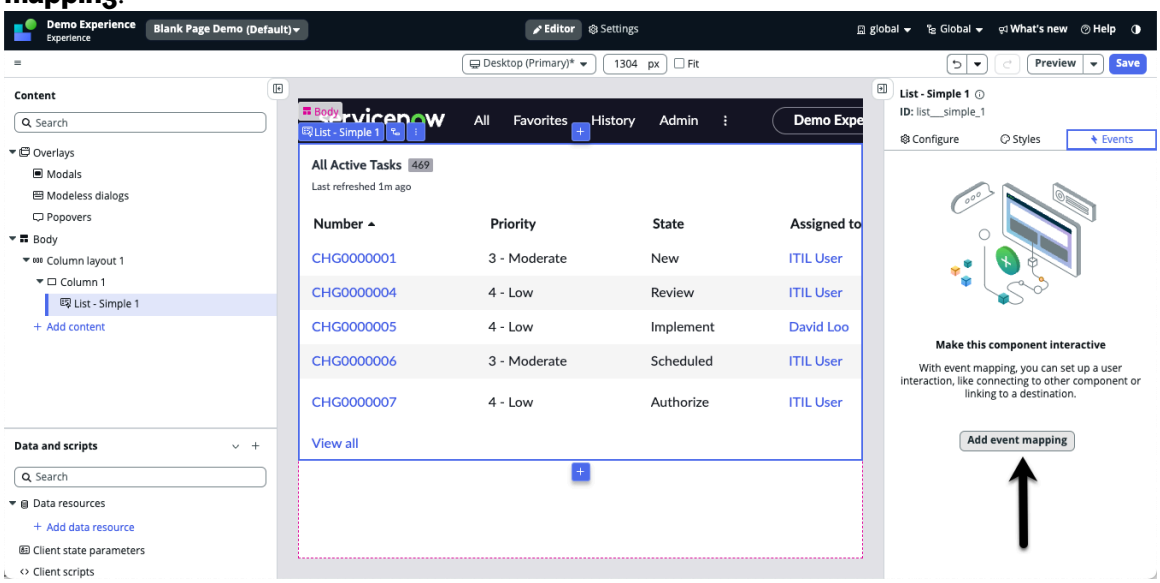
Some components do not have an event action that is applied to them. An example is the heading component. Many components require that you map an event to your component to make it perform an action, such as loading data.

To add actions to components, pages, and data resources on your page, you can add an event handler. A button component that you add to a page is static. By binding an event action to the button, you can link it to a web page.

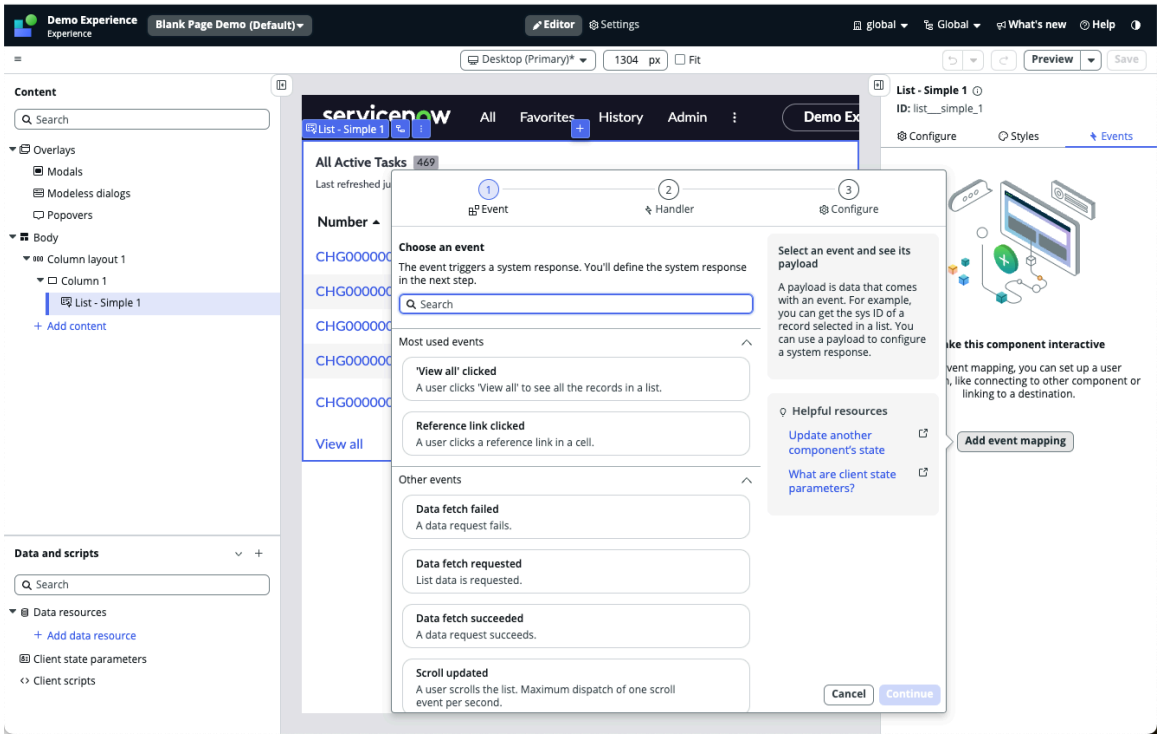
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Create or open a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Add a component that has events to which it can bind to your page, such as a button.
   For more information about adding components to a page, see Add and configure components.
5. To add an event handler to your component, select the **Events** tab.
   For more information on how to add event handlers to your component, see Manage actions in UI Builder pages.

   a. To start the process of setting up an event handler for your component, click **Add event mapping**.

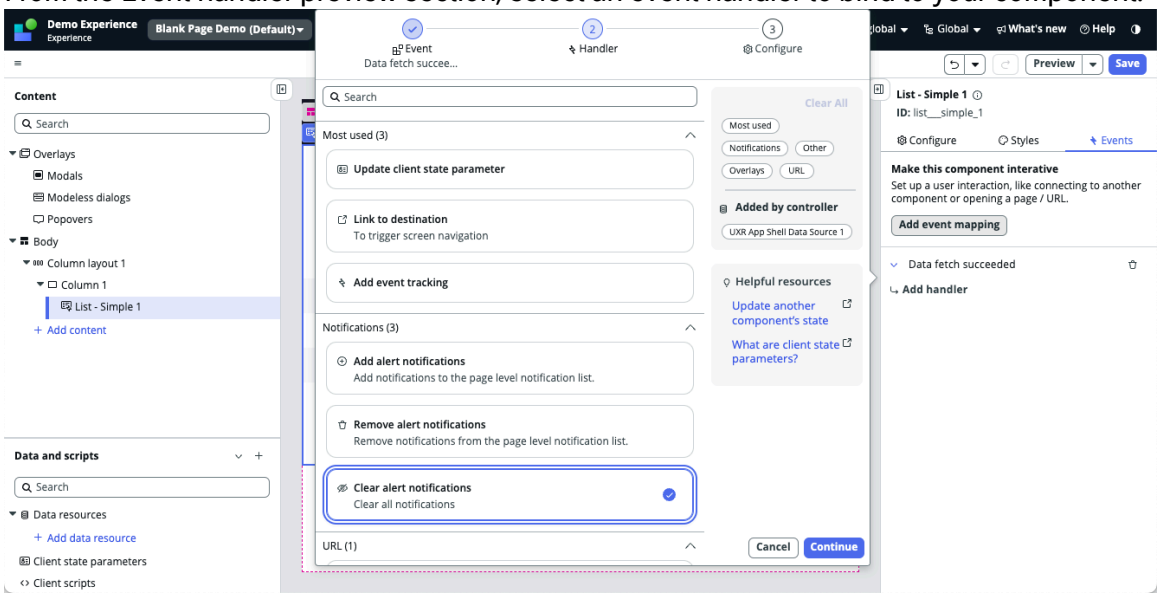


   b. Select the event mapping that you want to configure from the list.



c. Select **Continue**.

d. From the Event handler preview section, select an event handler to bind to your component.



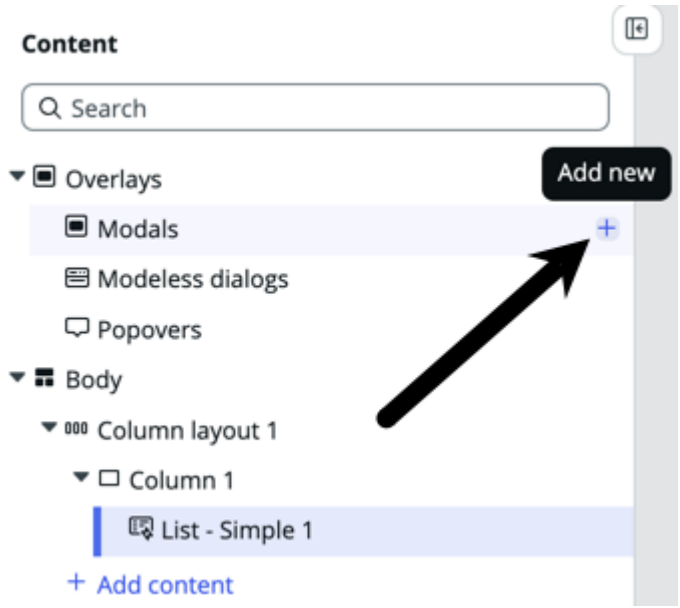
e. Select **Continue**.

f. Configure the payload for the event.
You configure each event handler differently, depending on the action of the event. For example, if you add an event handler for a button component, you can choose what that button action performs.

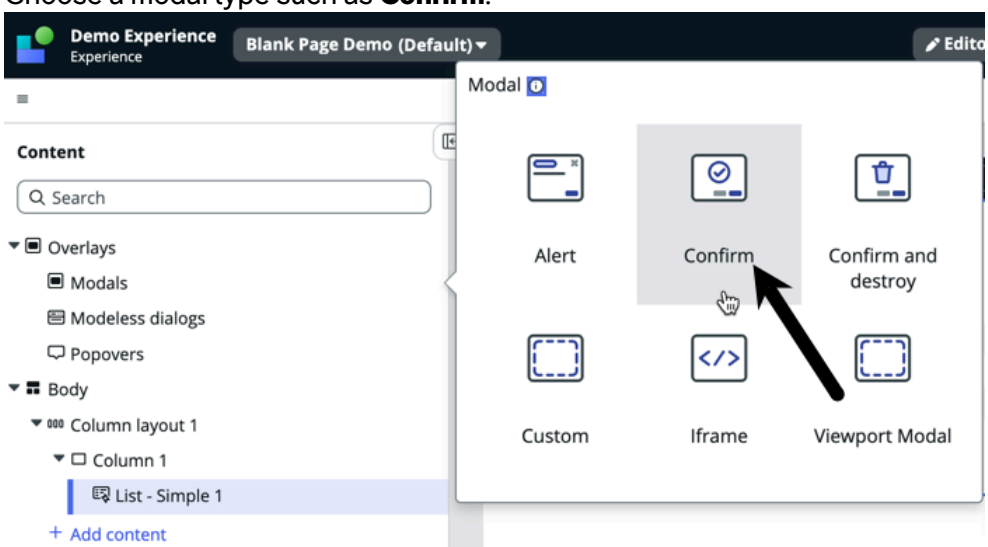
g. Select **Add**.

6. **Optional:** If you want a modal to pop up for your event, add the modal to the page before you bind your event to the component.
   A modal is a confirmation pop-up that appears when you click the component. For example, if you add a button component that deletes a record, you add a modal to ask the user to confirm that the user wants to delete the record. For more information, see Add modal to component.

   a. Select the **+** icon in the content tree next to **Modals**.



   b. Choose a modal type such as **Confirm**.



   c. Configure the modal.
   You can change the text in the modal, the names of the buttons in the modal, and the size of the modal screen. You can also set actions for the modal. When you finish configuring the modal, close it. Notice that the modal you created sits above the body of your page structure in the content tree.

7. Bind an event to a button component.
   You bind an event to the button to trigger an action. Select the button component to highlight it again, either in the content tree, or on the page.

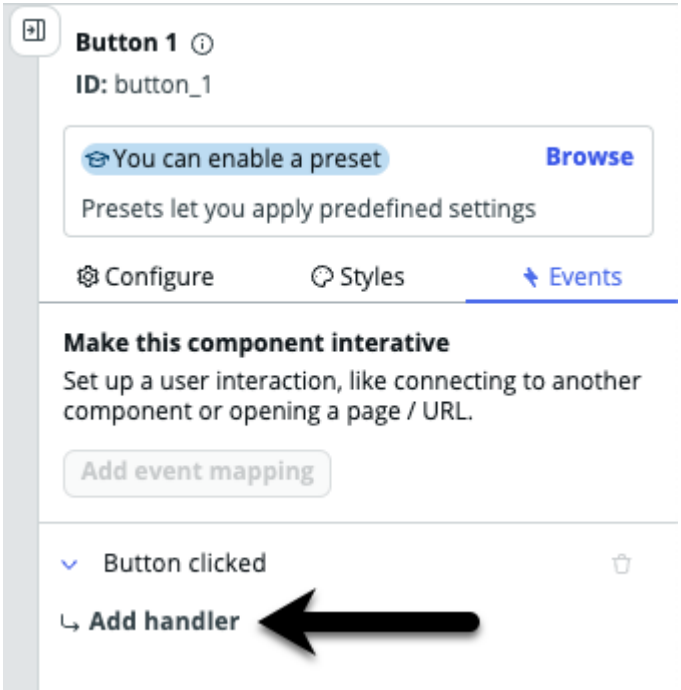
a. Select **+ Add content** in the content tree.

b. Select the **Button** component in the toolbox.

c. Select **None** when asked to choose a preset.

d. In the Configuration pane, click **Events**.

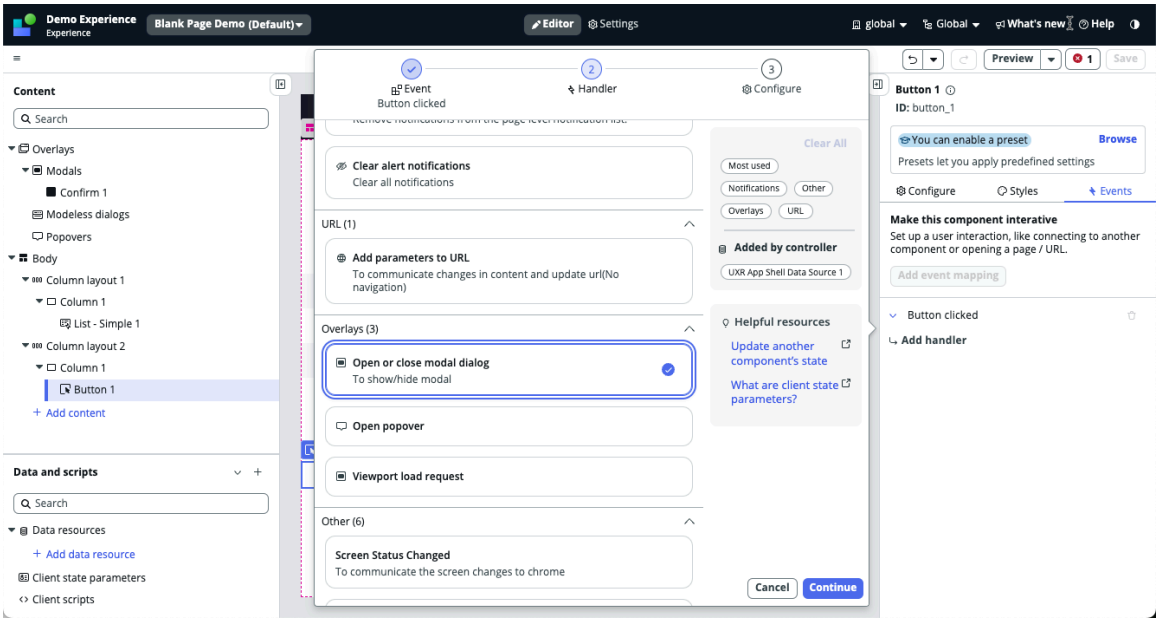
e. Select **Add handler**.



The button component only has the button-clicked event associated with it. Other components can have more than one event.

f. Select an handler that you want assigned to the button.

For example, to add an event handler for the button component, you could link it to another destination and add alert notifications. In this example, you can choose **Open or close modal dialog** to make the button open the modal that you created earlier. In this button scenario, select the Confirm and destroy modal dialog event handler that you created earlier.



g. Select **Continue**.

h. Select a modal from the drop-down.

i. Select **Add** to add the modal event handler to your button component.

8. Click **Save**.
9. Test the modal by selecting **Preview**.
10. To trigger the modal that you created, click the button on the page.

Bind an event to a page

Use page event mappings to bind data elements within UI Builder so that you can add event actions to your page.

Before you begin

Role required: ui_builder_admin

About this task

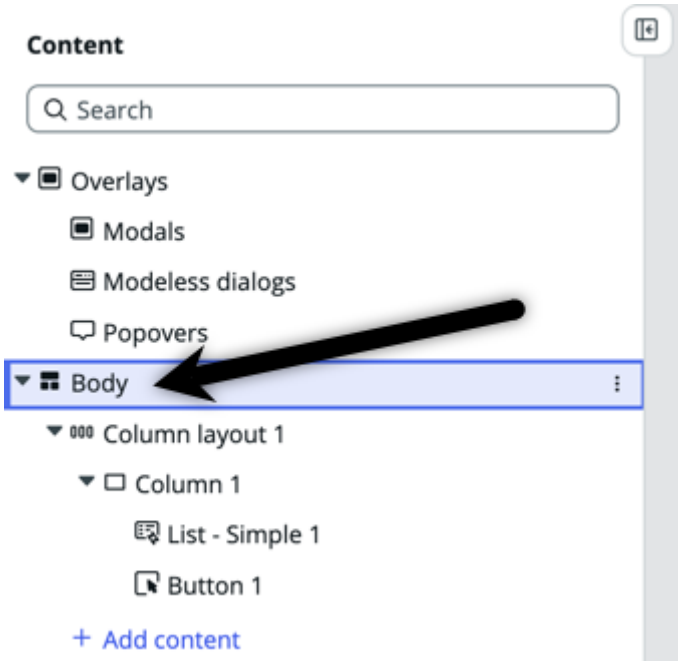
You can bind an event to a page by using the following types of events:

- Page event mappings. Add, remove, or clear alert notifications on your page.
- Variant event mappings. Add, remove, or clear alert notifications on your page variant.
- Dispatched events. Create dispatched events for your page to create relayed event mappings that model events after a parent event handler. Select a target parent event handler to model the payload fields after it.
- Handled events. A handled event is an event that is exposed and available for use by other users. After you create a handled event, it is available under *`Page event mappings`* for other users to use. You can also set up payload fields that you create manually or choose a template for your handled event, such as an open or close a modal dialog.

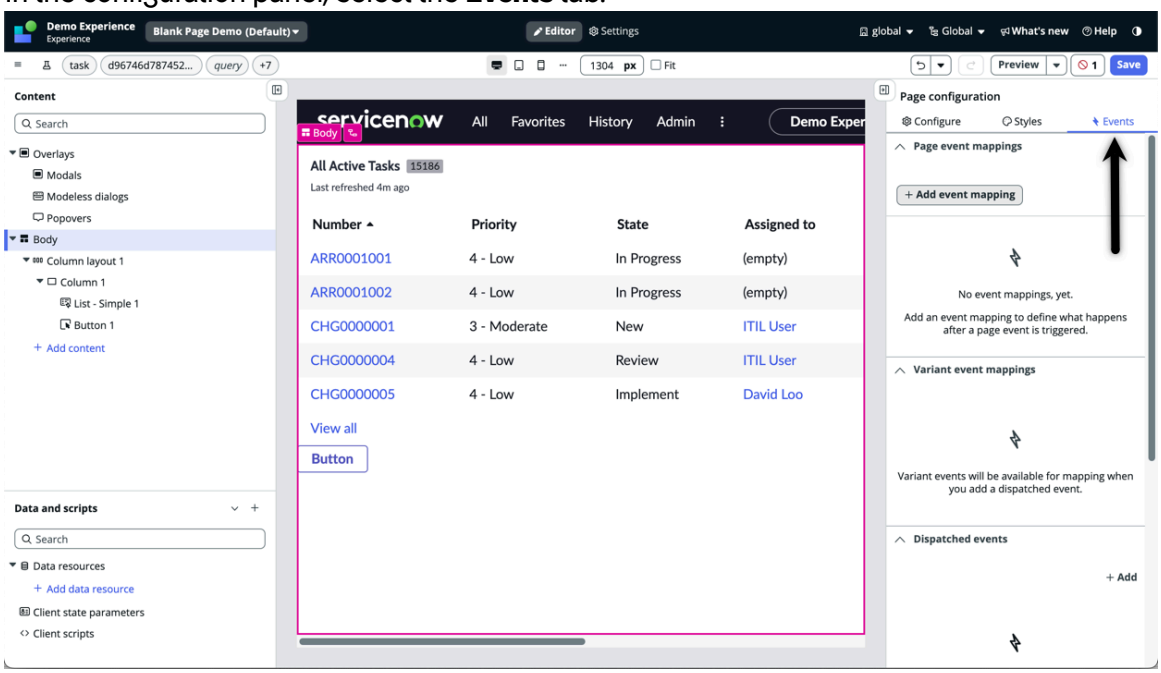
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information.
3. Open or create a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Highlight the page body in the content tree.

   The body is the top-level line of the content tree. When you highlight the whole page, you can add page-level events.



5. In the configuration panel, select the **Events** tab.



6. Add an event handler in one of the following ways.

7. Select **Save**.
8. To preview your page and test the data resource event that you set up, select Preview.

Bind an event to a data resource

Assign event handlers within UI Builder to individual data resources on your page. When a data resource successfully fetches new data, it executes an event handler to take a user to the next step in a flow.

Before you begin

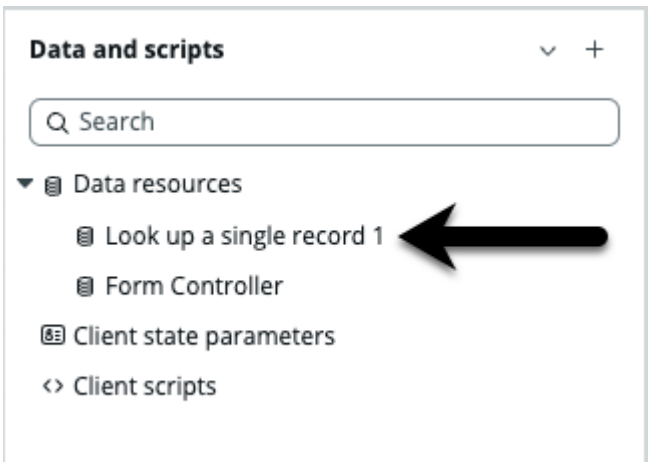
Role required: ui_builder_admin

About this task

Bind an event to a data resource, to perform data-related actions on your page. For example, you can add a button to reload data to a page if it doesn't load the first time. You can also set up a data resource event to reload data behind the scenes of your page.

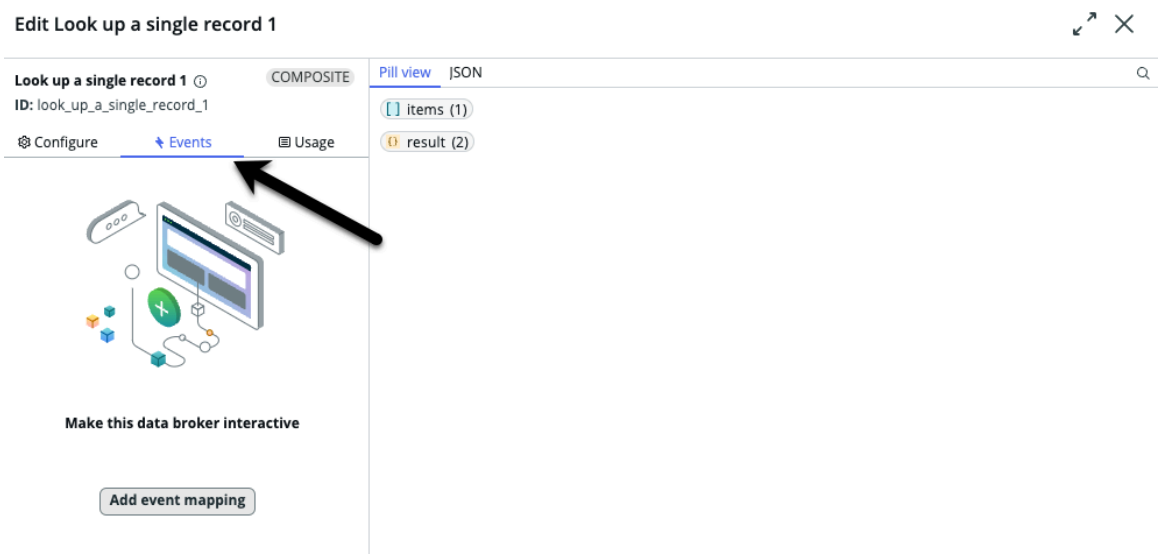
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Create or open a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. **Optional:** If you do not have any components on your page, add one to your page.
   For more information, see Add and configure components.
5. Add a data resource to your page.
   For more information, see Add and configure data resources to a page.
6. Select a data resource instance.

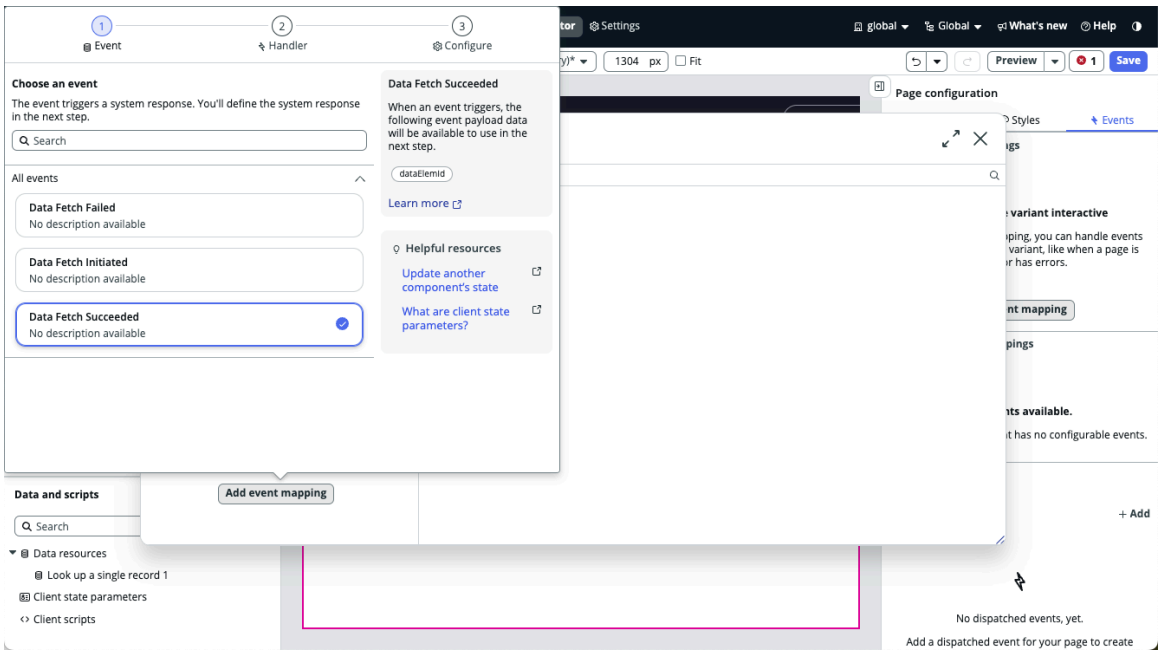


7. To add an event handler to your data resource, select the **Events** tab in the configuration panel of the data resource.

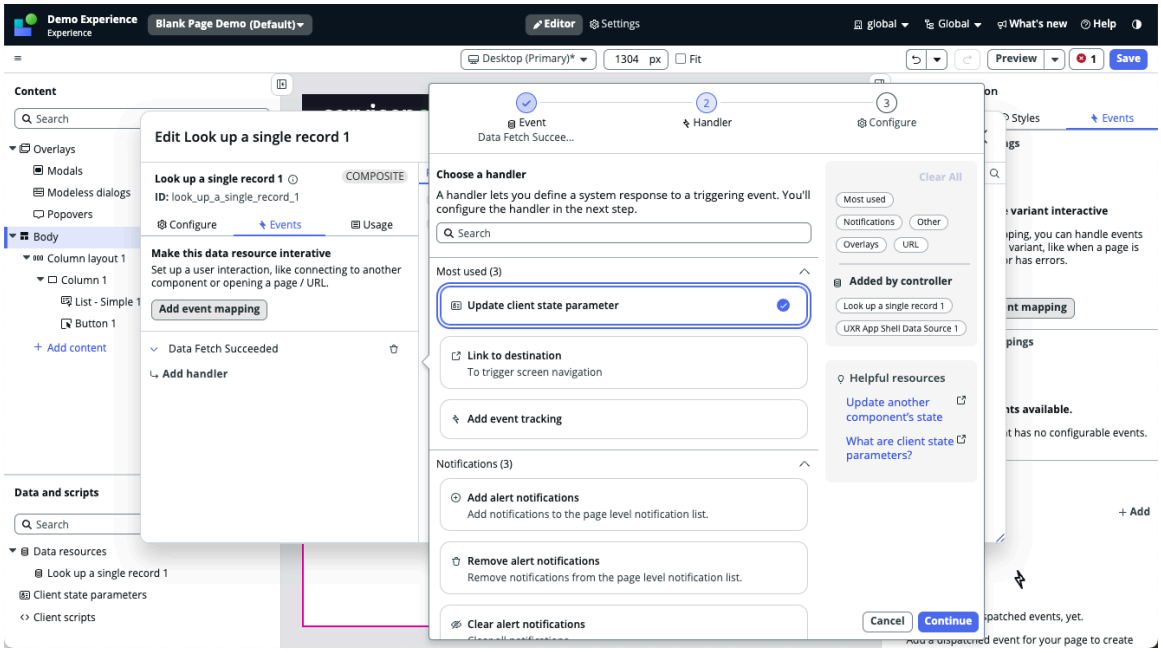
   You can add event handlers to your data resource to handle actions for Operation Initiated, Operation Succeeded, and Operation Failed.



8. Select **Add event mapping**.
9. Select an event from the list.



10. Select **Continue**.
11. Select an event handler from the list.



12. Select **Continue**.
13. Fill in the required information for the event handler.
    The configuration varies depending on which event handler is selected.
14. Select **Add**.
15. Select **Save**.
16. To preview your page and test the data resource event that you set up, select Preview in the UI Builder header.

Link an event to another page

Add a link to the destination event handler within UI Builder so that an event action can open another page. You can also configure the event handler to follow the App Route to the desired page.

Before you begin

You must have a workspace page that contains a component that is intended to open another page when a user clicks it. The dashboard overview is an example of such a component. Components such as **Link to Destination** do not support the link to destination event handle. The component link property takes priority over the link to destination event handler.

Role required: ui_builder_admin

About this task

To configure an event action to open another page, you must know what page you want to open, what the required and optional parameters are for that page, and what payload values to set on the event handler to pass the required parameters to the destination page.

Tip:

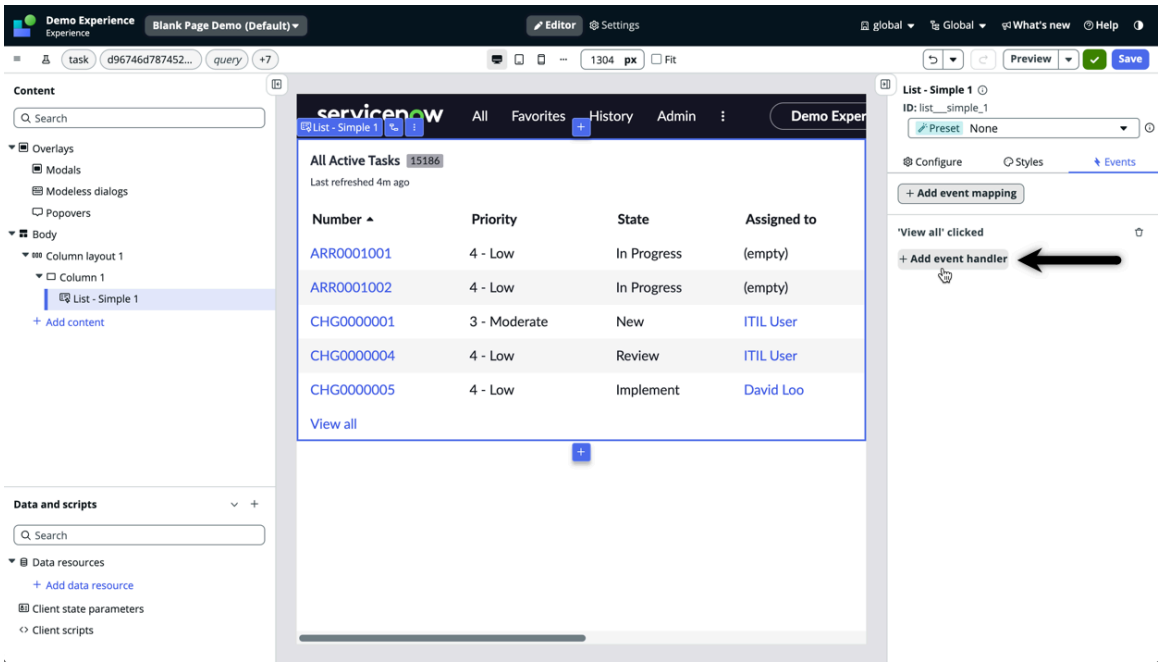
You may be able to find examples of both the components that you want to link from and the destination pages that you want to link to in the Base Agent Workspace Experience. This Next Experience is provided in the base system. If you create a page from a page template, you should only copy the contents of the template. Do not reference it. For more information about the difference between copying and referencing a page template, see Create a page from a template.

Procedure

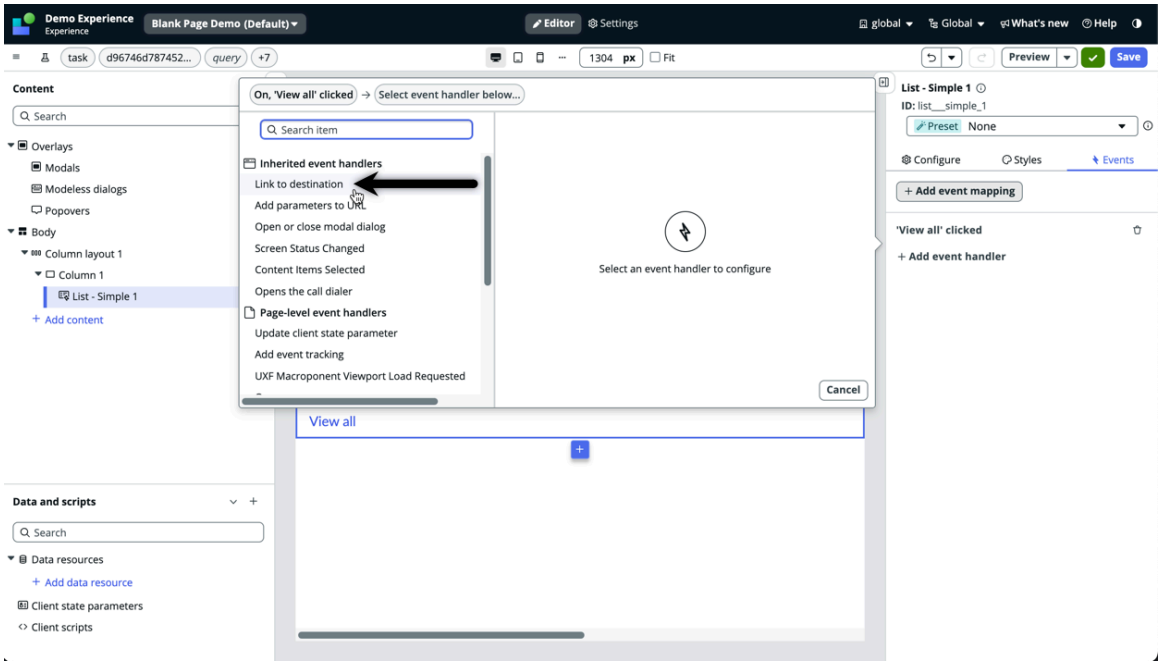
1. Open your experience in UI Builder.
2. **Optional:** If the destination page doesn't exist in your experience, create one.
   For information about creating pages, see Create a page in UI Builder. Make sure that you set the required and optional parameters for the page so that you can use it as a destination. If a particular component in the page is a destination, you must include that component. You also must configure the properties on the component to consume the page parameters with `@context.props.<parameter-name>` values.

   You might consider creating the page from a page template. The Base Agent Workspace Experience has several page templates that are already configured to be destinations for other components. If you create a destination page from a template, the components are already configured with the correct properties. Any necessary state parameters or client scripts are also copied over. You have to add the page parameters. You can copy these parameters from the UX App Routes related list on the Agent app config [sys_ux_app_config] record of the experience that contains the page templates.

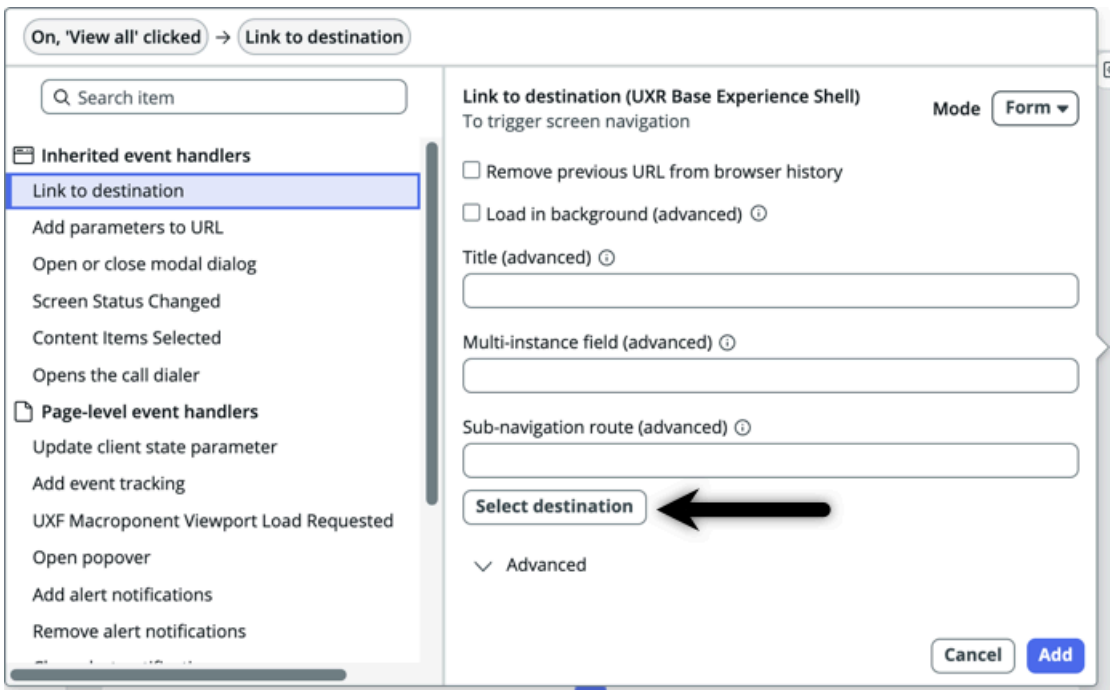
   To make sure that the pages that you are creating work reliably as destinations in your experience, your experience must have the same app shell UI as the experience with the page templates.
3. Switch to the page that you want to link to the destination page.
4. Navigate to the relevant component and select it.
5. Select the **Events** tab.
6. Select **+ Add event mapping**.
7. Select the event you want to use.
8. Select **+ Add event handler**.



9. In the Inherited event handlers section, select **Link to destination**.

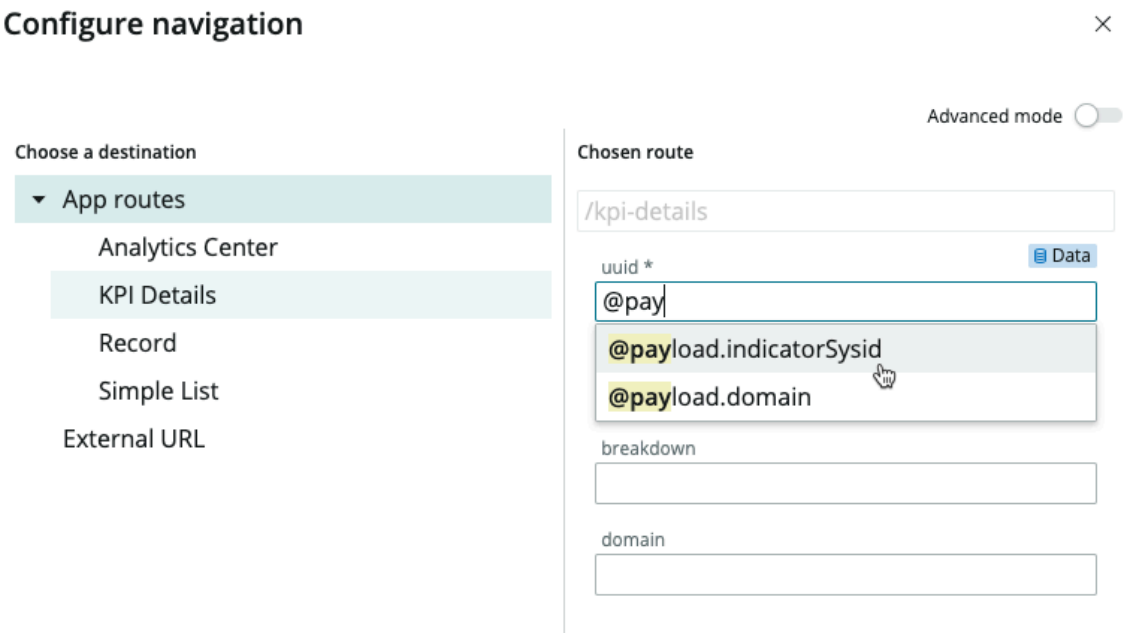


10. Click **Select destination**.



11. Expand **Pages** and select the page in the experience that you want to link to.
    Fields appear for each of the parameters on the destination page that the route leads to.
    Required parameters are marked with an asterisk (*).

12. Complete each required parameter field and the applicable parameter fields with an appropriate `@payload.*` value.
    If the developers of your component included default payload values in your event, you can select one through autocompletion. As shown in the following example, the payload value may not match the parameter name.



Note:
You have the option to link to an External URL instead of specifying an **App route**.

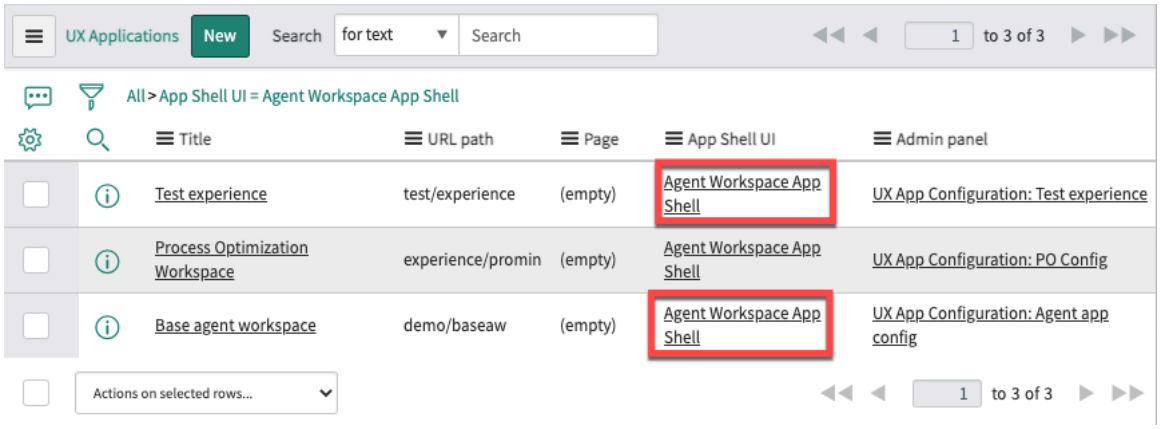
If no default values are provided, or you can't determine which values are correct for some fields, refer to the configuration and API documentation for the component in the ServiceNow® *Developer Site*. If you still can't find the necessary `@payload.*` values, contact Customer Service and Support.

Tip:
If you create your linking component by creating a page from a Base Agent Workspace page template, the component contains Link to destination Relay event handlers. These event handlers do not work. However, they contain the applicable `@payload.*` values for the parameters.

Example: Configuring the event handlers for an Analytics Q&A component

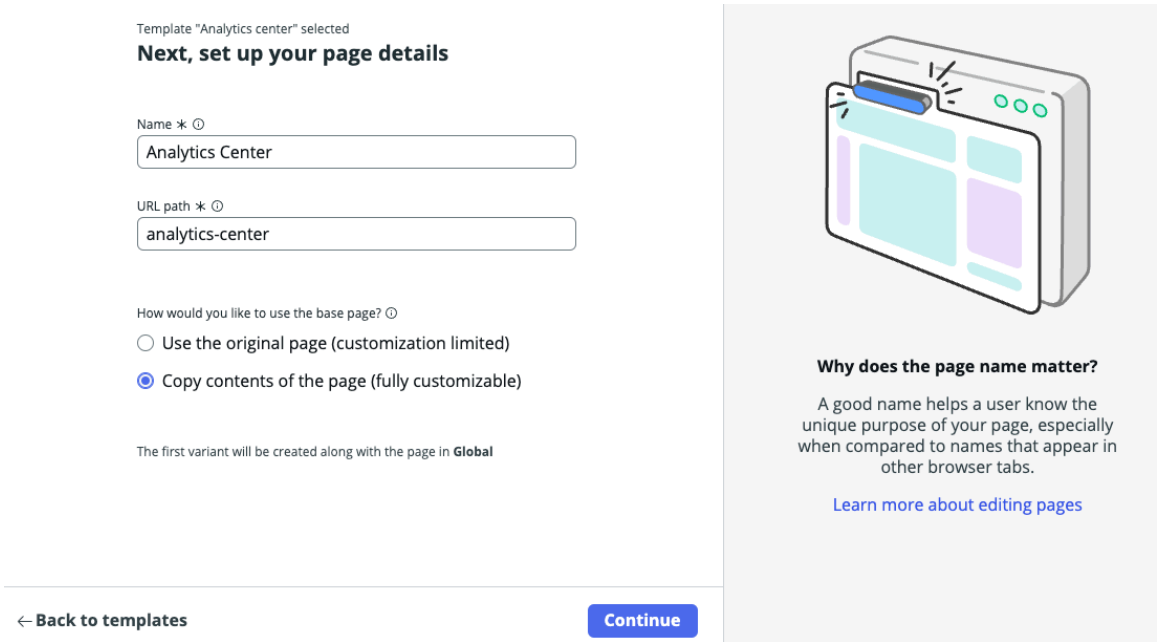
Let's say that you want to take a new Next Experience and add a page with an Analytics Q&A component. First, you create the page from the Analytics Center page template that is provided in the Base Agent Workspace experience. Next, you create a target page for the first of the three events in Analytics Q&A and then you configure an event handler for that event.

By navigating to **Now Experiences Framework** > **Experiences**, you see the Test experience UX application. Because it uses the same Agent Workspace App Shell UI as the Base Agent Workspace, you can use the page templates from the Base Agent Workspace.



You next select the Test workspace admin panel, find an UX App Configuration record with no UX app routes or pages, and then click **Open**.

As the example shows, in the UI Builder, you have created a page named Analytics Center that is based on the Analytics Center page template from the Base Agent Workspace. Next, you select the option to copy only the contents of the page template.



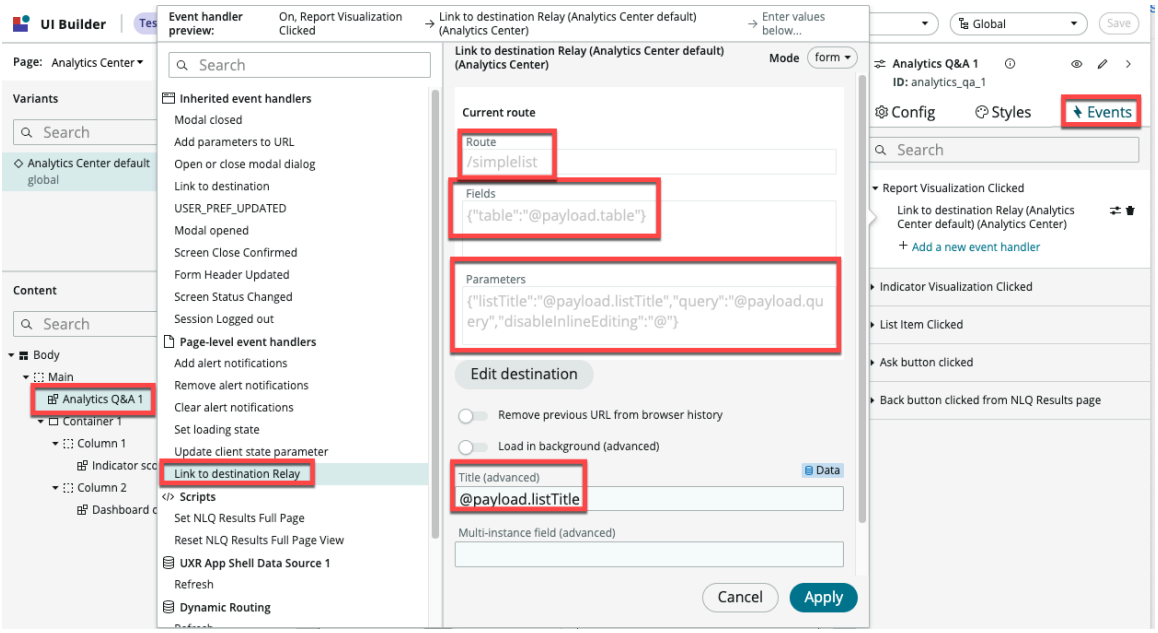
You select the Analytics Q&A 1 component and open the **Events** tab. From here, you can open the**Link to destination Relay** event handler for the **Report Visualization Clicked** event. When a question in Analytics Q&A returns a report, you can trigger this event by clicking a value in the report. When you click a value, you also see a list of the records that contribute to this value. In the **Route** field, you see that the destination is expected to be a page that is based on the Simple List page template. You also see the parameters of the page that the `@payload.*` values correspond to, and that the **Title** field can be populated with `@payload.listTitle`.

Parameters of suggested Simple List destination page and corresponding payloads

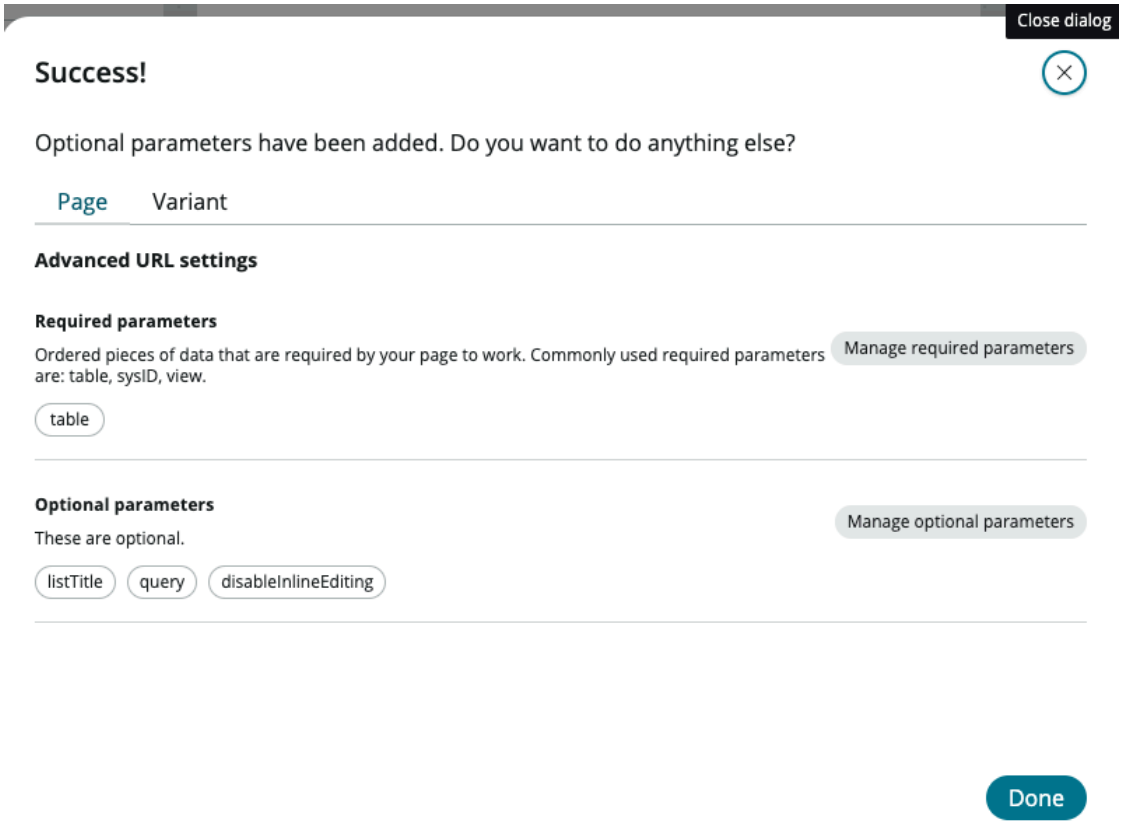
Parameter	@payload.* value
table (required)	@payload.table
listTitle	@payload.listTitle

Parameters of suggested Simple List destination page and corresponding payloads (continued)

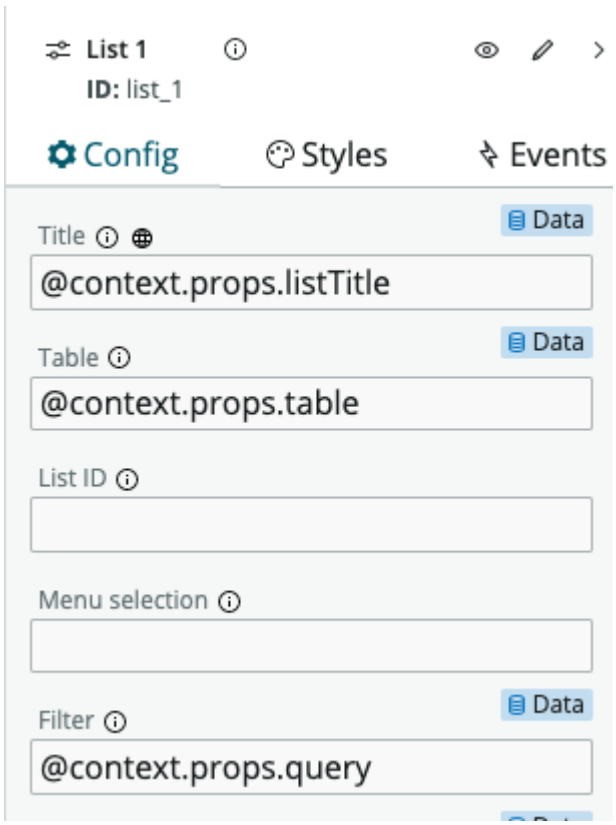
Parameter	@payload.* value
query	@payload.query
disableInlineEditing	none



Next, you navigate to **Menu** > **Create page** and create a page that is based on the Simple List template. Let's say that you name the page as Record list. You then follow a similar process as when you created the Analytics Center page. This time, in the last steps of the process, you would add `table` as a required parameter and `listTitle`, `query`, and `disableInlineEditing` as optional parameters.

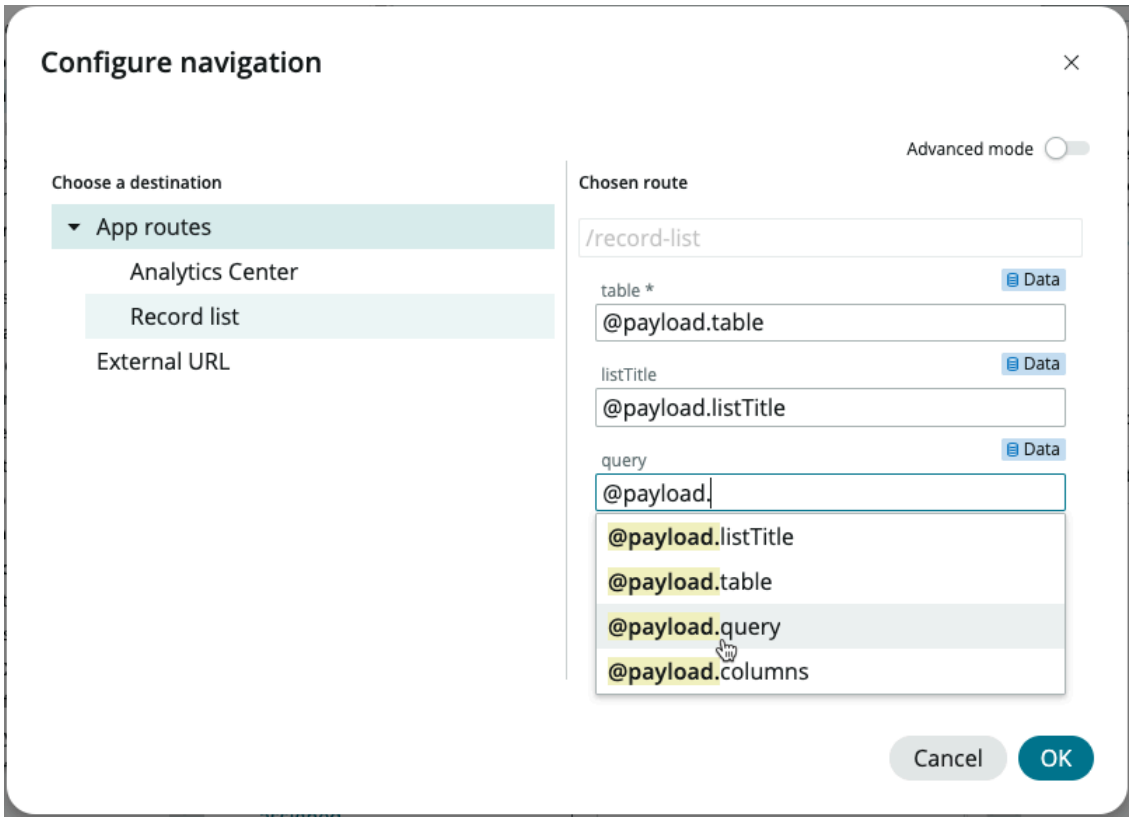


Because this page already contains a List component, when you open the **Config** tab for this component, you see that the parameters are already passed in the `@context.props.*` values.



Now, you return to the Analytics Center page. In the **Report Visualization Clicked** event, you add a new event handler. Next, you select the Record list page that you created and add the

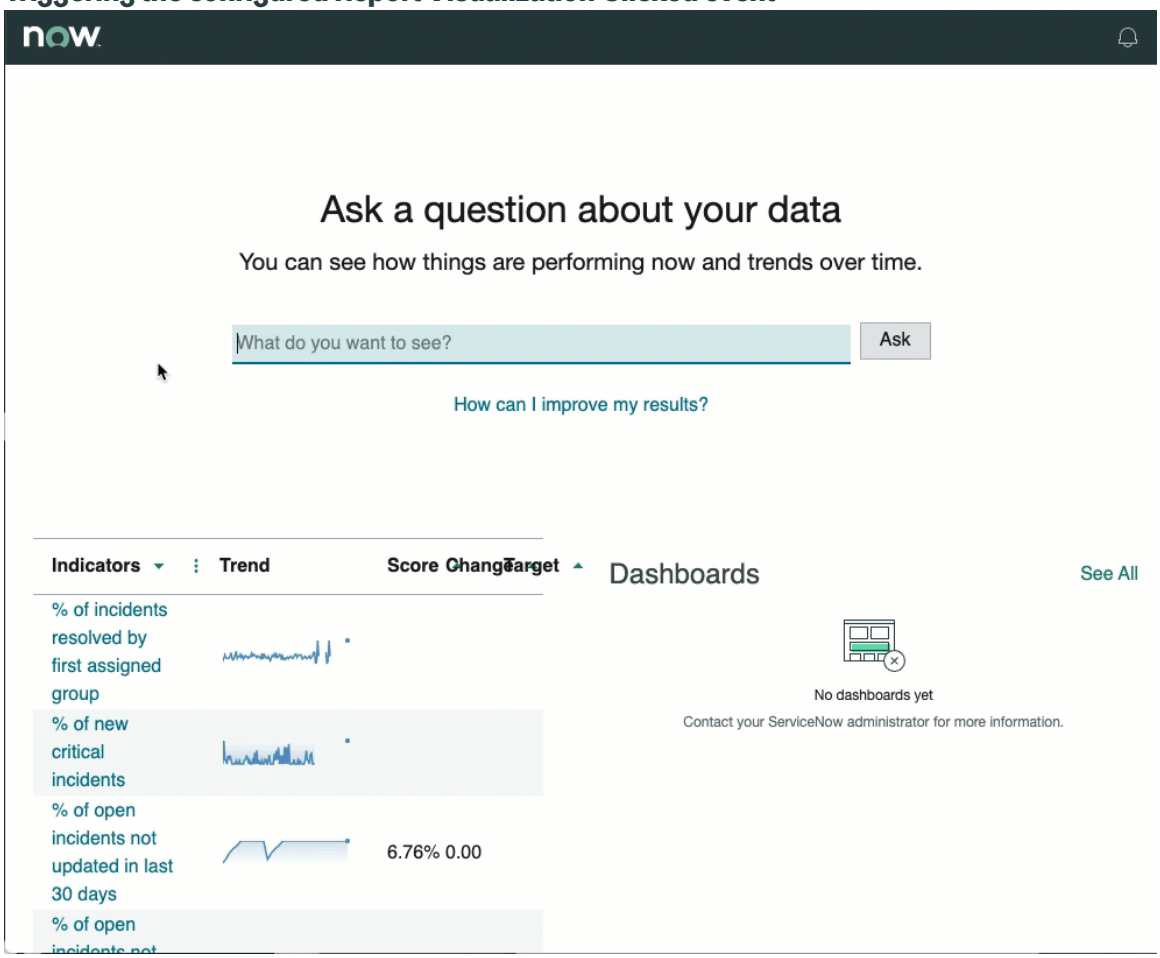
`@payload.*` values in the **table**, **listTitle**, and **query** fields, following the information that you got from the **Link to destination Relay** event handler. Predictive typing helps you to fill in these fields.



After you click **OK** and add `@payload.listTitle` as the **Title**, the event handler is done. You can now delete the**Link to destination Relay** event handler for this event.

The following example shows an Analytics Center page. On this page, you can enter a query for Incidents by Priority and get a report as a result. Also, by clicking a column, you trigger a **Report Visualization Clicked** event. The event handler enables you to see a simple list of the incidents in the report.

Triggering the configured Report Visualization Clicked event



Bind an event to a declarative action

Bind data elements within UI Builder so that you can add event actions to a declarative action.

Before you begin

Role required: ui_builder_admin

About this task

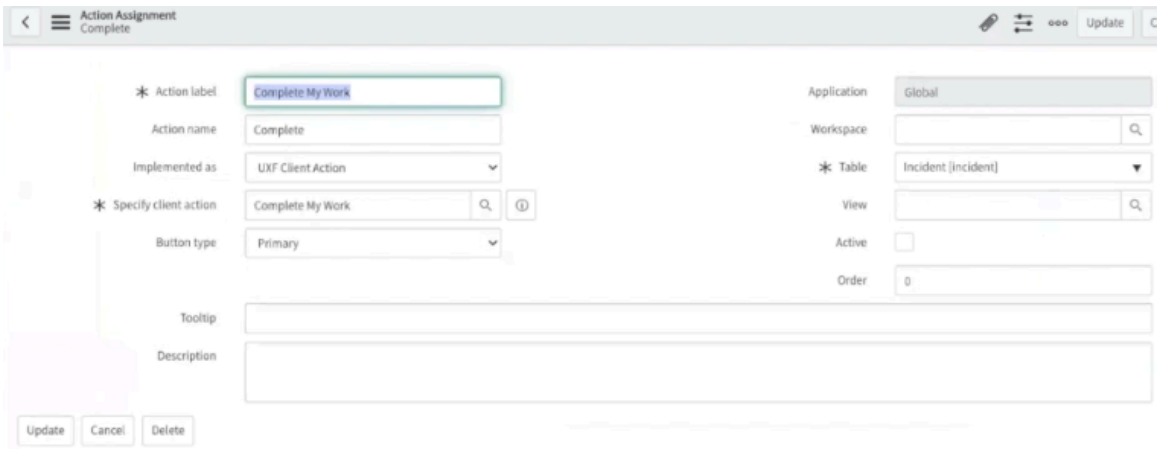
Bind a handled event to a component so that an action is performed when a user selects the component.

Procedure

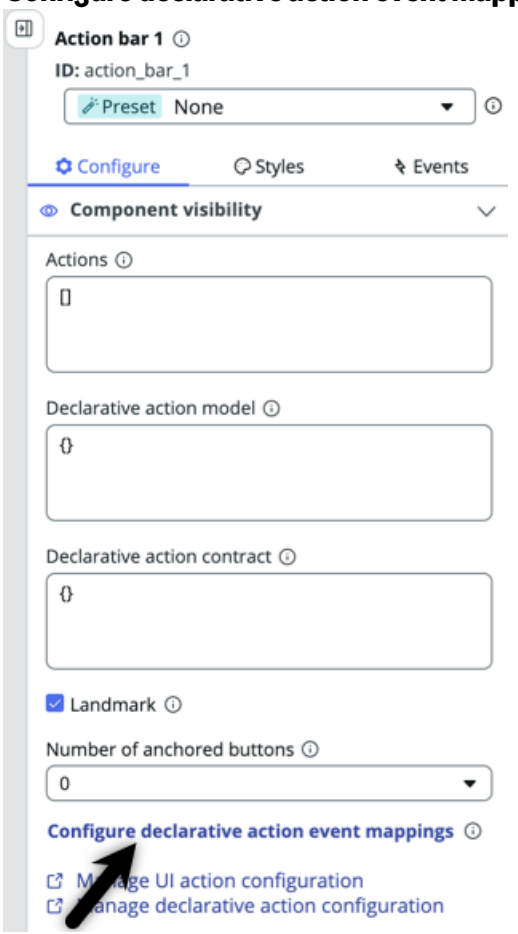
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Open or create a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Add a component to your page that can have a declarative action, such as an action bar or related list.
   For more information, see Add and configure components.
5. To create a declarative action definition in a table in the ServiceNow AI Platform®, navigate to **Workspace Experience** > **Actions & Components** > **List actions**.

Choose a table where you want the declarative action to be available in. For example, you could create a Complete my work action in an incident table or you could use an existing declarative action definition record.

Platform declarative action definition record



6. Select the **Active** check box, and then save or update the record.
7. Return to the UI Builder.
8. To invoke a handled event for the declarative action, go to the Configure panel and click **Configure declarative action event mapping**.



9. Choose the declarative action that you created earlier.
   To continue with the example in step 5, the declarative action could be something like **Complete my work**.

10. To define what the declarative action does on your page, click **+ Add event handler**.

    a. Give the event handler a name.
    The name should describe what action you want the event handler to perform.

    b. **Optional:** Provide a description of the event handler.

    c. Choose the handled event that you want to invoke.

    d. **Optional:** Add payload values for your event handler.

    e. Select **Save**.

11. Select **Done**.

12. Select **Save**, and then select Preview.

13. Test the declarative action on your page by clicking **Complete my work** to see if it works.

Disable component preset event mappings

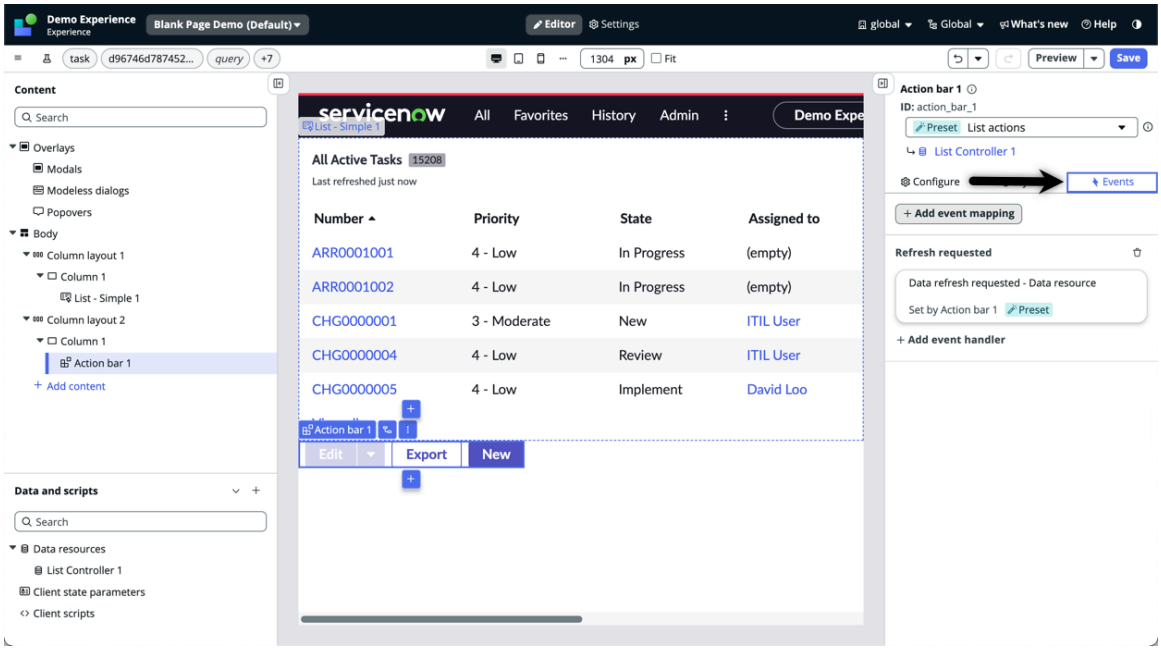
You can disable event mappings that are automatically created by a component preset.

Before you begin

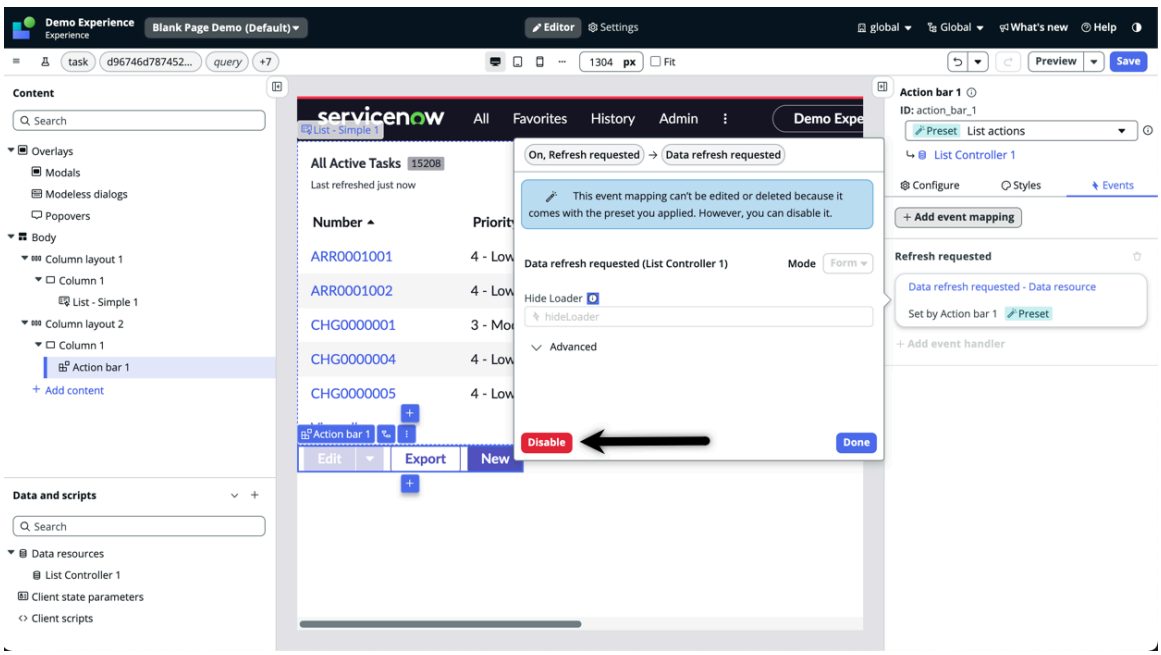
Role required: admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Open or create a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Select a component with a component preset configured.
5. Select the **Events** tab in the configuration panel.



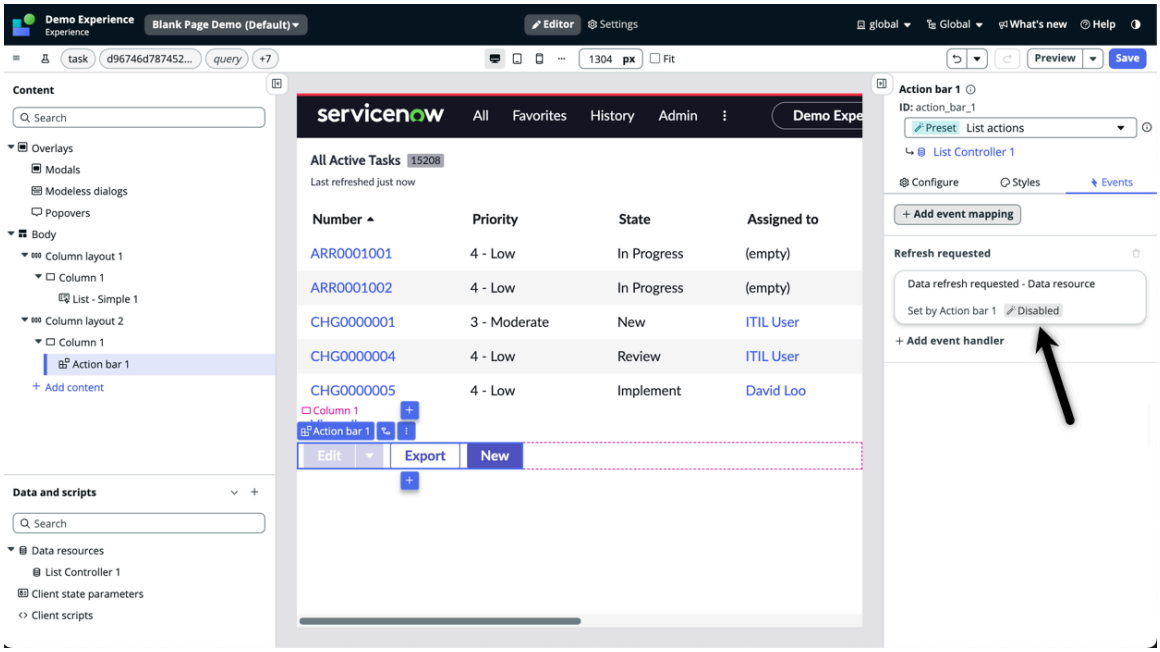
6. Select the event that you want to disable.
7. Select **Disable**.



8. Select **Done**.

Result

The component preset event shows **Disabled** in the **Events** tab of the configuration panel.



Delete an event handler

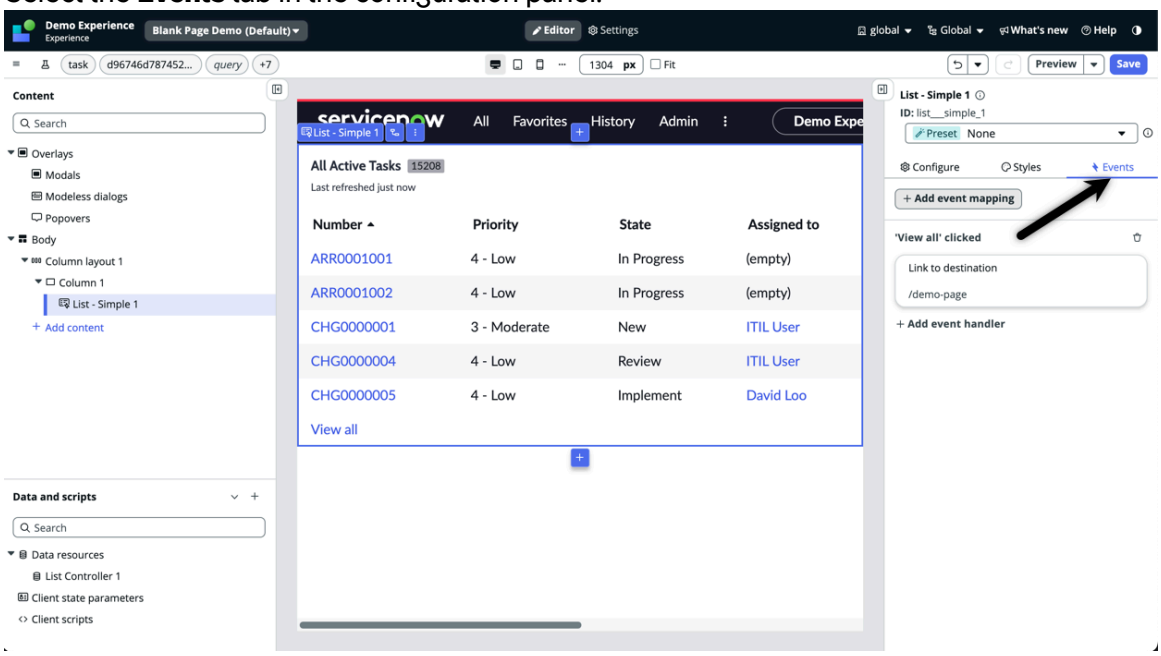
Delete an event handler that you no longer need in UI Builder.

Before you begin

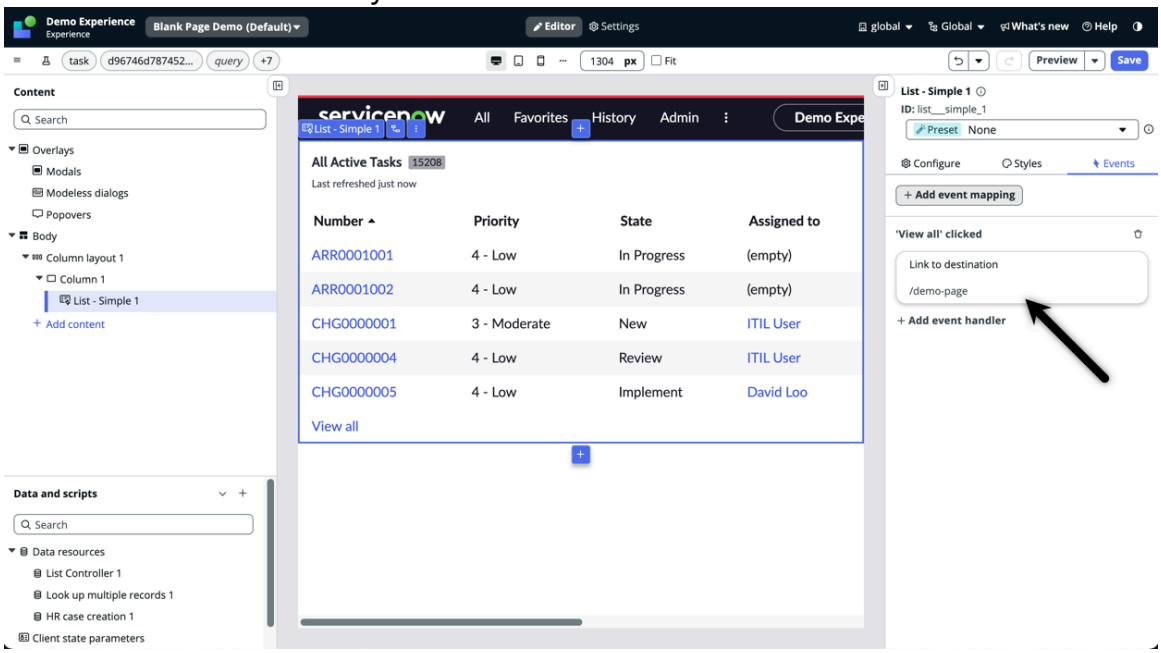
Role required: admin

Procedure

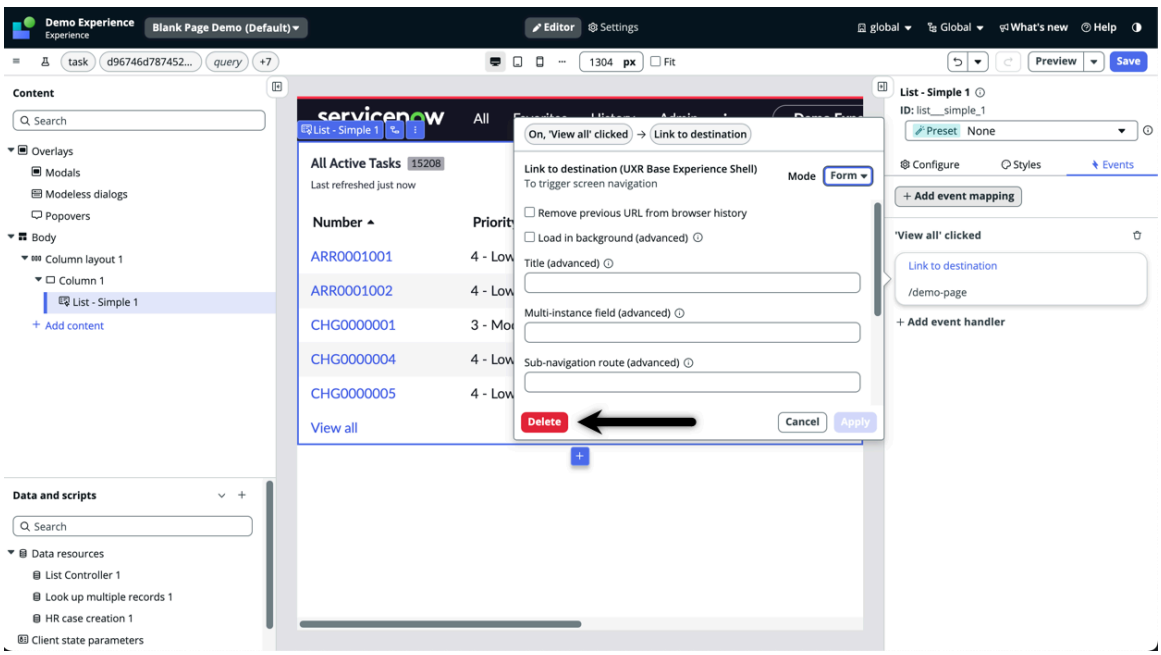
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Open or create a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Select a component with a preset configured.
5. Select the **Events** tab in the configuration panel.



6. Select the event handler that you want to delete.



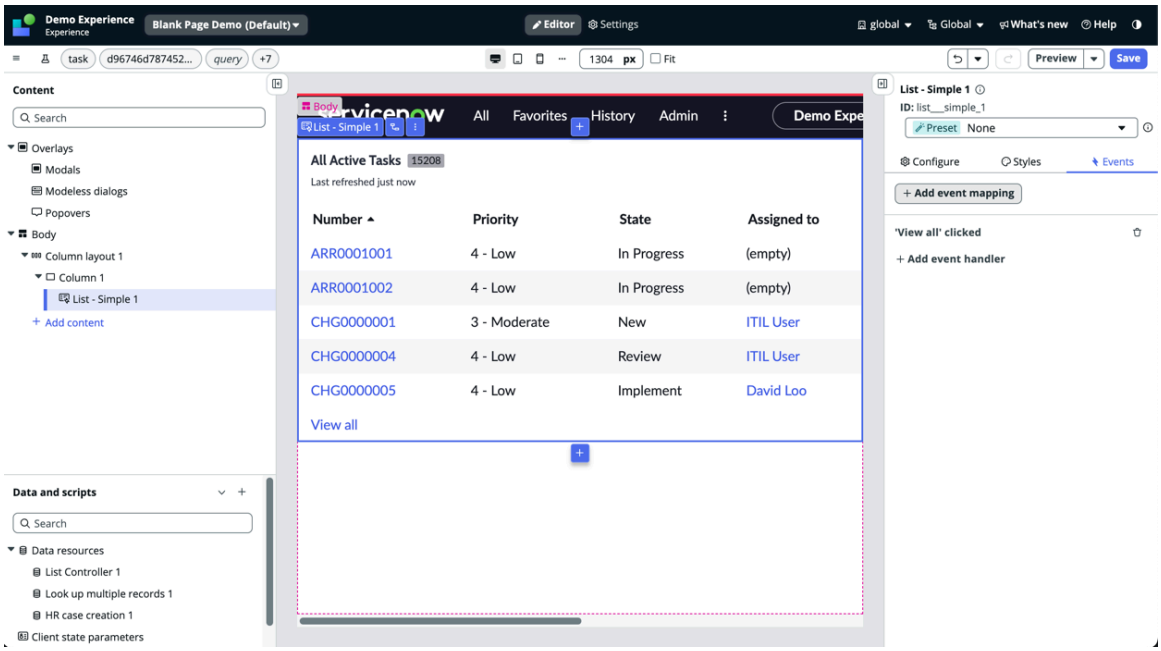
7. Select **Delete**.



8. Select **Delete** in the confirmation modal.

9. The event handler is removed from the associated event mapping.

If you would like to delete the event mapping, see Delete an event mapping.



Delete an event mapping

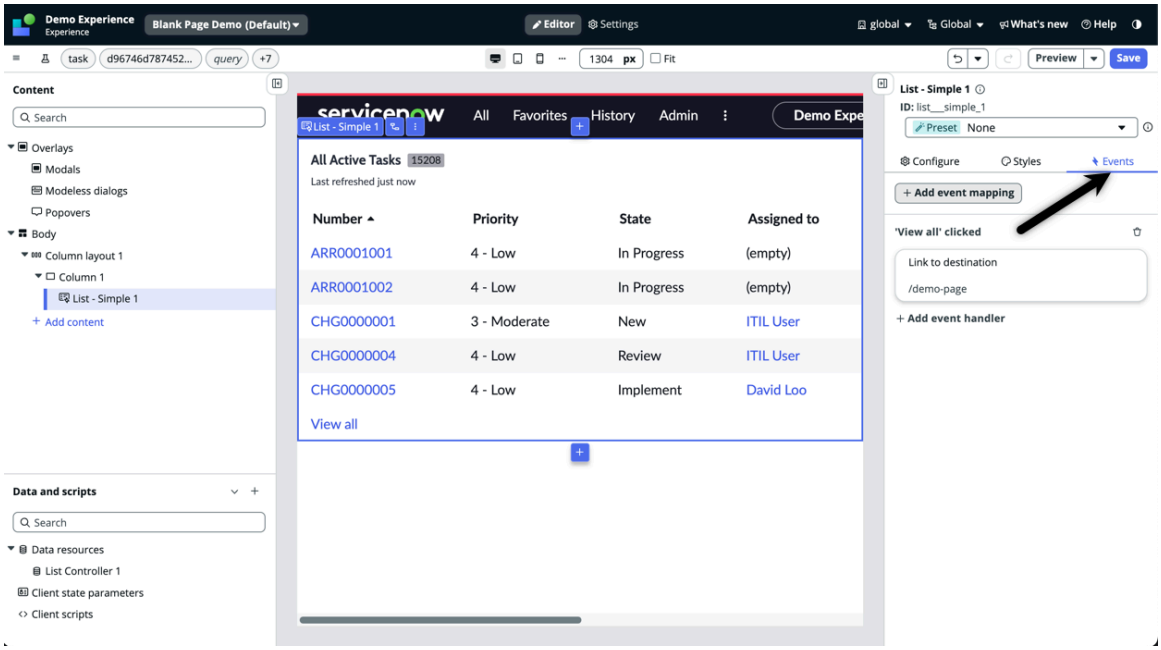
Delete an event mapping that you no longer need in UI Builder.

Before you begin

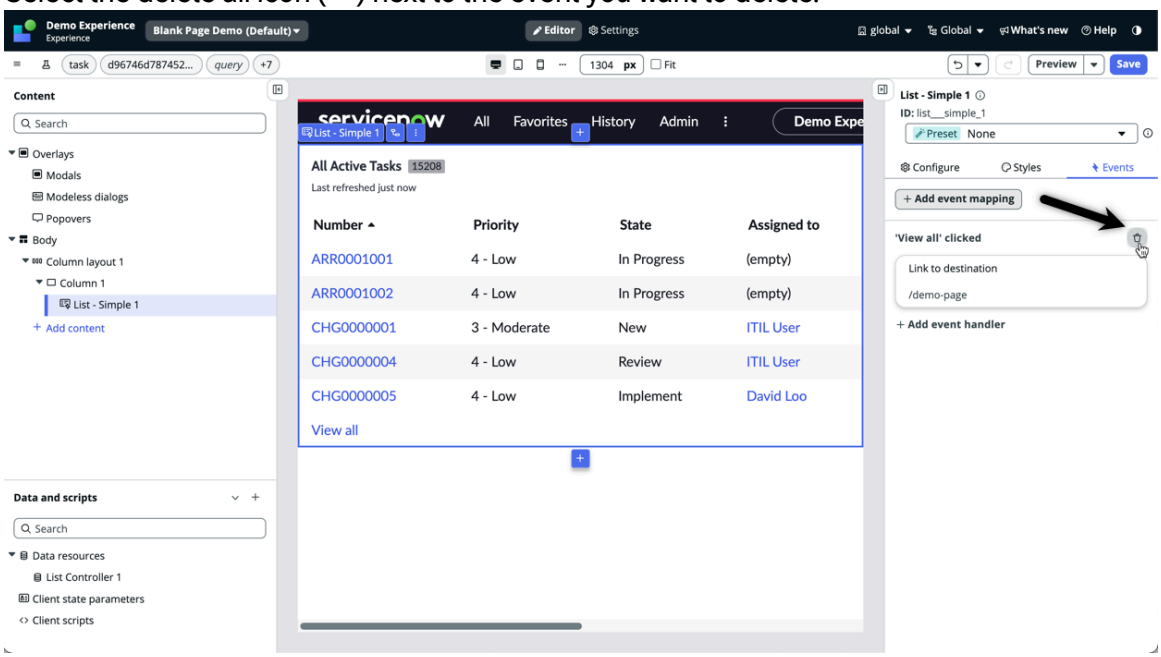
Role required: admin

Procedure

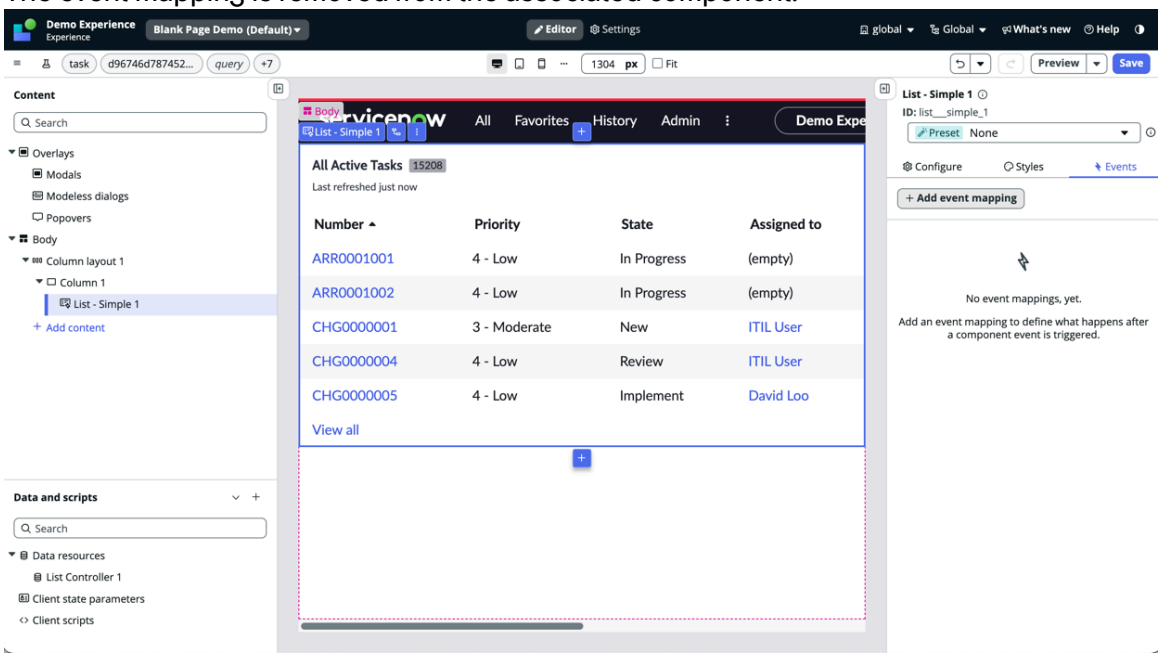
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Open or create a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Select a component with a preset configured.
5. Select the **Events** tab in the configuration panel.



6. Select the delete all icon ( ) next to the event you want to delete.



7. The event mapping is removed from the associated component.



Manage the visual style of UI Builder experiences

Themes enable you to change the visual style of your UI Builder experiences so that they express the look and feel of your brand

Getting started with themes

By default, the Polaris theme is applied to UI Builder experiences. For more information, see Working with themes in Next Experience.

Create a new theme with Theme Builder

Theme Builder enables you to create, edit, manage, and apply Next Experience themes using a friendly visual interface. For more information, see Configuring Next Experience with Theme Builder.

Create a custom theme

If you are an admin user comfortable working in style records and JSON code, you can create a custom theme to override the default Polaris style. This option gives you the most control over typefaces, colors, and images (including the banner and logo) in a theme. For more information, see Configuring Next Experience themes and preferences.

Working with dark theme

If you are interested in dark theme, see Working with the dark theme.

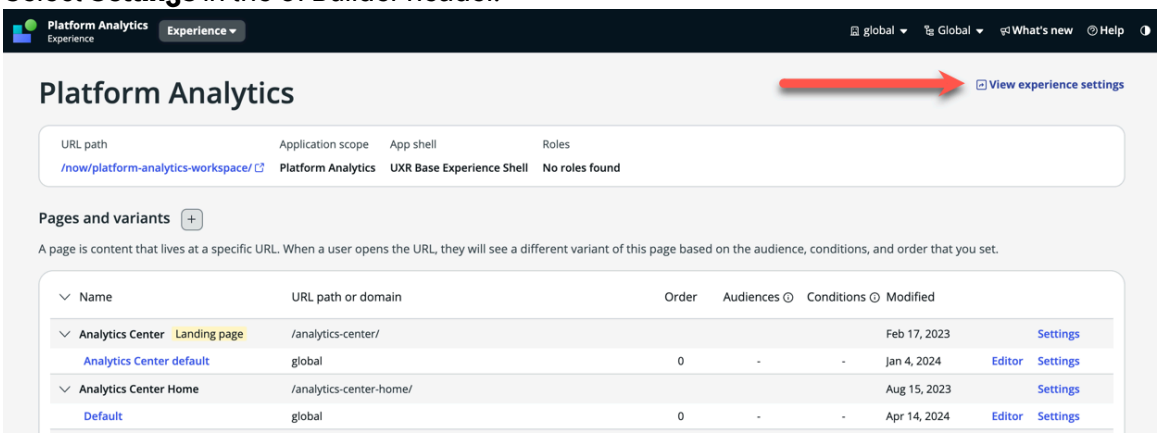
View experience theme

View the theme applied to your experience. The theme sets the visual style of the experience and provides a consistent look and feel across all pages.

Before you begin
Role required: ui_builder_admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Select **Settings** in the UI Builder header.



4. Scroll to the **Branding and theming** section and note what is listed in **Themes**. By default, the Polaris theme is applied to UI Builder experiences.

What to do next

If you want to customize the theme, there are two options:

- **Theme Builder** enables you to create, edit, manage, and apply Next Experience themes using a friendly visual interface. For more information, see Configuring Next Experience with Theme Builder.
- If you are an admin user comfortable working in style records and JSON code, you can create a custom theme to override the default Polaris style. This option gives you the most control over typefaces, colors, and images (including the banner and logo) in a theme. For more information, see Configuring Next Experience themes and preferences.

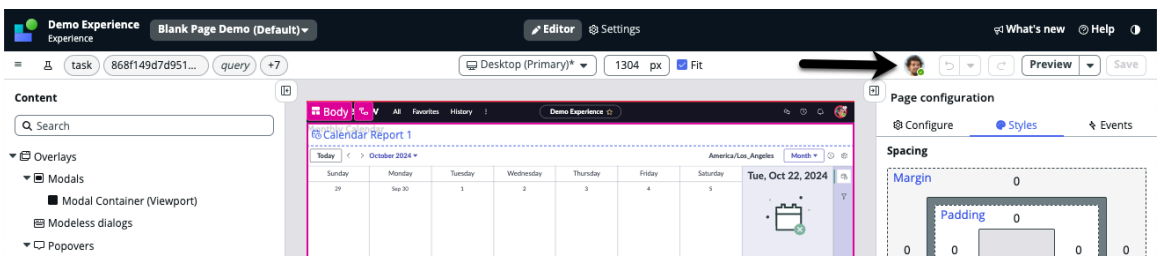
Collaborate with other UI Builder developers

UI Builder provides real-time collaboration tools and user presence indicators for more efficient and intuitive UI development.

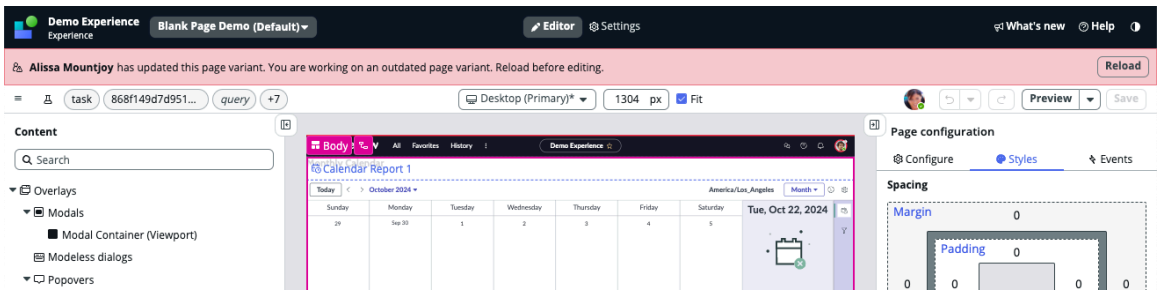
UI Builder lets you see who is online and working on the same page in an experience.

The avatars of other developers appear in the UI Builder header next to the undo button. A dot on the avatar of the user represents online status.

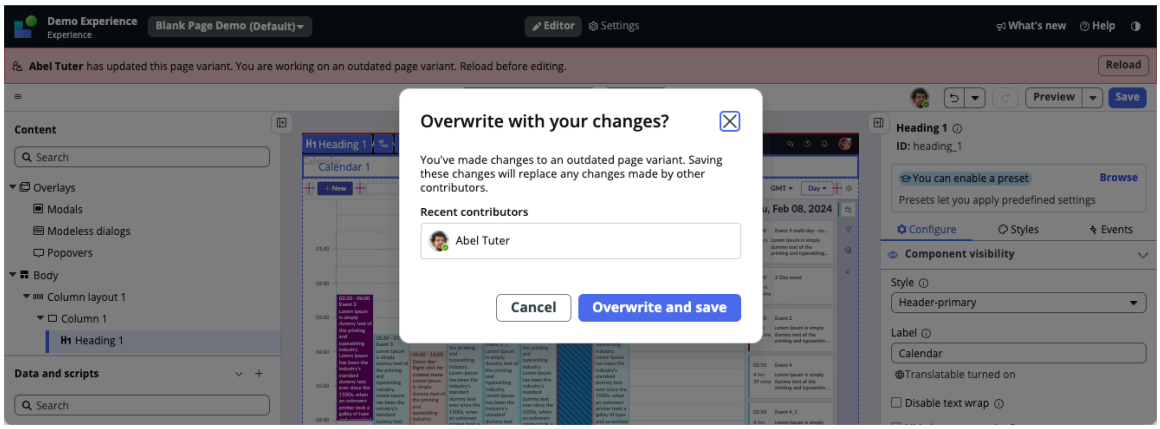
- Green dot if the user is currently working on the page
- No dot if the user is not logged in



A banner will appear asking you to reload the page to view the latest version if changes are made by other users while you are editing a page.



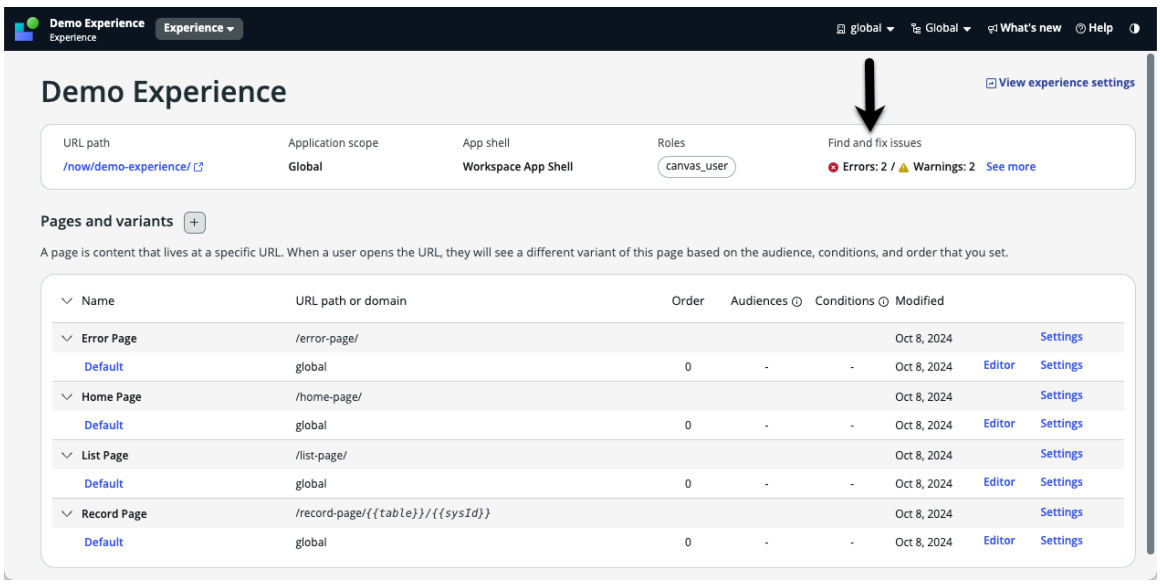
If you have made changes at the same time as another user you will be prompted to **Overwrite and save** your changes to proceed with previewing the page.



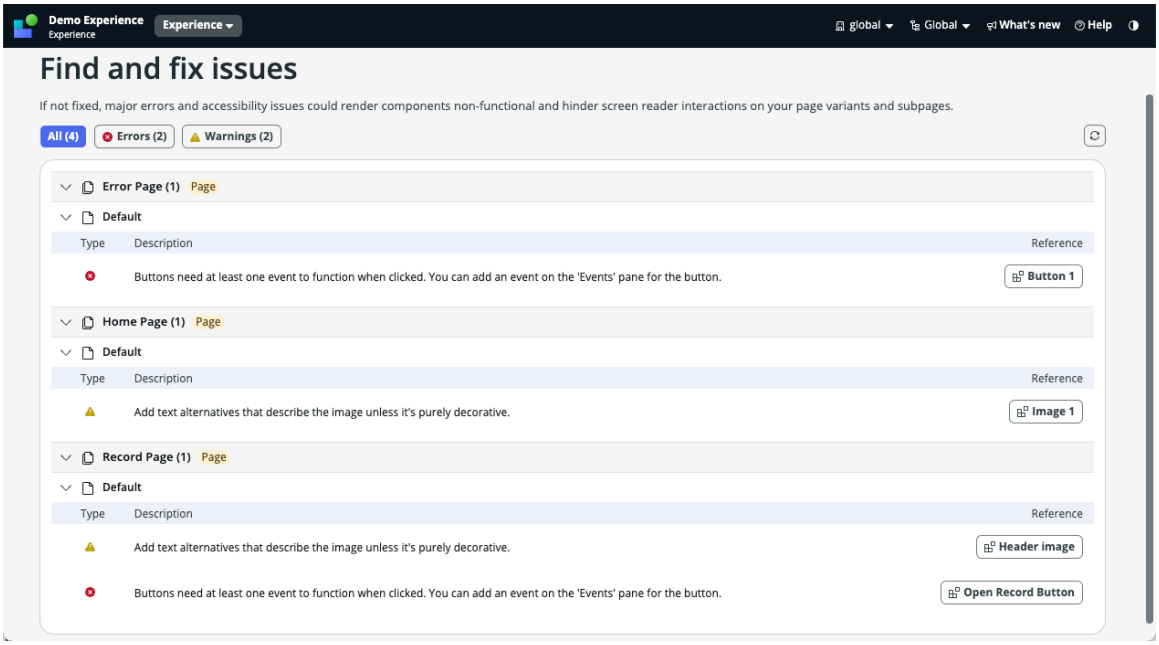
Find and fix issues in UI Builder

UI Builder can help identify common configuration issues and guidance on how to fix them.

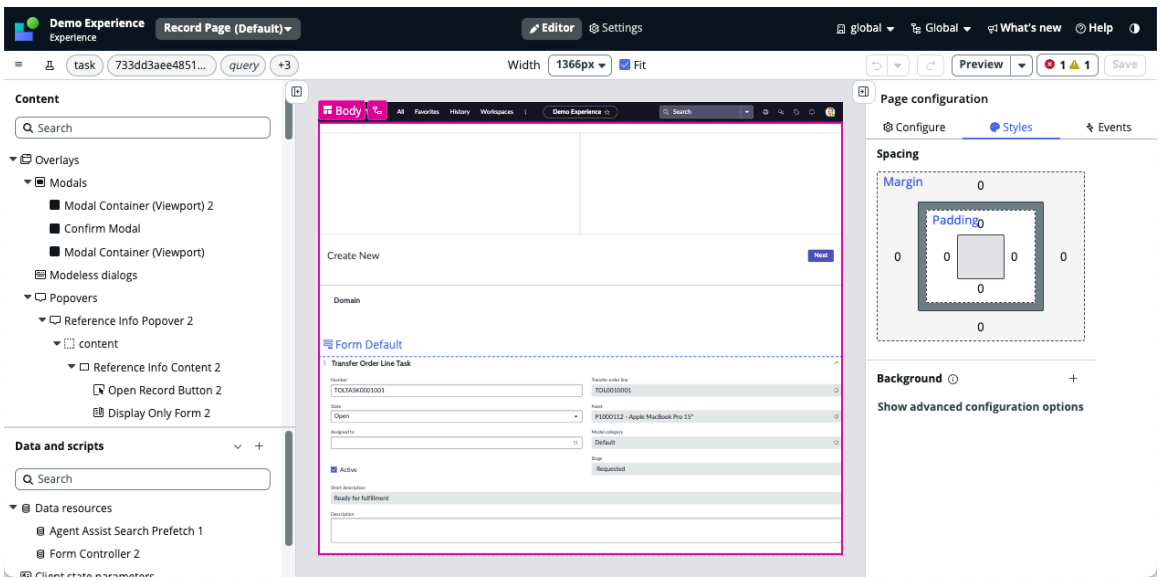
UI Builder can help you find and fix issues by checking your experience for missing configurations, errors, and accessibility standards. The list of issues can be found within the experience view of UI Builder under the **Find and fix issues** section.



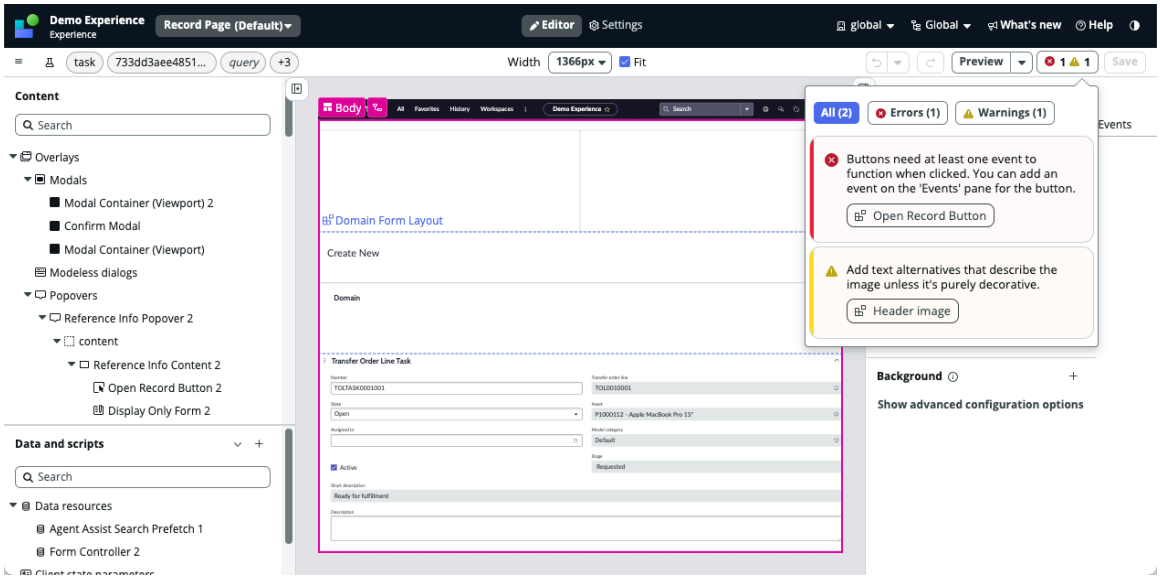
Find and fix issues list



You can also find UI Builder issues in the header of the page builder next to the **Save** button. The button auto updates as it finds errors and recommendations for the content on your page.



To find more information, select the builder icon to view the list of recommendations made by UI Builder. Each error and warning will list the component name that contains the issue.



Find and fix issues in UI Builder

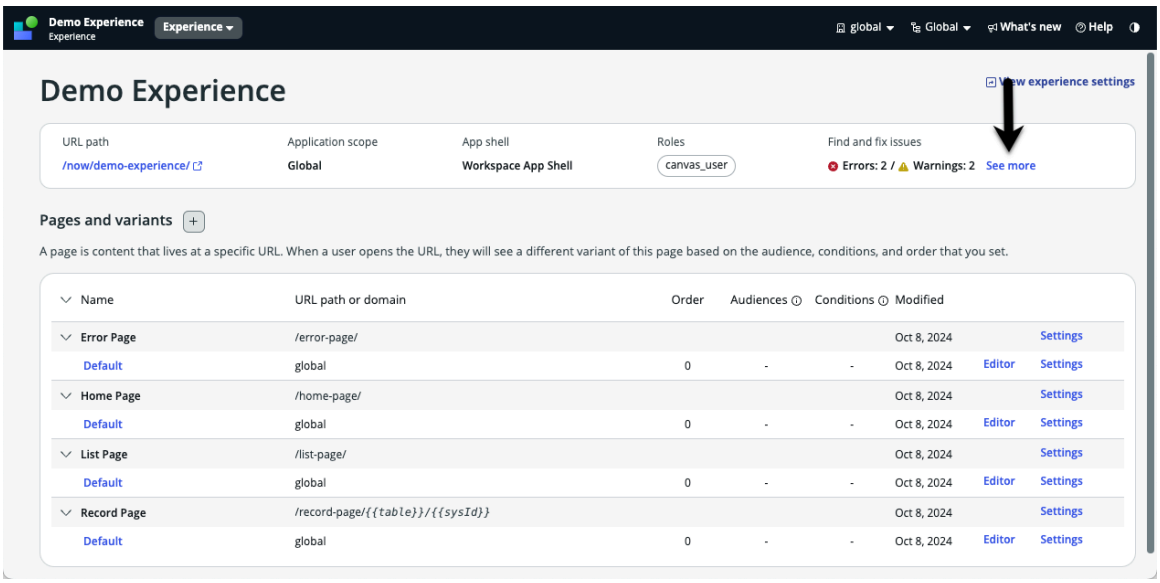
Find and fix issues found in your UI Builder experience.

Before you begin

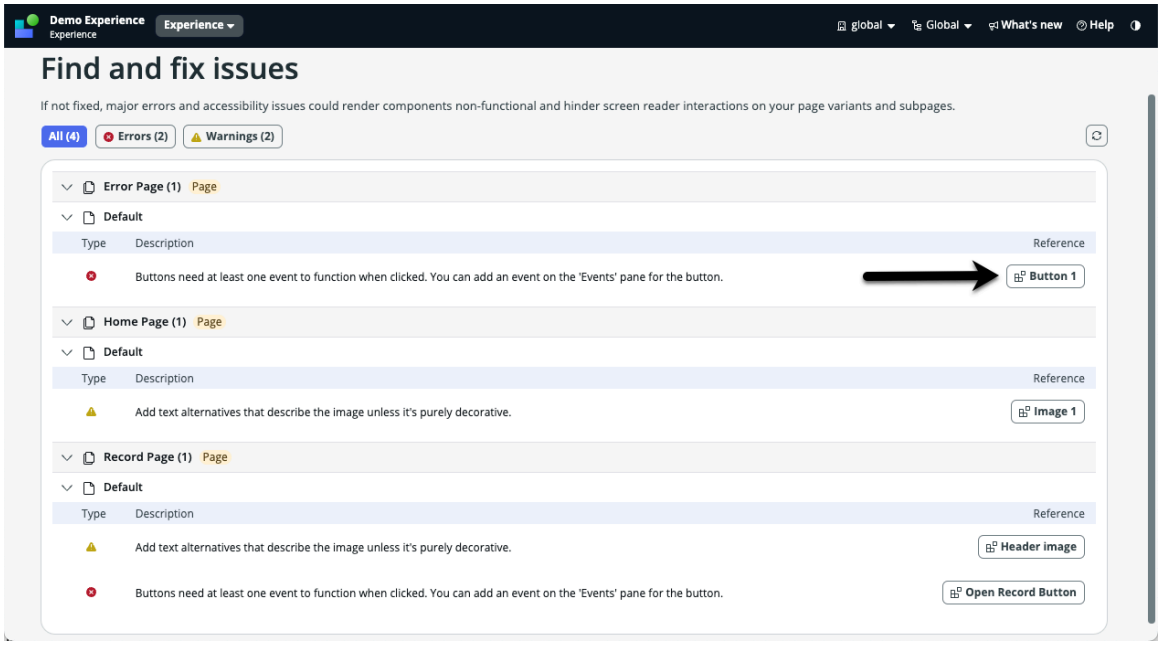
Role required: ui_builder_admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open the experience you want to check for issues by selecting **Experiences**.
3. Select **See more** under the **Find and fix issues** section.

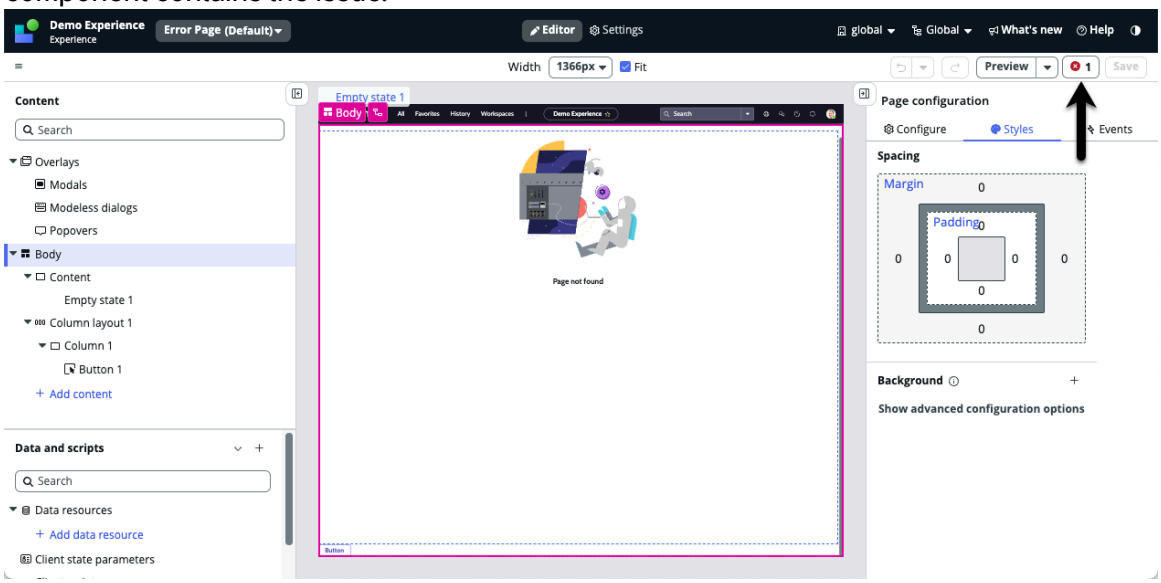


4. Fix the issues found in your experience by selecting button under the **Reference** section.

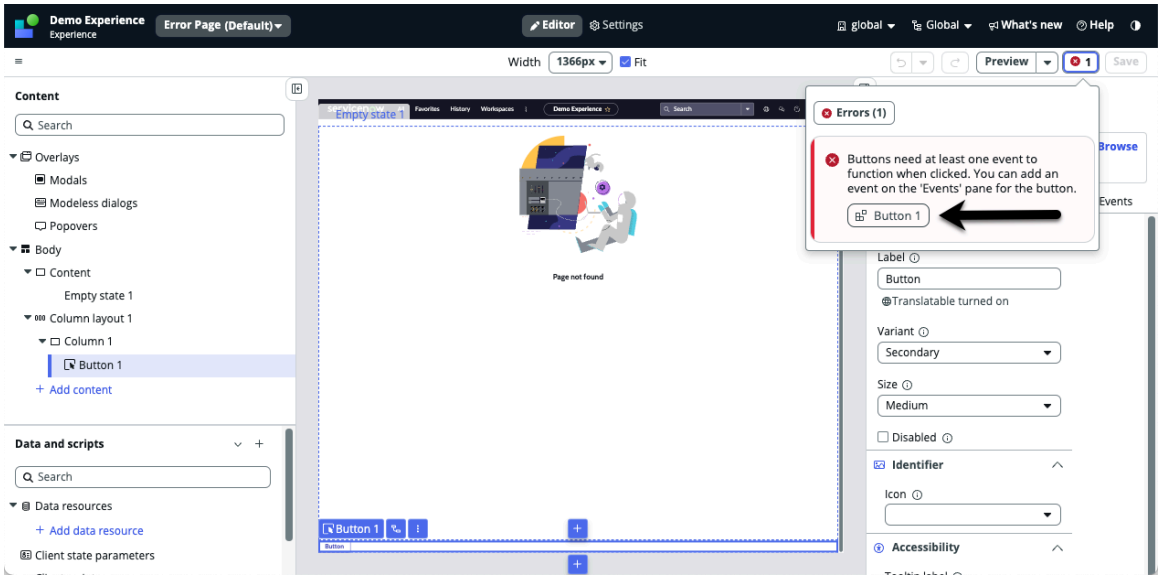


A new tab will open to the page with the issue.

5. Select the error button in the header to get a brief description of the issue and which component contains the issue.



The component name will be listed in the error or warning.



6. Select the component with the issue and follow the instructions listed in the warning or error text.
7. Select **Save**.

Advanced UI Builder

UI Builder can be used by application developers with a variety of skill levels. We recommend developers with a high level of experience, sometimes referred to as pro-coders, perform the procedures in this section.

Configure components and repeaters (advanced feature)

You can add and configure components as you build pages in UI Builder. You can also use repeaters to repeat components or multiple components with results from a data resource.

Add and configure components

Learn how to add components to your page in UI Builder. A page is built by adding components.

Before you begin

Role required: ui_builder_admin

About this task

Components are the building blocks used to create custom pages in UI Builder. Learn how to add a component to a page. After you add a component to a page, you must configure the component. For more information about configuring components, see Component documentation on the ServiceNow Developer Site.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Open or create a page.
   If you open an existing page, make sure you are in the same scope as the original page. If not, change the scope before you start editing the page. Application scoping protects applications by identifying and restricting access to application files and data. Administrators set the scope to specify what parts of an application are accessible to other applications. Application scope

protects data and application files. See Learn about security and roles for more information about application scope.

4. Select an existing container or create a column layout.
   See Organize components in UI Builder pages for more information.
5. Add components to your page.
   UI Builder comes with a library of components to choose from. You add components as the building blocks of your page. For example, you can add a heading, data visualizations, and so on. The following table shows you the different ways you can add a component to a page.
6. Now configure the properties of the components that you just added.
   Configuring components means to customize them to your needs. For more information about configuring components, see Next Experience Components.

   a. Select the component that you want to configure.

   b. Select the **Configure** tab from the configuration panel in UI Builder.

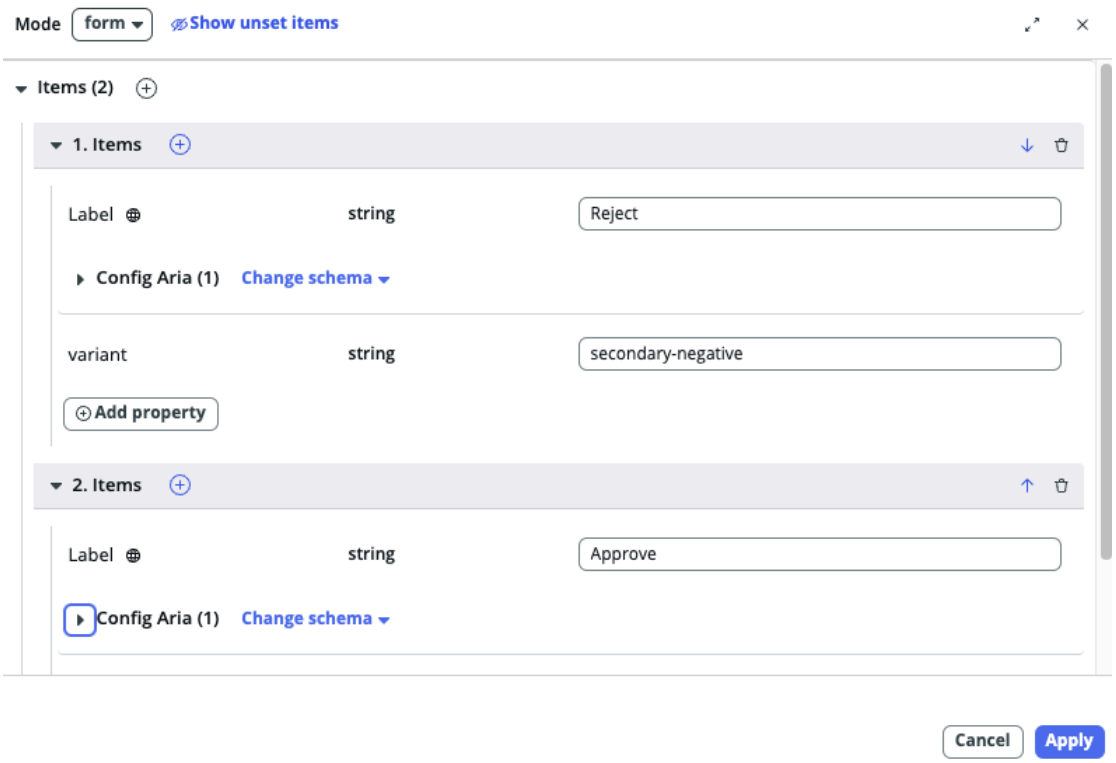
   c. Customize the component properties for the component.
   For example, you could add a name for a button component. Some components, like data visualizations, require a data source before you can configure the properties. Each component has different configuration properties based on the requirements and options for each component. For example, the **Button** component configuration is simple, while a **List** component requires more configuration.

   d. For components with configurable JSON properties, you can use UI Builder's low-code JSON editor to edit component properties without needing to edit JSON code.

   The JSON editor UI displays all available properties, even properties not defined in the dummy data or defined values. You can update properties or add your own custom properties within the JSON editor. For more information about configuring components, see Edit code with the Now Code Editor (advanced feature).

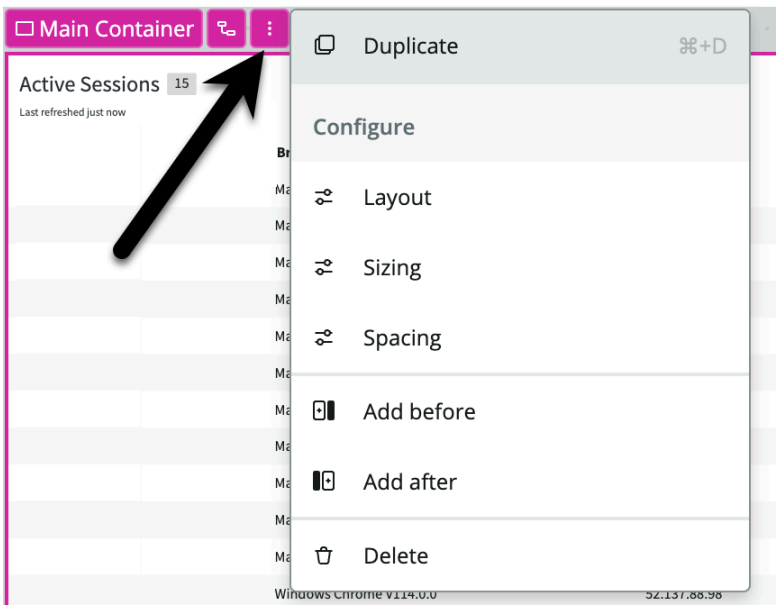
   Note:
   The low-code JSON editor is only available to properties that have a schema and whose JSON input matches said schema.

   The low-code JSON editor supports simple objects, simple array properties, complex arrays, arrays of objects, and partially supports complex objects. For complex arrays, you can add, delete, or move the position of array items. You can also select **Hide unset items** to hide objects with empty or null values for a simplified editing experience.

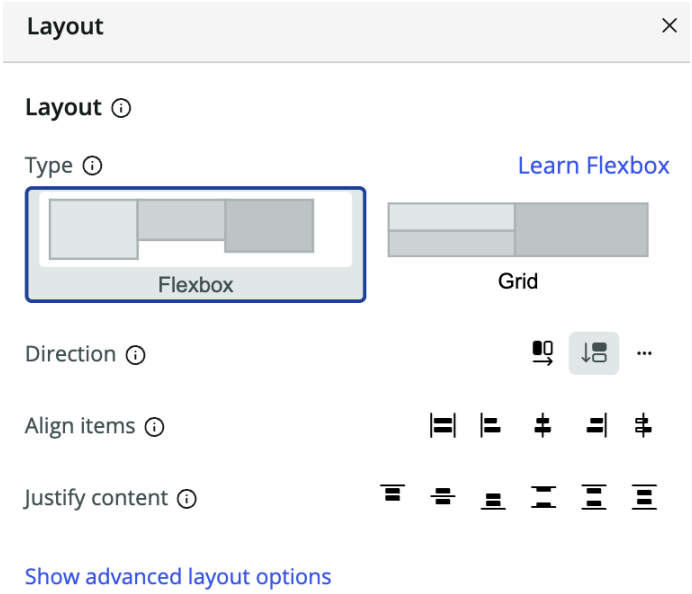


e. When working with a container component, you can edit some styles by using the floating panels.

Select the **Menu** icon to view a menu.



Select the **Layout**, **Sizing**, or **Spacing** option to open a floating panel with the most common options displayed. Drag the floating panel to another location as needed. Only one floating panel can be open at a time.



For more information about configuring components, see Next Experience Components.

7. **Optional:** Select the **Events** tab to add an event handler to your component.
   Add an event handler to add actions to the components on your page. For example, a button component is static and doesn't do anything until you bind an event action to it, such as saving a record. Some components don't have an event action applied to them, such as a heading component. But many components require you to map an event to your component to actually make it perform an action. See Bind an event to a component for more information on how to add event handlers to your component.
8. **Optional:** You can override any styles for a component by adding CSS styling under the **Styles** tab.
   For more information, see Change the default appearance of components.

   Note:
   Style changes only affect a single component at a time. To change the visual style of all the components in your experience, you must apply a theme to your experience. For more information, see Manage the visual style of UI Builder experiences.

9. **Optional:** Add additional containers to your page to display your components in an organized way.
   For example, one container could have a heading component. Another container below it could include a list component, a button component, and so on.
   a. Select the **+** at the top of your container to add a container component before the existing component and select the **+** at the bottom of a component to insert a container component after it.
   b. Drag a container component from the Components list to your existing container on the page; Hold over the top line of the container to insert the new container before the existing container or over the bottom line of the existing container to add it after.
10. **Optional:** Add more components to your page by selecting the **+** on the top or bottom of the components on the page (the **+** changes to **+ Add**).

a. Select the **+** at the top of your component to add a component before the existing component or select the **+** at the bottom of a component to insert a component after it.

b. Drag a component from the component list to insert the component before or after an existing component.

11. **Optional:** To make a modal appear when you select a component such as a button, you must add the modal to the component first.
A modal is a confirmation pop-up that appears when you select the component. For example, if you add a button component that deletes a record, you add a modal to ask the user to confirm they want to delete the record. See Create modals in UI Builder for more information.

12. Select **Save** often on your page as you work.

13. View and test your page by selecting the **Preview** button.

What to do next

You have added and configured components on your page. See Dynamically expose data in UI Builder pages (advanced feature) for more information. A data resource in UI Builder is the data that a page fetches to display content in components. Components use data resources as a way to reuse data and configurations across different experiences, and make the components dynamic on a page.

Supported functions in the UI Builder component formula editor

Learn about the various functions supported in the UI Builder component formula editor.

The following table lists the functions you can use in the UI Builder component formula editor. For more information about the component formula editor, see Customize UI Builder pages using components.

Operators available for condition builders

Operator label	Example condition	Equivalent query operator	Example query	Example output
All empty	[Short description] [ALL_EMPTY]	ALL_EMPTY	short_descriptionALL_EMPTY	All records in which there is no value in the **Short description** field.
All equal	[Short description] [ALL_EQ] [Network storage unavailable]	ALL_EQ	short_description=Network storage unavailable	All records in which the value for the **Short description** field is equal to "Network storage is unavailable."
All greater than	[Impact][ALL_GT] [2 - Medium]	>	impact>2	All records in which the **Impact**

Operators available for condition builders (continued)

Operator label	Example condition	Equivalent query operator	Example query	Example output
				field has a value of **3 - Low**
All greater than or equal	[Impact][ALL_GTE][2 - Medium]	>=	impact>=2	All records in which the **Impact** field has a value of **2 - Medium** or **3 - Low**.
All less than	[Reassignment count][ALL_LT][2]	<	reassignment_count<2	All records in which the value in the **Reassignment count** field is any number less than (but not equal to) **2**.
All less than or equal	[Reassignment count][ALL_LTE][2]	<=	reassignment_count<=2	All records in which the value in the **Reassignment count** field is one of the following: • a number less than 2 • 2
All not equal to	[Impact][ALL_NEQ][1 - High]	!=	impact!=1	All records in which the value in the **Impact** field is anything but **1 - High**.
All not empty	[Impact][ALL_NOTEMPTY]	ALL_NOTEMPTY	impactALL_NOTEMPTY	All records in which the **Impact** field has any value.
All not one of	[Impact][ALL_NOTONEOF][1 - High, 2 - Medium]	ALL_NOTONEOF	impactALL_NOTONEOF1,2	All records in which the **Impact** field is populated by anything except the following values: • **1 - High** • **2 - Medium**

Operators available for condition builders (continued)

Operator label	Example condition	Equivalent query operator	Example query	Example output
All one of	[Impact] [ALL_ONEOF][1 - High, 2 - Medium]	ALL_ONEOF	impactALL_ONEOF1,2	All records in which the **Impact** field is populated by one of the following values: • **1 - High** • **2 - Medium**
Any empty	[Short description] [ANY_EMPTY]	ANY_EMPTY	short_descriptionANY_EMPTY	Any record in which there is no value in the **Short description** field.
Any equal	[Short description] [ANY_EQ] [Network storage unavailable]	ANY_EQ	short_description=Network storage unavailable	Any record in which the value for the **Short description** field is equal to "Network storage is unavailable."
Any greater than	[Impact][ANY_GT] [2 - Medium]	ANY_GT	impactANY_GT2	Any records in which the **Impact** field has a value of **3 - Low**
Any greater than or equal	[Impact] [ANY_GTE][2 - Medium]	ANY_GTE	impactANY_GTE2	Any record in which the **Impact** field has a value of **2 - Medium** or **3 - Low**.
Any less than	[Reassignment count][ANY_LT] [2]	ANY_LT	reassignment_countANY_LT2	Any record in which the value in the **Reassignment count** field is any number less than (but not equal to) **2**.
Any less than or equal	[Reassignment count][ANY_LTE] [2]	ANY_LTE	reassignment_countANY_LTE2	Any record in which the value in the **Reassignment count** field is one of the following: • a number less than 2 • 2

Operators available for condition builders (continued)

Operator label	Example condition	Equivalent query operator	Example query	Example output
Any not equal	[Impact] [ANY_NEQ][1 - High]	ANY_NEQ	impactANY_NEQ1	Any record in which the value in the **Impact** field is anything but **1 - High**.
Any not empty	[Impact] [ANY_NOTEMPTY]	ANY_NOTEMPTY	impactANY_NOTEMPTY	Any record in which the **Impact** field has any value.
Any not one of	[Impact] [ANY_NOTONEOF] [1 - High, 2 - Medium]	ANY_NOTONEOF	impactANY_NOTONEOF1,2	Any record in which the **Impact** field is populated by anything except the following values: • **1 - High** • **2 - Medium**
Any one of	[Impact] [ANY_ONEOF][1 - High, 2 - Medium]	ANY_ONEOF	impactANY_ONEOF1,2	Any record in which the **Impact** field is populated by one of the following values: • **1 - High** • **2 - Medium**
CONCAT	CONCAT(value)	CONCAT	CONCAT("Welcome, ",@context.session.user.firstName)	Create a new string that combines all supplied strings into one
EMPTY	EMPTY(value)	EMPTY	EMPTY(@context.session.user.roles)	Returns true if value is null or undefined
IF	IF(if, then, else)	IF	IF(@context.props.bare,"bare page","not bare page")	If condition is true, return the value then. If condition is false, return the value else.
LEN	LEN(list)	LEN	LEN([1,2,3])	Returns the number of items in the array

Operators available for condition builders (continued)

Operator label	Example condition	Equivalent query operator	Example query	Example output
Pick	PICK(array, field)	Pick	PICK(@context.session.user_preferences,"name")	Creates a new array where each item is picked from field in each item in the array. If the field does not exist, the item in the new array will be EMPTY
Range	RANGE(from, to)	Range	RANGE(1,10)	Creates an array of numbers, starting with from, up to, incrementing by step
Sum	SUM(array)	Sum	SUM([1,2,3])	Starting at 0, add the number value of each item in the array and return the resulting summation
Translate	TRANSLATE(text)	Translate	TRANSLATE("Welcome back")	Returns the string from the first argument after the characters specified in the second argument are translated into the characters specified in the third argument.
Where empty	[Short description] [WHERE_EMPTY]	WHERE_EMPTY	short_descriptionWHERE_EMPTY	Extract records where there is no value in the **Short description** field.
Where equal	[Short description] [WHERE_EQ] [Network storage unavailable]	WHERE_EQ	short_descriptionWHERE_EQNetwork storage unavailable	Extract records where the **Short description** field is equal to "Network storage is unavailable."
Where greater than	[Impact] [WHERE_GT][2 - Medium]	WHERE_GT	impactWHERE_GT2	Extract records where the **Impact** field has a value of **3 - Low**

Operators available for condition builders (continued)

Operator label	Example condition	Equivalent query operator	Example query	Example output
Where greater than or equal	[Impact] [WHERE_GTE][2 - Medium]	WHERE_GTE	impactWHERE_GTE2	Extract records where the **Impact** field has a value of **2 - Medium** or **3 - Low**.
Where less than	[Reassignment count] [WHERE_LT][2]	WHERE_LT	reassignment_countWHERE_LT2	Extract records where the value in the **Reassignment count** field is any number less than (but not equal to) **2**.
Where less than or equal	[Reassignment count] [WHERE_LTE][2]	WHERE_LTE	reassignment_countWHERE_LTE2	Extract records where the value in the **Reassignment count** field is one of the following: • a number less than 2 • 2
Where not equal	[Impact] [WHERE_NEQ][1 - High]	WHERE_NEQ	impactWHERE_NEQ1	Extract records where the value in the **Impact** field is anything but **1 - High**.
Where not empty	[Impact] [WHERE_NOTEMPTY]	WHERE_NOTEMPTY	impactWHERE_NOTEMPTY	Extract records where the **Impact** field has any value.
Where not one of	[Impact] [WHERE_NOTONEOF] [1 - High, 2 - Medium]	WHERE_NOTONEOF	impactWHERE_NOTONEOF1,2	Extract records where the **Impact** field is populated by anything except the following values: • **1 - High** • **2 - Medium**
Where one of	[Impact] [WHERE_ONEOF] [1 - High, 2 - Medium]	WHERE_ONEOF	impactWHERE_ONEOF1,2	Extract records where the **Impact** field is populated by one of the following values:

Operators available for condition builders (continued)

Operator label	Example condition	Equivalent query operator	Example query	Example output
				• 1 - High • 2 - Medium

Add repeaters to components

In UI Builder, use repeaters to repeat one or more components with results from a data resource.

Before you begin

Role required: ui_builder_admin

About this task

The repeater component acts as a basic loop that repeats the data you provide in multiple components within UI Builder. Repeaters use an array or an array of objects. For example, the array `[{"task": "A"},{"task": "B"}]` repeats the content inside it two times. Repeaters enable you to bind values to a data array property. `@item.value.{property_name}` binds the values to the component inside the repeaters. If you want to bind a task, you can bind it as `@item.value.task`, and the repeater displays the correct value.

Components inside a repeater repeat the number of items in the data source, regardless of whether the component output is made dynamic or not. For example, say that you place a Header component inside a repeater with three data elements but don't change the Label field. Then you see three instances of the Header with the same output Label.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Open the editor for the page where you want to add the repeater.
   If you haven't created a page for your experience, see Create a page in UI Builder for more information on how to create a page.
4. Connect a data resource to your page.

   For example, add the **Look up multiple records** data resource to your page. See Add and configure data resources to a page for more information on connecting a data resource.

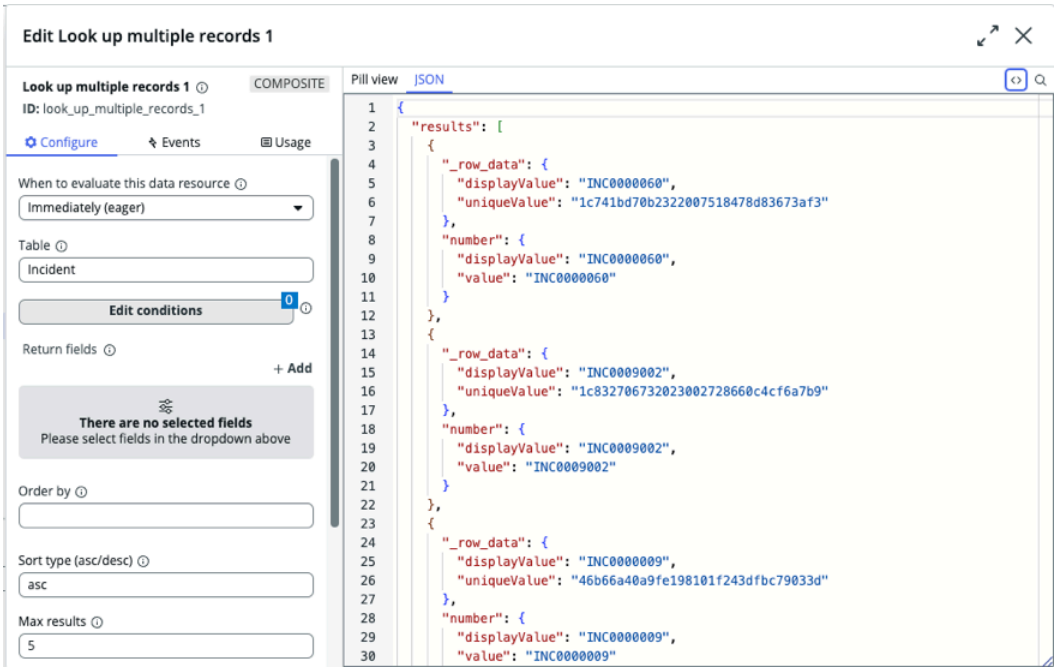
   a. In the data resource drawer, select **+ Add data resource**.

   b. In the modal, search for the **Look up multiple records** data resource and select **Add**.

   c. In the **Table** field, enter `Incident`.

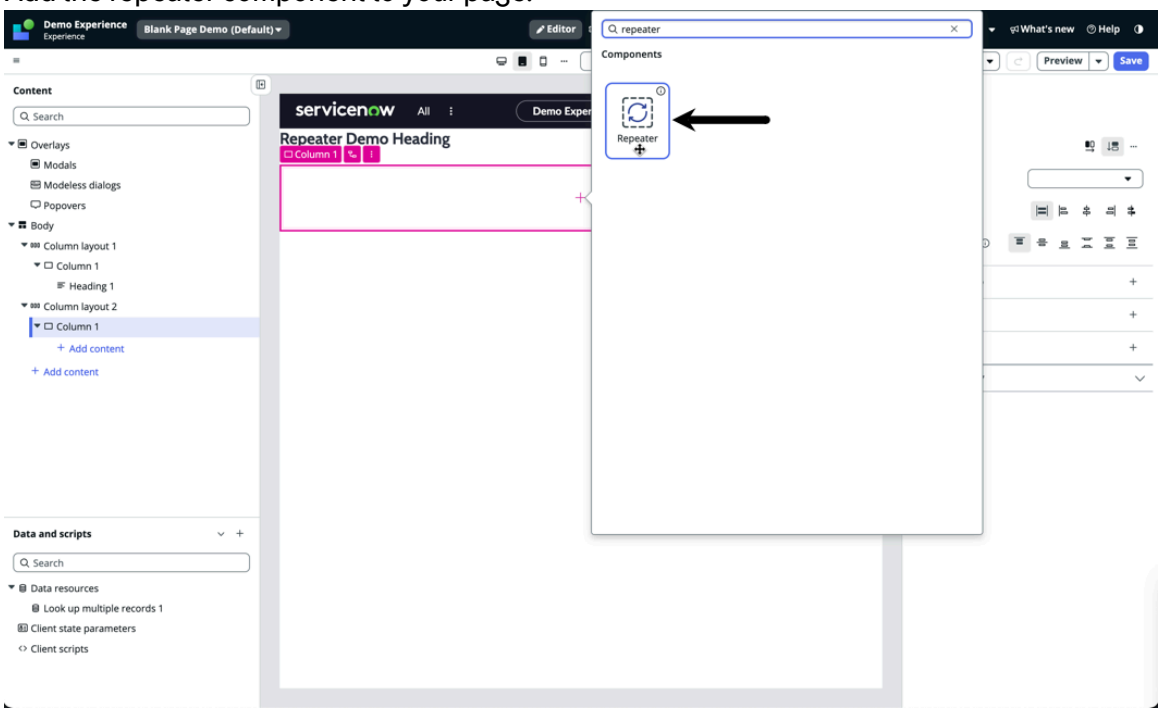
   d. In the **Max results** field, enter `5`.

   The **Look up multiple records** data resource is configured.



e. Close the modal.

5. Select an existing container or create a column layout.
   See Organize components in UI Builder pages for more information.
6. Add the repeater component to your page.



For information about adding components, see Add and configure components.

7. In the **Configure** tab, select **None - Configure the component manually**.
8. In the **Configure** tab, add an array that defines the data for repeated components.

a. Point to **Data array** and select the Bind data icon ( ).

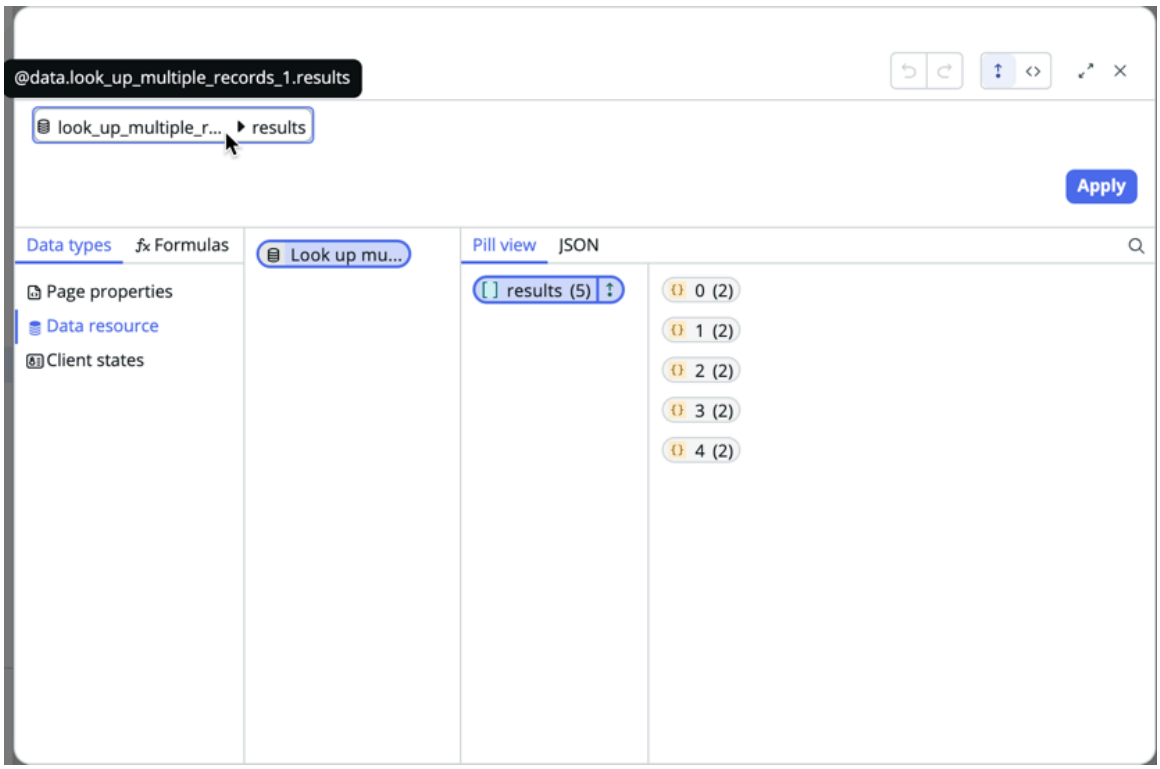
b. Under **Data types**, select **Data resource**.

c. In the next column, select the **Look up multiple records 1** pill.

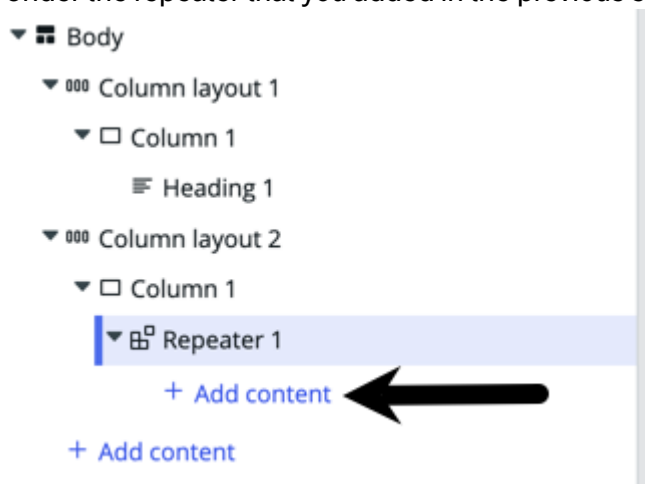
d. Under **Pill view**, double-click or drag **results** to add the **@data.look_up_multiple_records_1.results** output.

e. Select **Apply**.

The results from the data resource are bound to the repeater component. Within the repeater, one component represents each result.



9. Under the repeater that you added in the previous step, select **+ Add content**.



10. Add a **Heading** component.
11. In the **Configure** tab, select **None - Configure the component manually**.
12. Configure the component you nested in the repeater.

    For example, bind the **displayValue** to the component. See Connect data to your components for more information.

    a. Move your cursor to the **Label** field and select the Bind data icon ( ).

    b. Under **Data types**, select **Repeater**.

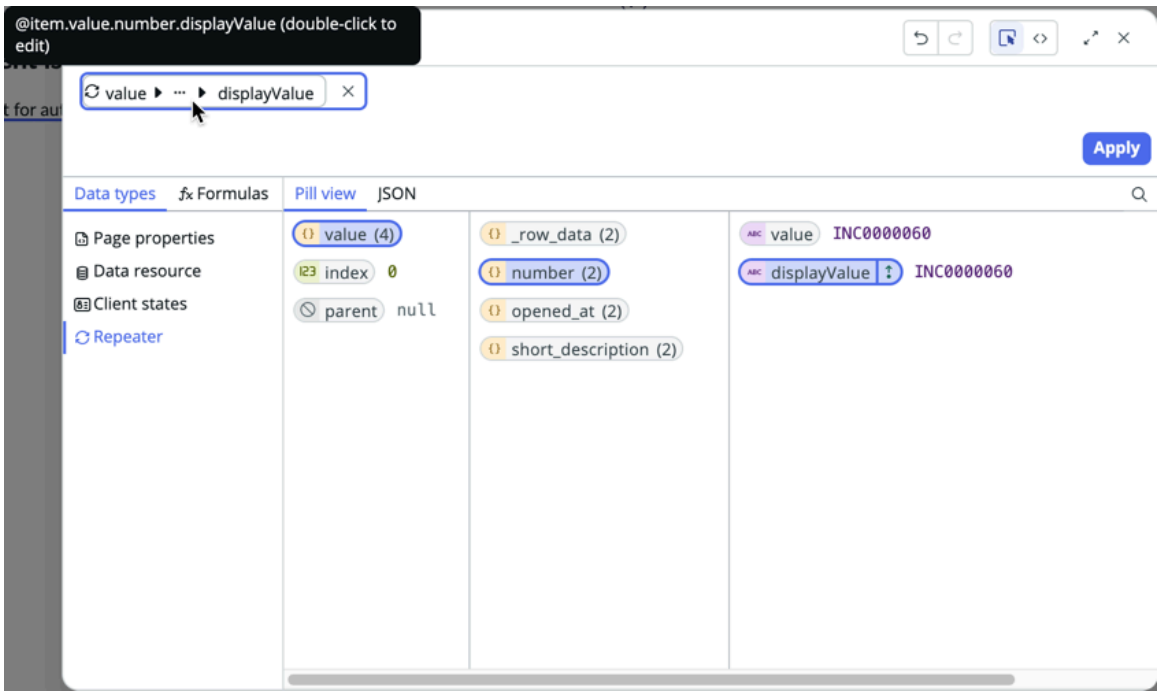
    c. Under **Pill view**, select the **value** pill.

    d. In the next column, select the **number** pill.

    e. In the next column, double-click or drag **displayValue** to add the **@item.value.number.displayValue** output.

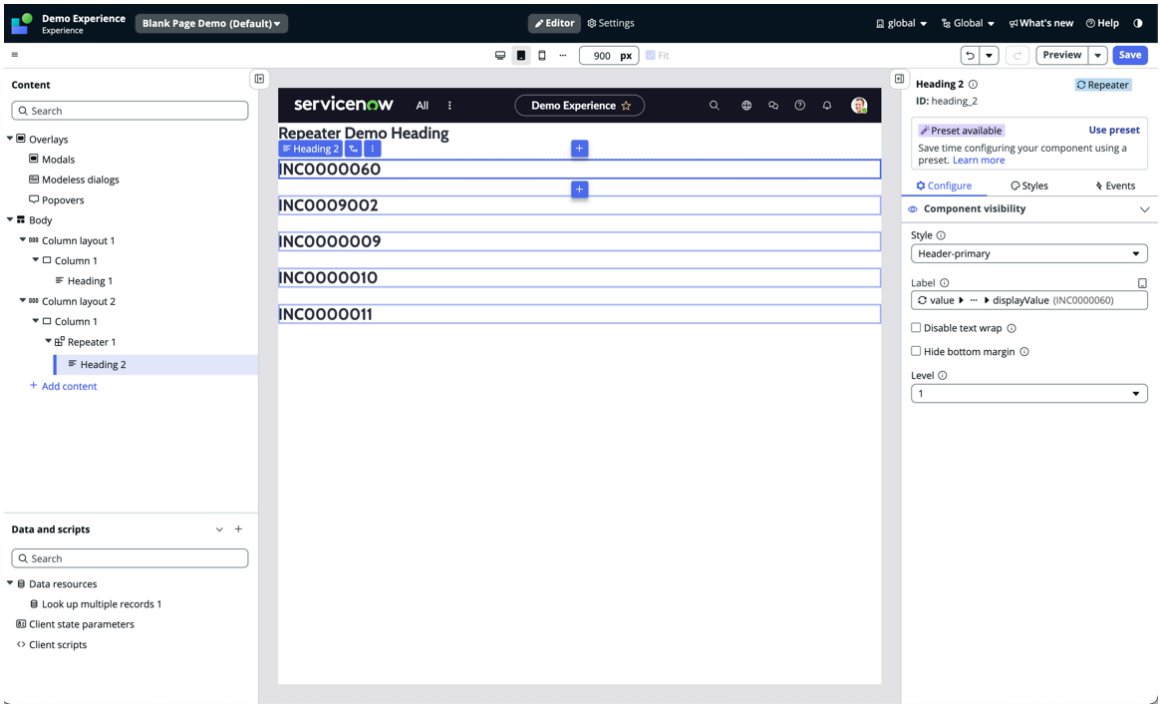
    f. Select **Apply**.

    The **displayValue** is bound to the Heading component.

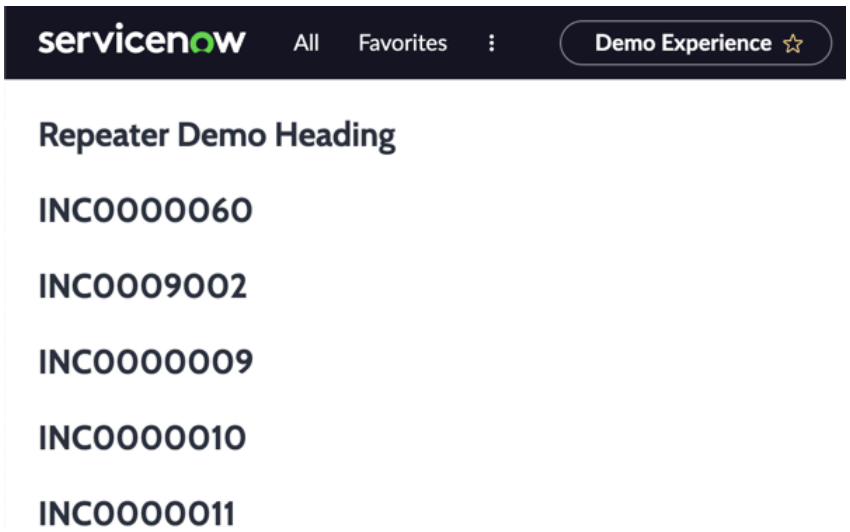


13. Select **Save**.

    The component you bound to the data resource appears on the stage and is repeated five times, once for each result returned from the Incident table.



14. View and test your page by selecting the **Preview** button.
A tab opens to display the page preview with repeated data.



Dynamically expose data in UI Builder pages (advanced feature)

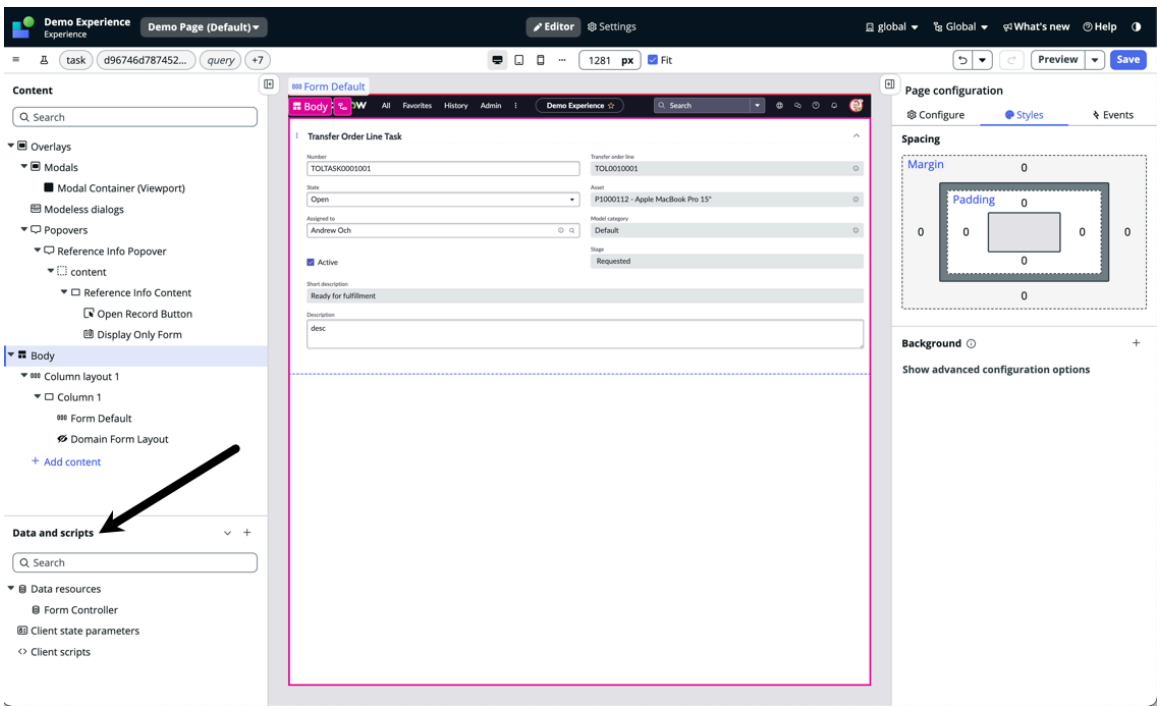
Sync data between ServiceNow tables and data with the pages you build with UI Builder. Pages display synced data in real time and update data/tables when a user inputs information.

Learning about data resources in UI Builder

UI Builder syncs ServiceNow tables/data using data resources. A data resource fetches the data that UI Builder uses to display information in a component. UI Builder components use data resources to sync data across different experiences. Data resources make the data in components dynamic, which means that you don't have to recreate data for every page.

Data resources are found in the data resource drawer. The data resource drawer is where you can add and configure a data resource for your page. After you configure the data resource, you can sync the data between components on your page and ServiceNow tables/data.

Data Resource Drawer



The data resource drawer contains three sections:

- Data resources: The data resources that are part of the experience
- Configuration/Events: Configured data resources and events for the experience
- Preview: JSON for the information returned by the data resource

You can bind the configuration properties for components, other data resources, client scripts, client state, and events to these data resources.

Set conditions for a filter in your data resource. For more information, see Connect data to your components.

How data resources work in UI Builder

Data resources fetch data from Glide, GraphQL, and REST APIs, then transform it for use in a component on a UI Builder page.

Components use both inherited and local data resource instances. Inherited data resources are automatically loaded into a UI Builder page, and local data resource instances can be added and configured. Data resources are evaluated based on specified input values to make sure the right data is retrieved. For more information about inherited and local data resources, see Inherited versus local data resources in UI Builder.

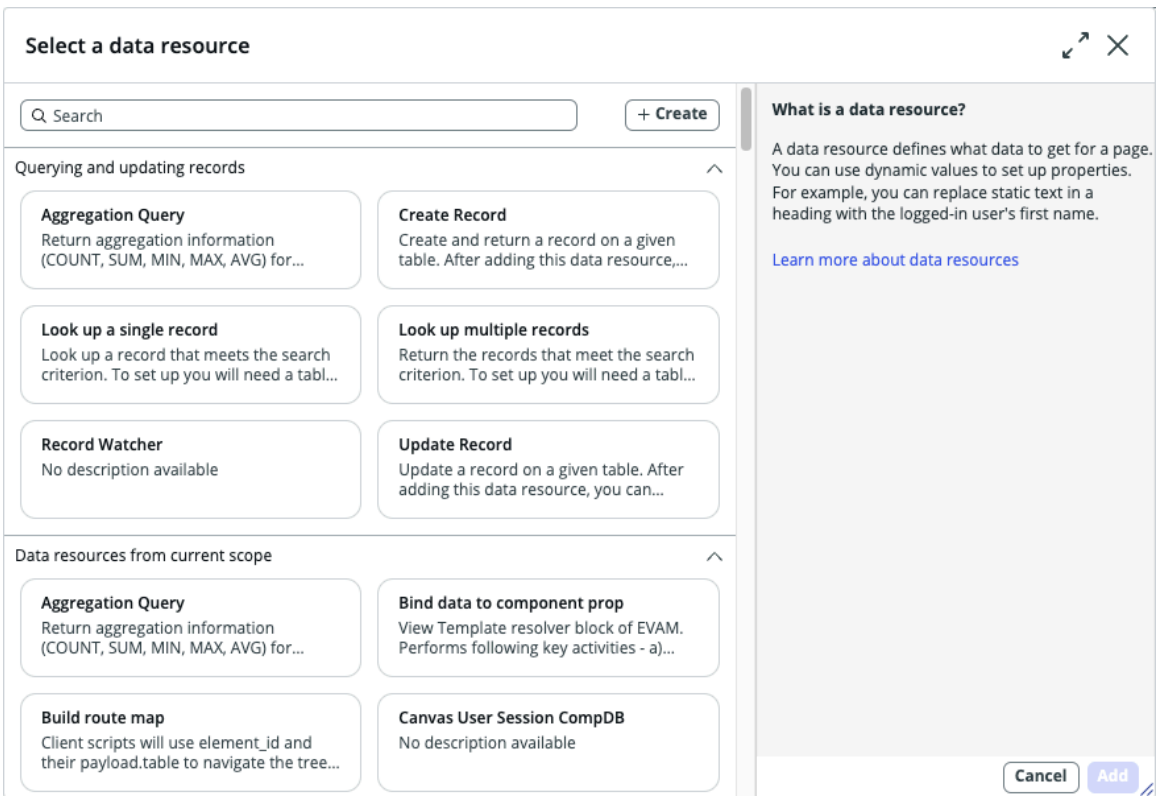
Local data resources in UI Builder

You can select local data resources, such as server data, operations, transforms, or client data, like the gForm API, to bring data to the UI Builder page.

Note:
Only one GlideForm is supported per page in UI Builder. For more information about GlideForm, see the ServiceNow Developer site.

Data resources are organized by application in the data resource drawer. They are then further categorized by the data resource type like Server data or Transform. For example, the Global application has several data resources, but the Customer Service Management (CSM) Workspace application has only a few data resources.

Data resources



You see different data resources depending on the application that you are in. If you select the Global application, you will see different data resources under Server data than if you select the CSM Configurable Workspace application.

Using data binding in UI Builder

Data binding enables you to create dynamic pages by syncing pages and components to data resources. You can bind data to a component to retrieve data from the back end, and use field parameters to get properties from the URL. Changing the URL parameters enables you to create dynamic pages that show different data depending on the parameters.

You can bind data to a component in the following ways:

Context binding

Use URL parameters to connect parts of the URL with your page's properties. For example, you can link the table name from the URL to your component by using the @context syntax, like this: `@context.props.table`.

Imagine you have a UI Builder page with a required field named `table`. The URL for your page might look like /demo/page/<table-name>. The <table-name> could be something like `incident`. The data can also come from parent data resources or be local properties specific to the page.

To link your component properties, other data resource properties, or event details to the page property, you use a `@context.props.table` binding. Be sure to either provide a test value in the URL or set a fixed value for that property in your page's configuration if you are using context binding.

Data resource binding

Use data resources to fetch data from the back end of your instance, such as Client state, GraphQL, or a REST API. These data sources have properties that can be linked to elements on a UI Builder page.

For instance, if you are using the Lookup Record data resource, you can utilize it in a button component. You might use a data expression in the "label" property like this: `@data.lookup_record_1.result.number.displayValue`.

Component binding

Use component binding to connect one component to another. Let's say that you have a List Menu component on the UI Builder page. The List Menu reveals the currently chosen list to other components on the same page. These other components can access the data by linking to it using an expression such as `@elements.list_menu_1.selectedListId`.

Client state parameter binding

Use client state parameter binding to connect and synchronize data between a client-side application and UI Builder components. Parameter binding allows the client state to automatically update data in components, and vice versa. Use `@state` syntax to bind a state property to a client state parameter.

Types of data sources available in UI Builder

You can use the following data resource types that are shown in the table.

Data resource types

Data resource type	Description
Controller	Encapsulates data and event logic and enables presets for components.

Data resource types (continued)

Data resource type	Description
GraphQL	GraphQL queries and mutations that are executed.
Transform	Script that transforms the input data into another format.
Client state	Client-side data resources that include the client information, domain-specific states or logic, user preferences, and so on.
Composite	Single reusable data resource that contains multiple data resources.
REST	Data resources that are made through REST API requests.

Inherited versus local data resources in UI Builder

Inherited data resources share information from the surrounding parts of a UI Builder page such as an application. Imagine your page is in a large frame, and it gets some information from the frame or other parts around it. You can use this info by connecting it to your page's properties, kind of like linking puzzle pieces together. If you get this information from the frame, you don't have to get it again yourself.

Local data resources are items you add to a UI Builder page yourself. Imagine you're making a page for travel requests. You can sync employee data to a list component so employees can request trips linked to their own info.

To use these data resources, either bring them in from an app or create them in the ServiceNow platform. Then, in UI Builder, make these resources available for your components. Next, tie the data to your component, so it can work with it. For instance, you could have a set of records, expose it in UI Builder, and link it to a component. From there, configure the component to do things with the data, like saving new records.

Once your page is syncing data, you connect it to the part of the page that needs it such as a component. It's like making sure the right puzzle piece fits in the right spot. Then you can tell that part of the page what to do with the data. For example, you might use it to add new travel plans for employees.

UI Builder Data resource properties

When you add a data resource in UI Builder, it's like adding a tool that knows where to get information from. Data resource properties are the instructions that tell a data resource how to bring data into UI Builder pages. These properties tell a data resource which data to get, how to organize it, and what conditions to follow. For instance, you can use these properties to specify

which tables to look in, how to sort the data, and more. These properties are what make the data resource work correctly and give you the data you need.

UI Builder Data resource scripts

Data resource scripts are special instructions you give UI Builder to return specific pieces of data. For example, if you're dealing with a list of products, a script could tell a data resource to show only products that are available in stock or to arrange them in a particular order. Data resource scripts enable you to customize how UI Builder treats the information, such as adding extra rules or actions.

Add and configure data resources to a page

Add data resources to dynamically expose data from tables, records, or other elements on your page in UI Builder.

Before you begin

Role required: ui_builder_admin

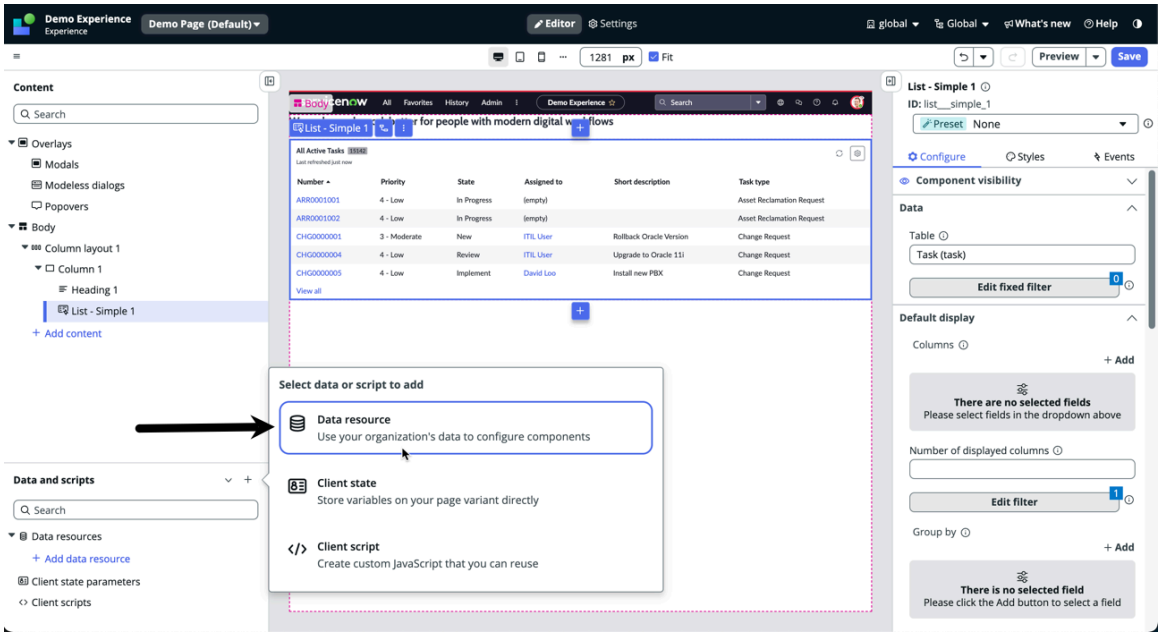
Choose the application containing your data resource. The Global application contains the most common data resources. The data resources relate to Server data, Operations, and Transform. You can add a data resource such as Look Up Records, Table route map, GlideRecord Query, and so on.

About this task

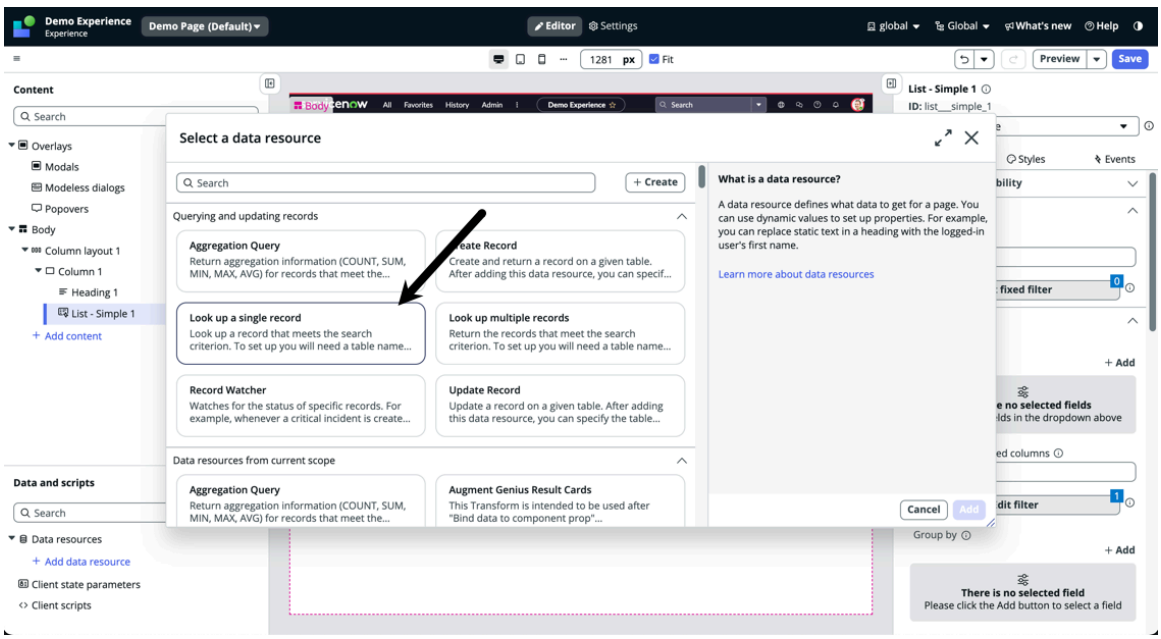
In the data resource section of UI Builder, you can add and configure local data resource instances for your page. Local data resource instances dynamically expose data to components. If you have any inherited data resources, they are listed in the inherited data resource pane and are read-only. You don’t configure inherited data in UI Builder. Data resources from any application can be added to any page unless restricted by security permissions. You can add multiple local instances of the same data resource, if you need different configurations of the same data resource on your page.

Procedure

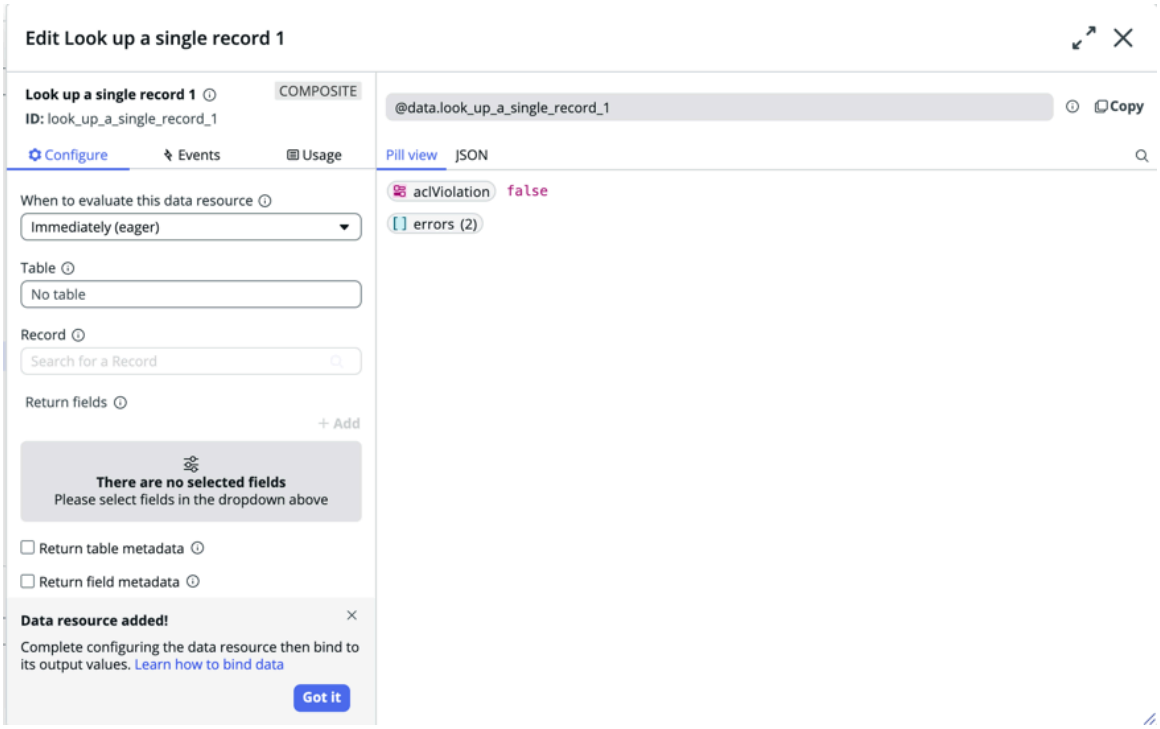
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page.
4. Add a component to your page.
   You need a component on your page before you can bind a data resource to it.
5. Select the **+** icon in the data resource drawer.
6. Select **Data resource**.



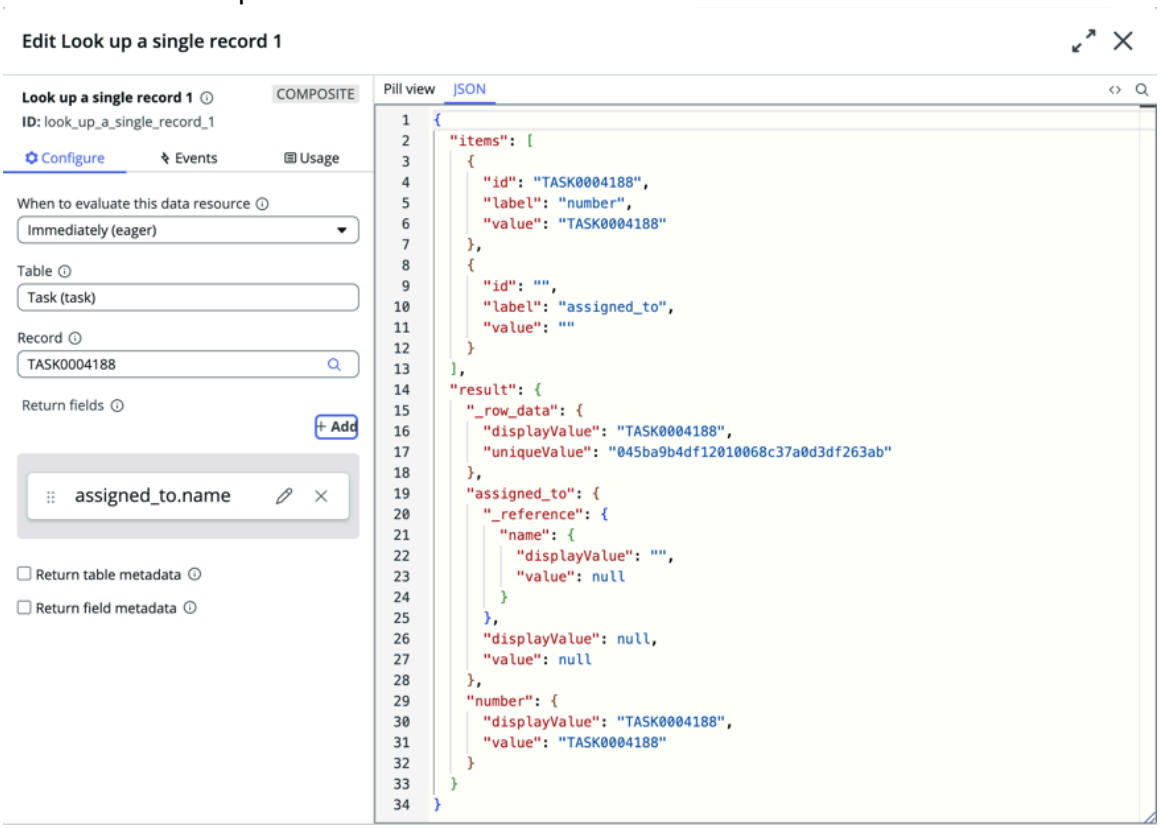
7. Select a data resource to add to your page.
For example, you could select **Look up a single record**.



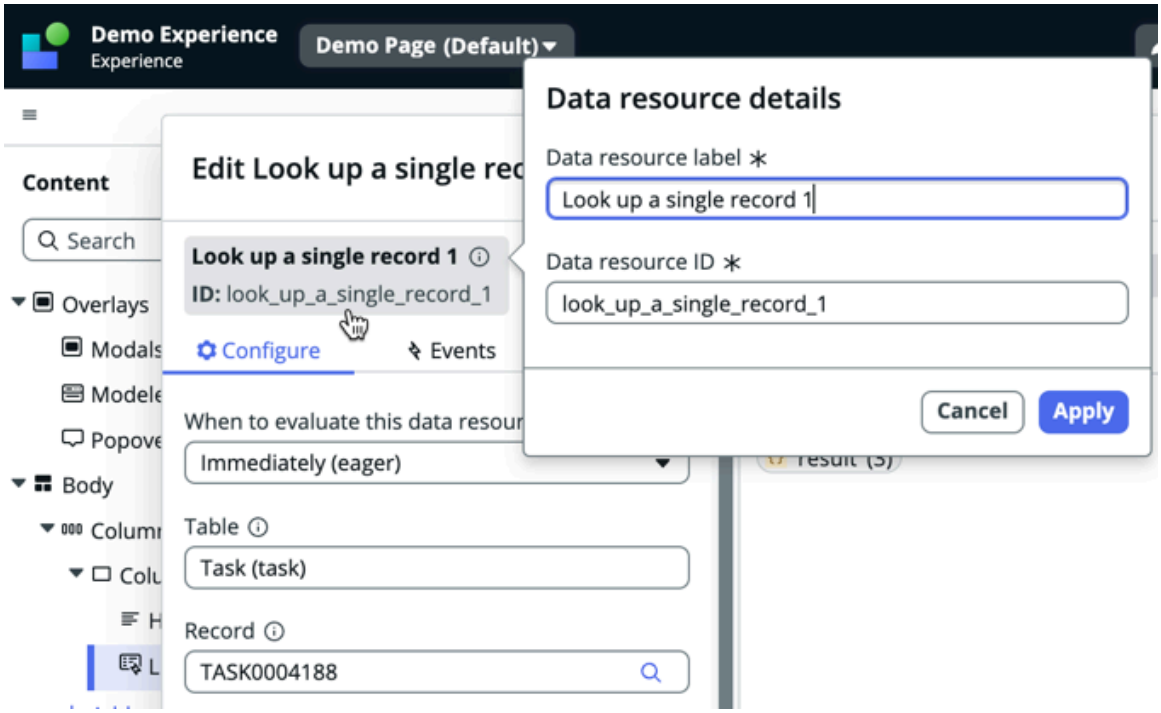
8. Select **Add** to add the data resource to your page.
The Data Resource configuration modal appears.



9. Fill in the required information for the data resource in the **Configure** tab to expose the data you want.
The configuration fields vary depending on which Data Resource instance is selected. The **Configure** panel and preview window may display errors when a Data Resource is added to a page. UI Builder attempts to evaluate Data Resources when they're added to a page. The errors remain until the required Data Resource fields have been filled.



10. Select the default label to provide a custom label and ID value.



The data resource appears in the data and scripts drawer.

11. Select **Save** in the UI Builder header.

What to do next

Now that you have a data resource connected to your page, you can bind the data to a component or bind an event to the data resource. For more information, see Connect data to your components and Bind an event to a data resource.

Bind data to UI Builder pages using controllers (advanced feature)

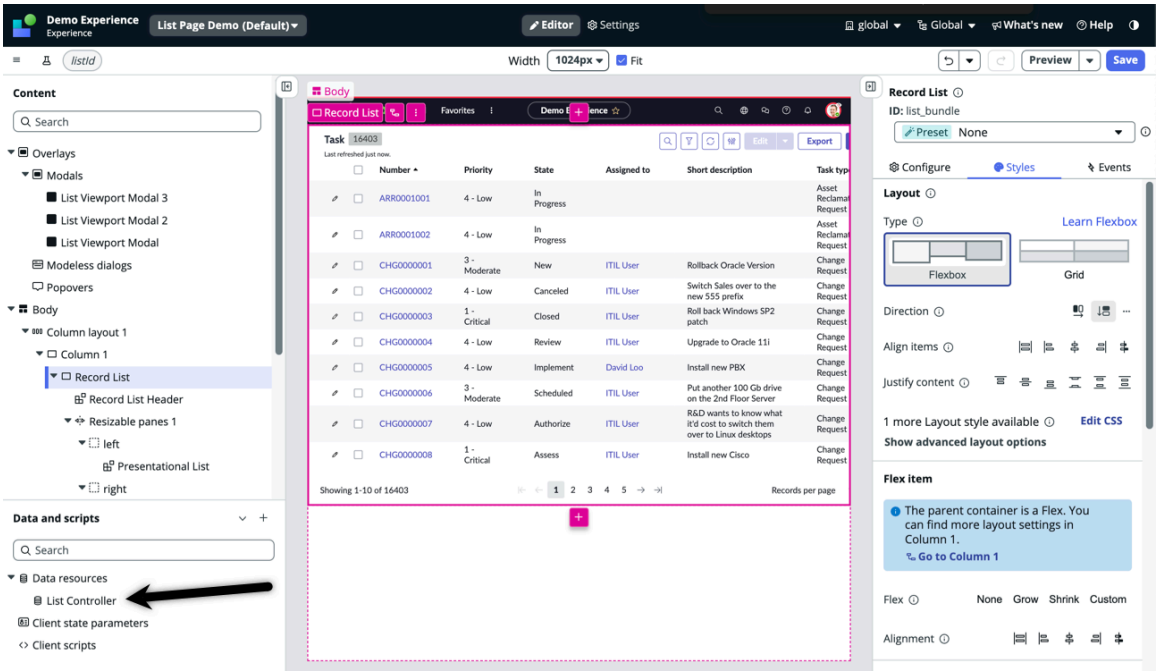
Controllers simplify the linking of data and event logic to enable component presets in UI Builder.

Controllers enable page builders to easily connect data and scripts to their pages in UI Builder. A controller is a type of data resource that component presets use to bind data to components. Controllers differ from other data resources in that they contain data and event logic that enables presets for components. A controller brings data from the server to a component, and it brings updated data back to the server based on interactions with the component. For more information about component presets, see Automatically configure components using presets.

Note:
Creating controllers is not supported in Xanadu.

Controllers are added by default when you use a UI Builder page template. You can add controllers to UI Builder pages without a controller within the data resources drawer or by selecting a component preset after adding a component to a page. You can view which controllers are configured on your page by opening the data resources drawer.

Controller in the Data resource drawer



Types of data controllers in UI Builder

Data controllers

Data controllers decide what information should be displayed on UI Builder pages. Data controllers use data resources to sync information in real time and update data/tables when a user inputs information. You can manually add data controllers to a page.

UI controllers

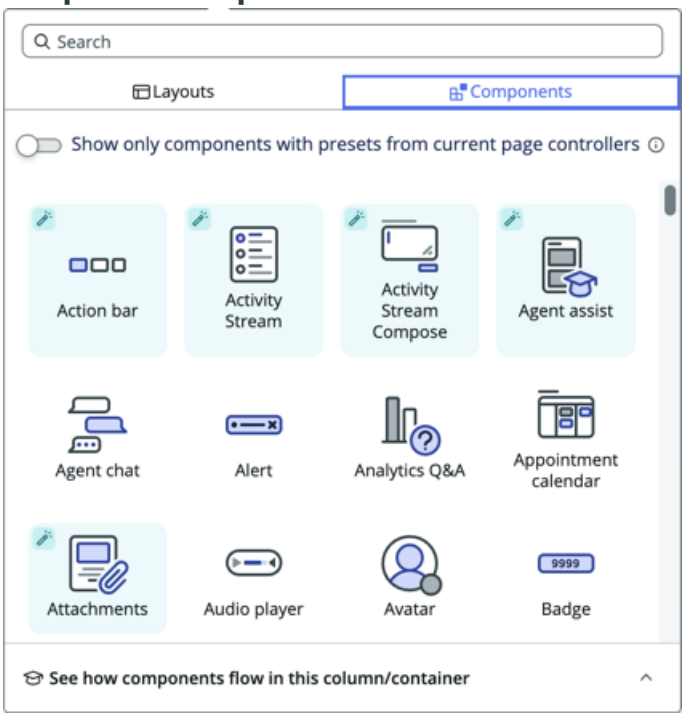
UI controllers are added to pages when using UI Builder page templates and cannot be added manually.

Using controllers with presets in UI Builder

You can add multiple controllers to a UI Builder page, but you can't use the same one twice. If you try to use a component preset that needs a controller, UI Builder prompts you to add it.

Not all components work with controllers, but you can easily see which ones do in the component library. If you have a controller configured on your UI Builder page, you can open the component library to view which components have presets. Components with presets available are highlighted in the component library.

Components with presets



Add a controller

Add a data controller to your page to use component presets.

Before you begin

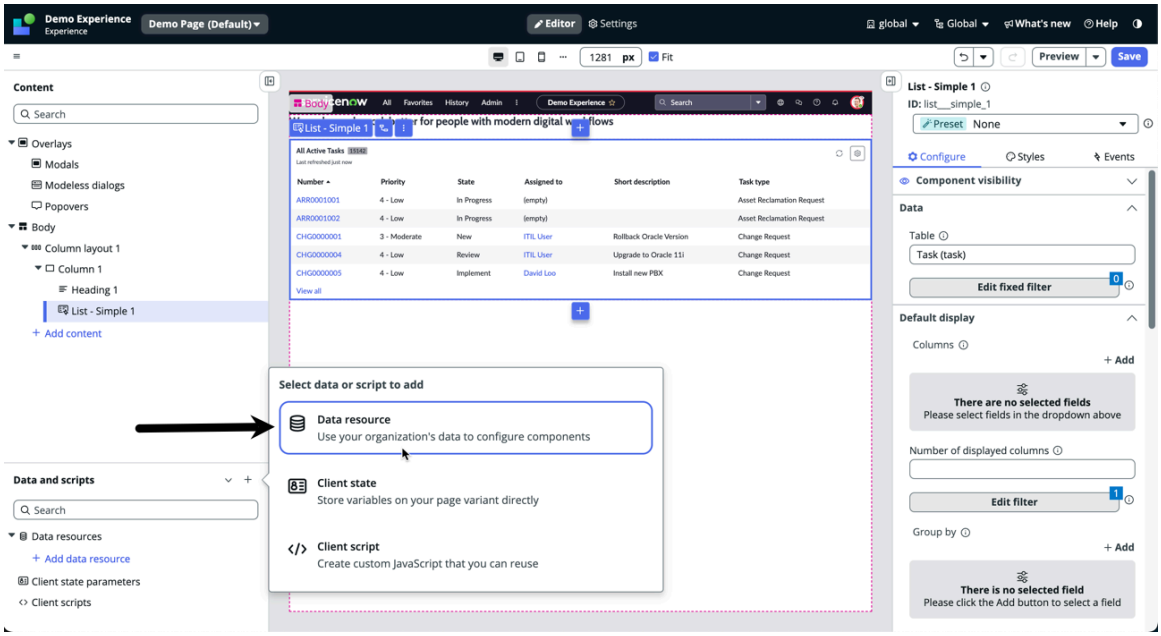
Role required: admin

About this task

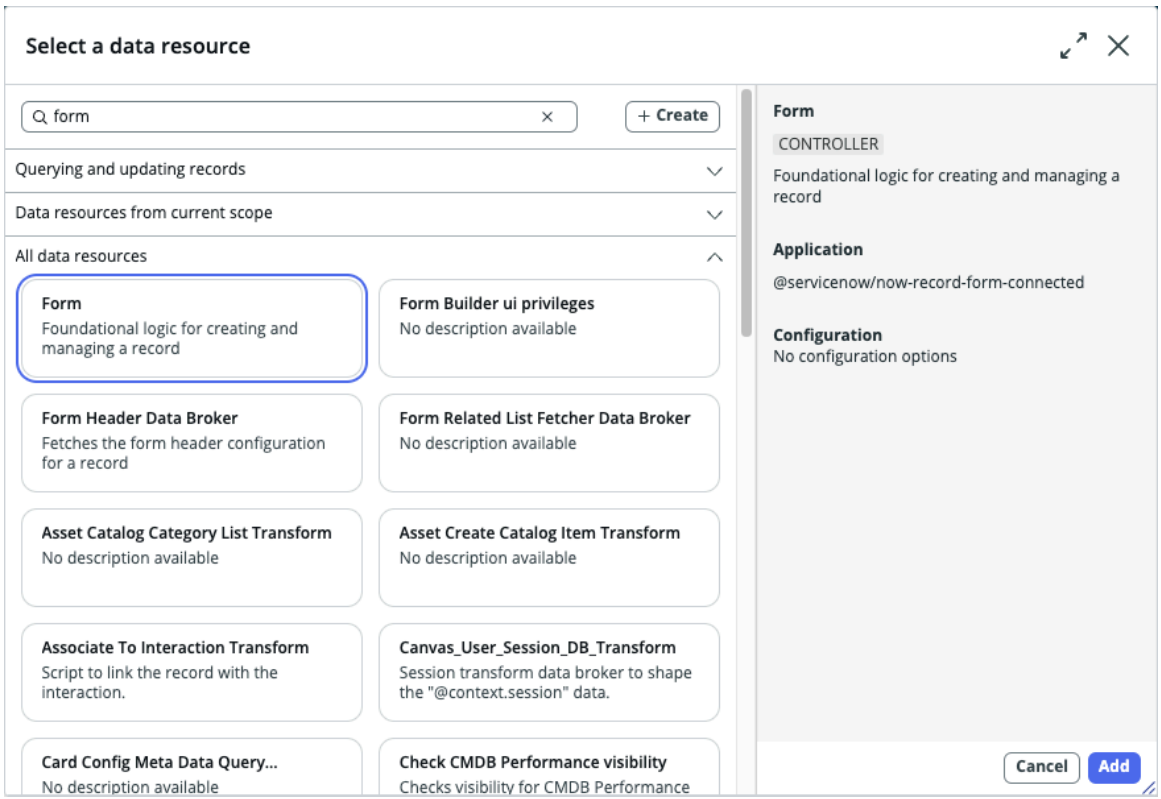
The record controller is the only controller that you can add to a page in Xanadu.

Procedure

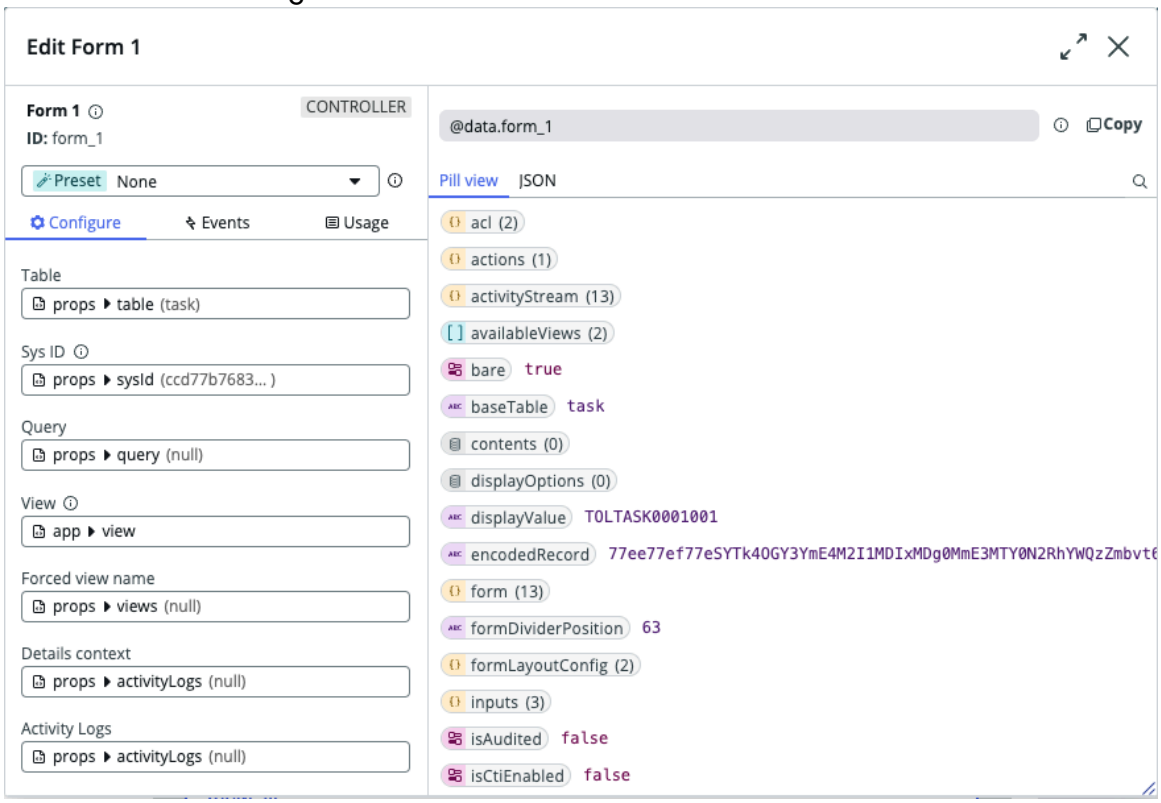
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Create or open a page or page variant.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Select the **+** icon in the data resource drawer.
5. Select **Data resource**.



6. Enter `form` in the search field.
7. Select the controller you would like to add to your page.
8. Select **Add**.

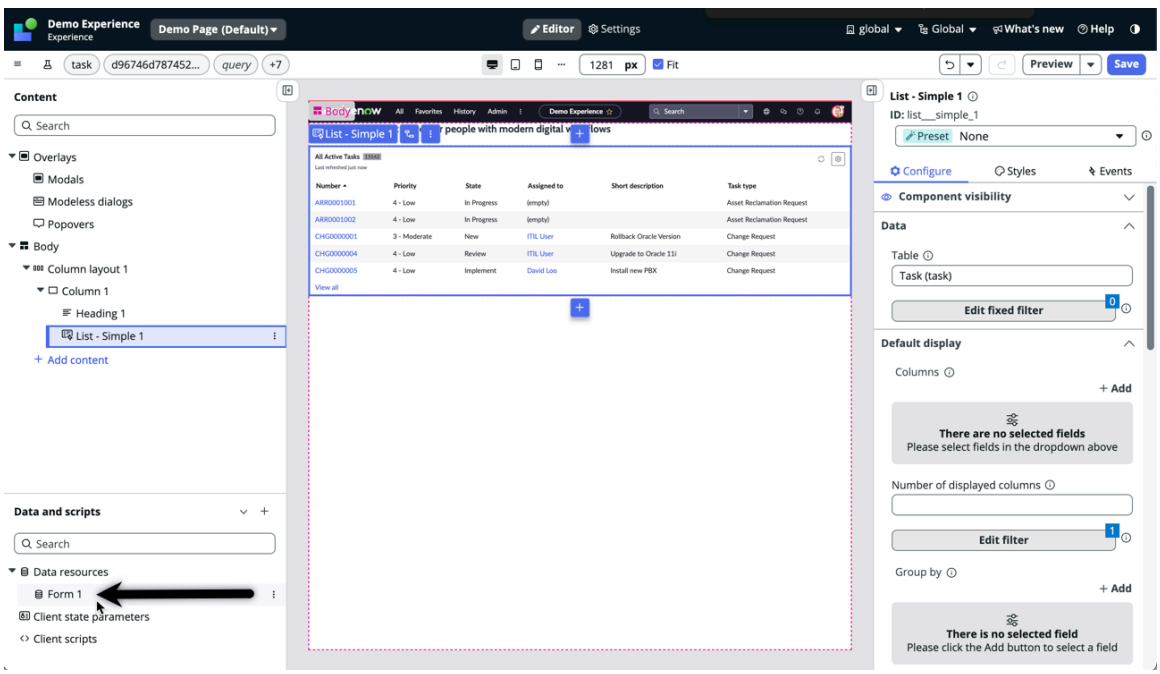


9. Fill in the fields to configure the controller.



Result

The form controller displays in the **Data resources** section.



Edit a controller

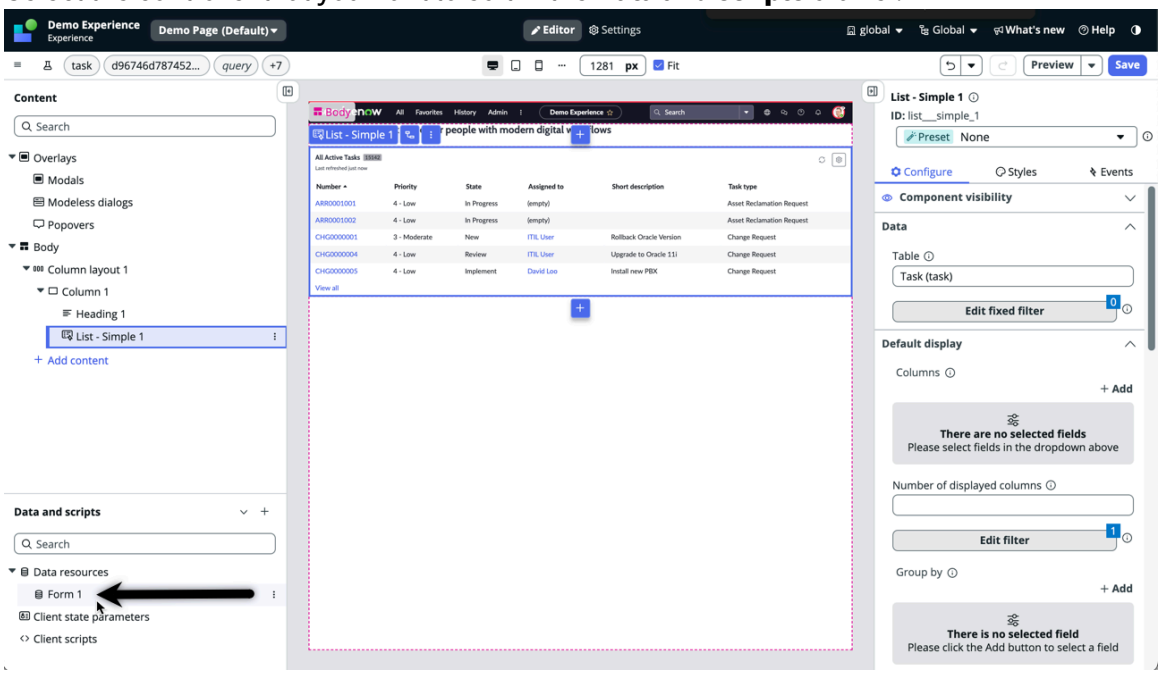
Configure a controller to pull data from a table.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Create or open a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. Select the controller that you want to edit in the **Data and scripts** drawer.



5. On the form, fill in the fields.

Form controller form

Field	Description
Type	Controller is predefined as the type for all controllers.
Table	Add a table that you want the controller to pull data from.
Sys ID	Enter the unique identifier for a record, provide a value, or use -1 to generate a new value.

6. Close the edit controller modal.
7. Select **Save**.

Delete a controller

Delete a controller that you no longer need in UI Builder.

Before you begin

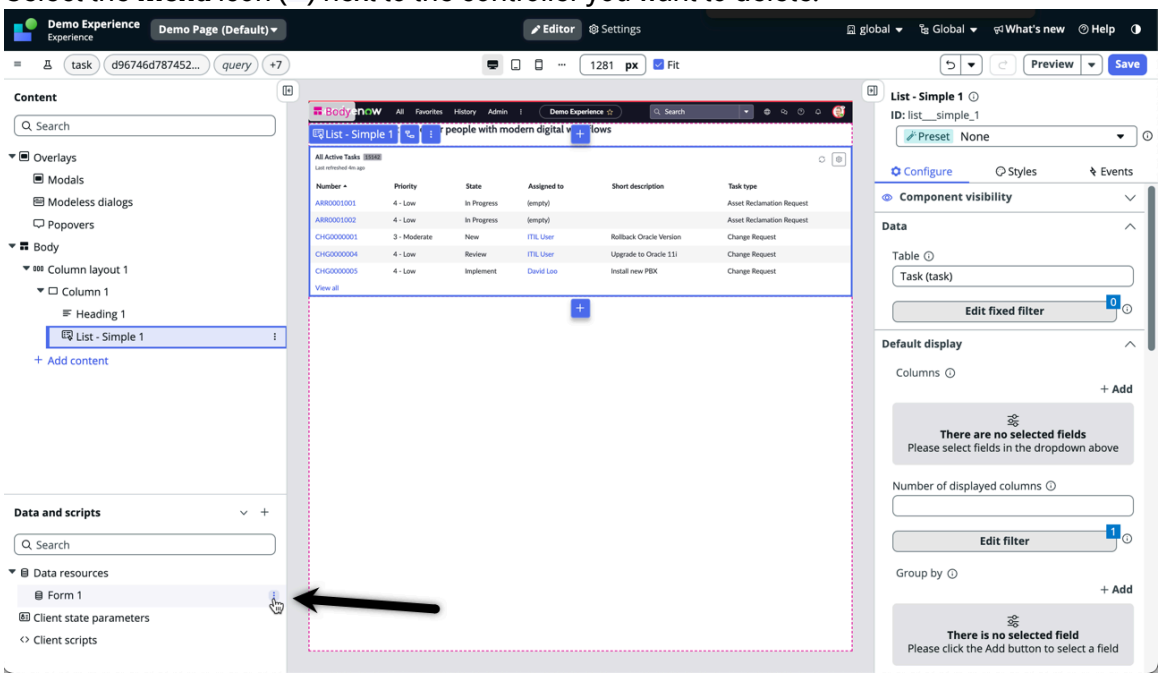
Role required: admin

About this task

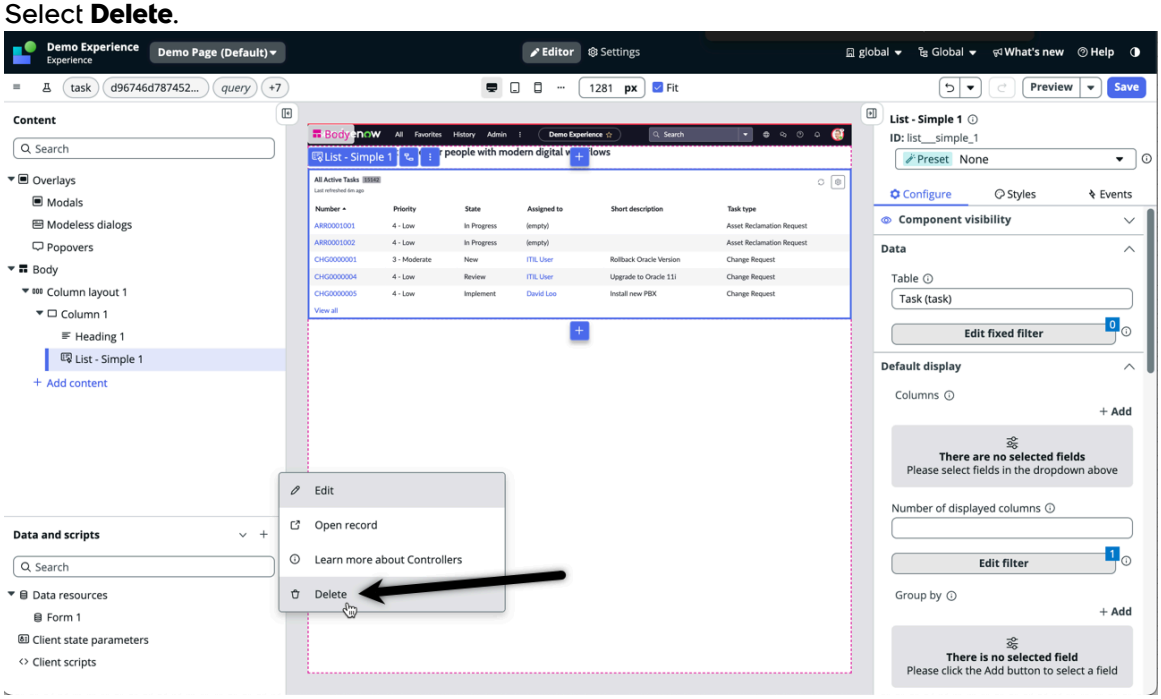
Controllers cannot be deleted from pages created with a page template.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. For more information, see Configure how users interact with your applications in UI Builder.
3. Open the page with the controller you want to delete.
4. Select the **Menu** icon (⋮) next to the controller you want to delete.



5. Select **Delete**.



The controller is removed from the local data resource instances.

View properties and events in the Controller API

The controller public API defines the output data that a controller provides to a preset. This includes the property values and handled events used by a component when a preset is selected. Property and event information is available to view in UI Builder.

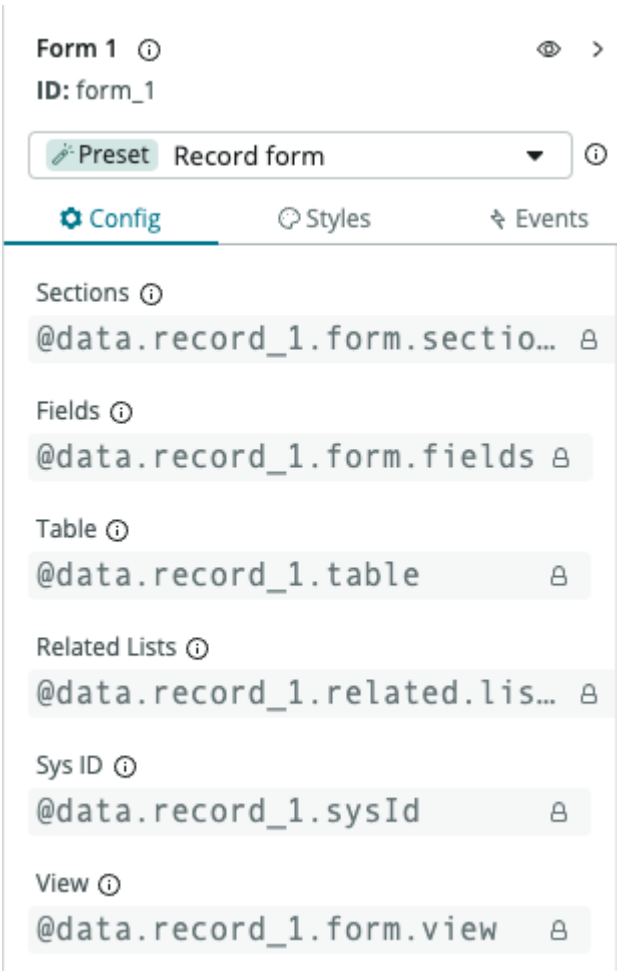
The component property values in a preset can be static values or paths to controller output data. You can use the data resource inspector in UI Builder to display the values from the controller data structure used by a preset. To view the payload carried by an event in the preset, you can look in the event handler picker for that event.

Viewing properties in the Controller API

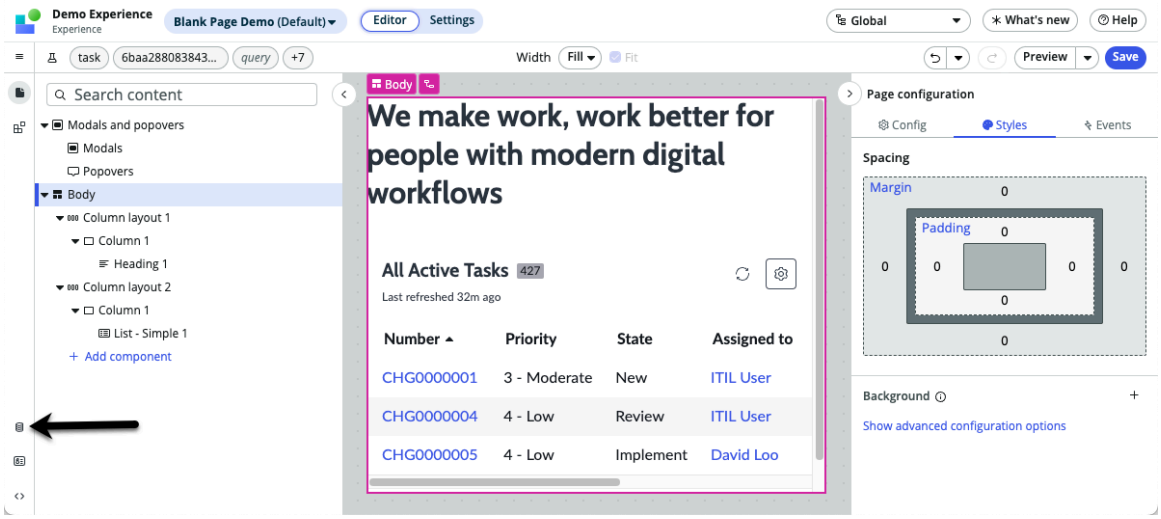
You can view preset properties in the UIB data inspector. Knowing the available values helps you understand how a component will behave in your design and helps you identify any properties in the preset you might want to override.

This procedure assumes that you have placed a component with a preset on the page and have configured a controller. For instructions, see Add a controller.

The configuration tab displays preset property values as a path to the controller output. The base data path is expressed as `@data.<controller_name>`. The remainder of the path is built using the contents of the categories within the controller data hierarchy. You can use this path to view the current values for the record for which the controller has been configured.



1. To view the data resources for a component, select the data icon in the lower left sidebar.

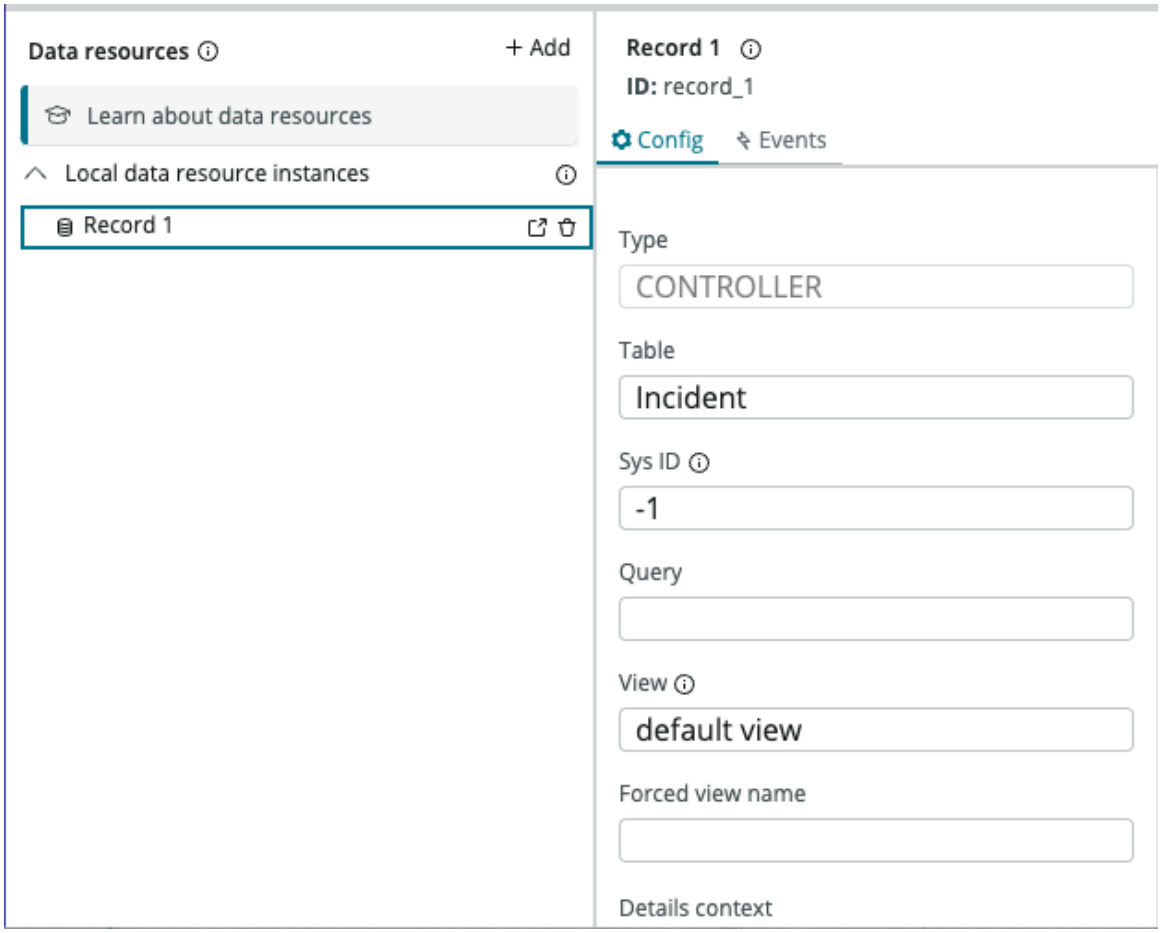


The three column UIB data inspector appears.

2. In the first column where the data resources are listed, select the controller whose output you want to view.

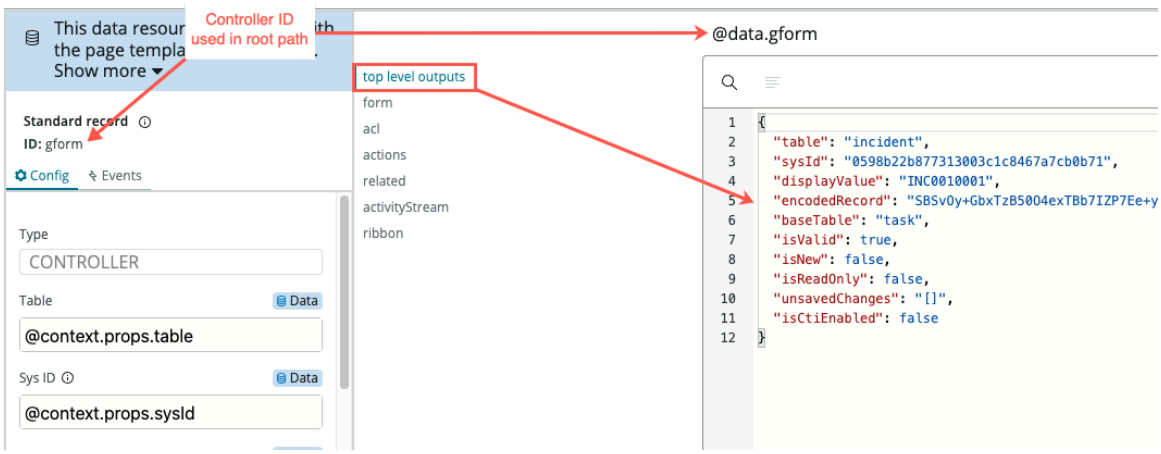
   The configuration data for the selected controller appears in the Config tab of the second column. These are the input properties you entered when you configured the controller for the first component with a preset you placed on the page. You can edit these connection values here if you want.

   Note:
   Configuring **-1** for the value in the **Sys ID** field configures the controller for a new record. The controller then generates a full GUID that you can use to store data against the record before it's saved. This allows you to perform actions in a newly created record such as saving attachments.



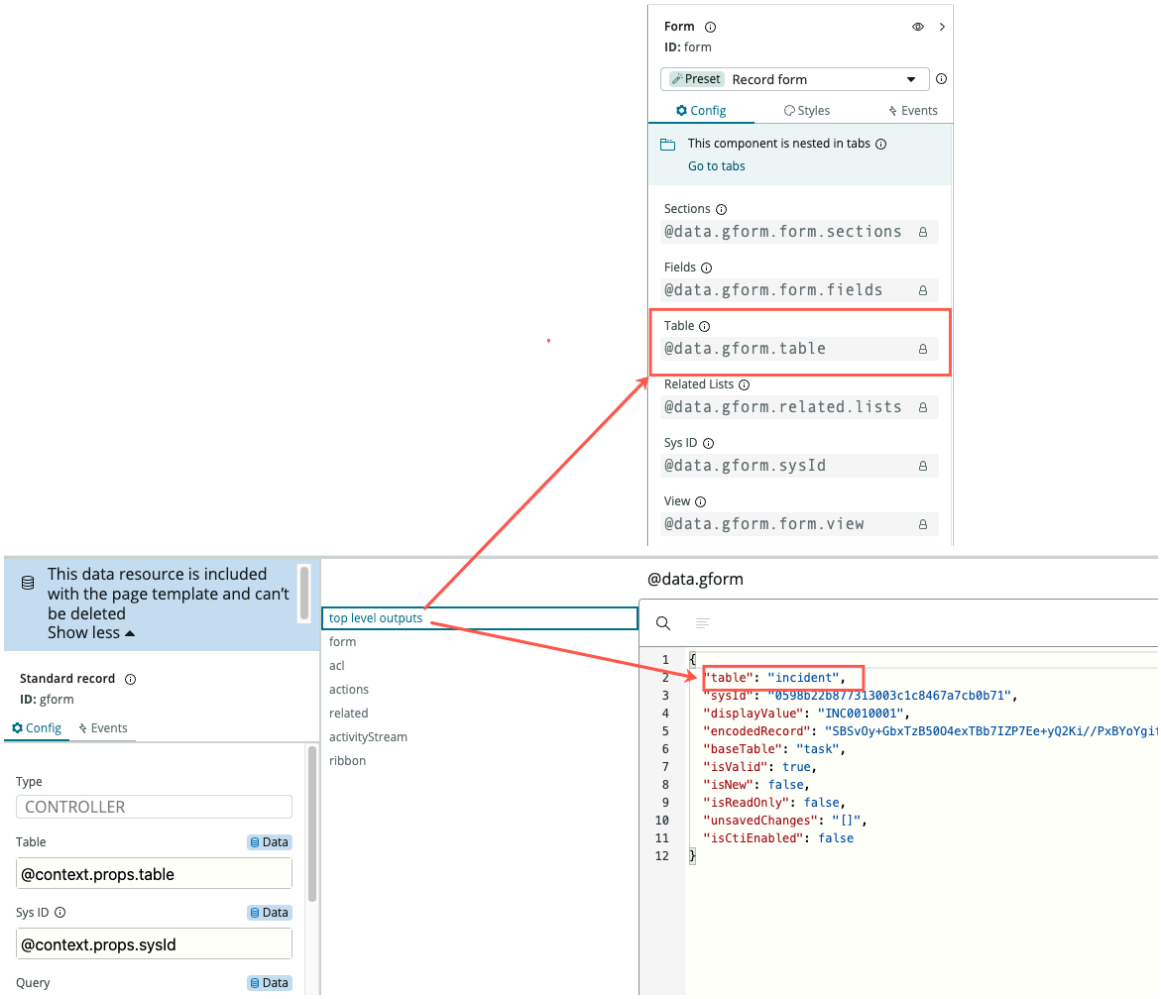
3. Select **top level outputs** in the third column.

   This is the parent level of the controller data hierarchy, indicated with the prefix `@data.`, followed by the controller ID. This combination forms the root path to the data. The data contained in the top level outputs are displayed in the adjacent code field and are available for presets and scripts to use. Other properties at that level are some common properties that apply to all records.



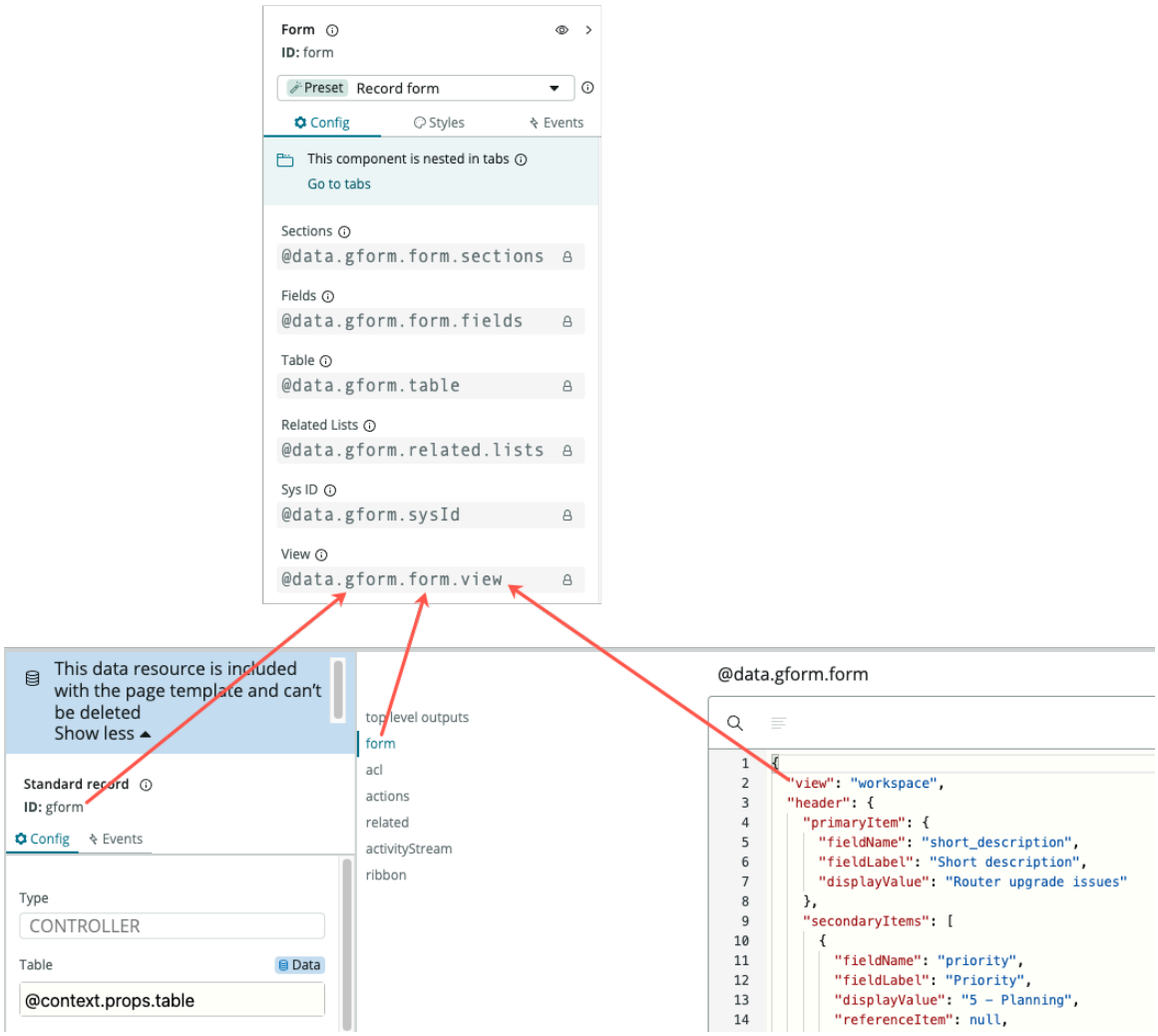
   When the preset maps a component property to controller output, the path to the output is shown instead of the value. Paths are denoted as **@data.<controller id>.<path from top level to output property>** and can be used as the values or as part of formula expressions.

   In this example, the mapping in the preset results in a path to `@data.gform.table`.

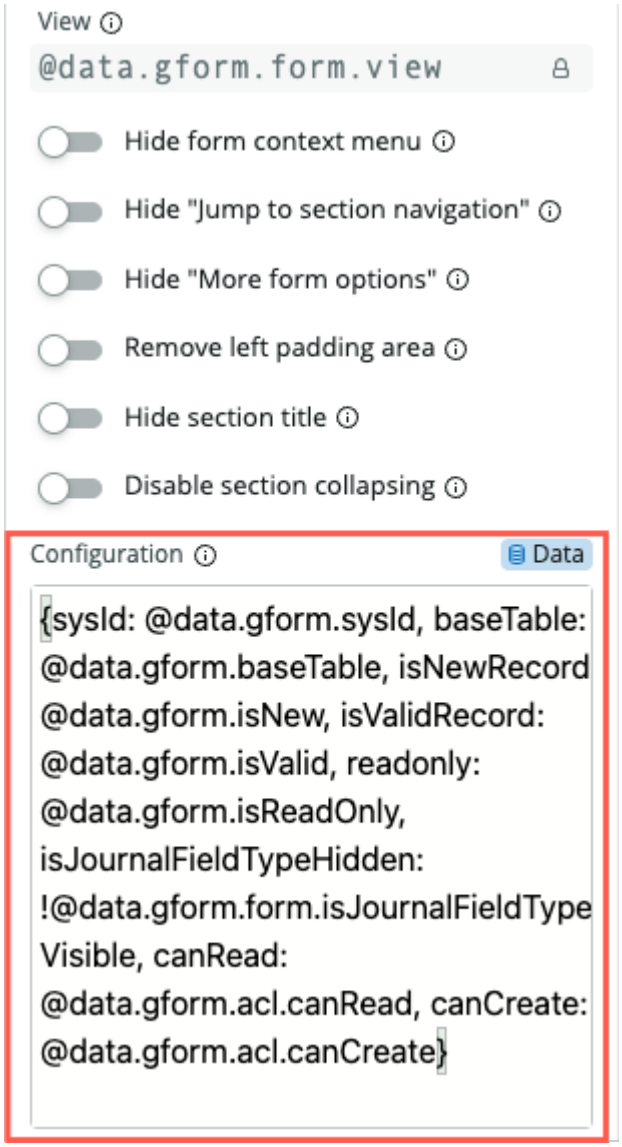


4. To view data below the top level, select the child category indicated by the path in the preset property value.

   The name of the child category appears in the path as `@data.<controller ID>.<category>`. The data from that category is then appended to that path. The root path above the code field indicates the child category being used. In this example, the path to the **View** property in the `form` category is expressed as `@data.gform.form.view`. That preset value displays forms in the **workspace** view.

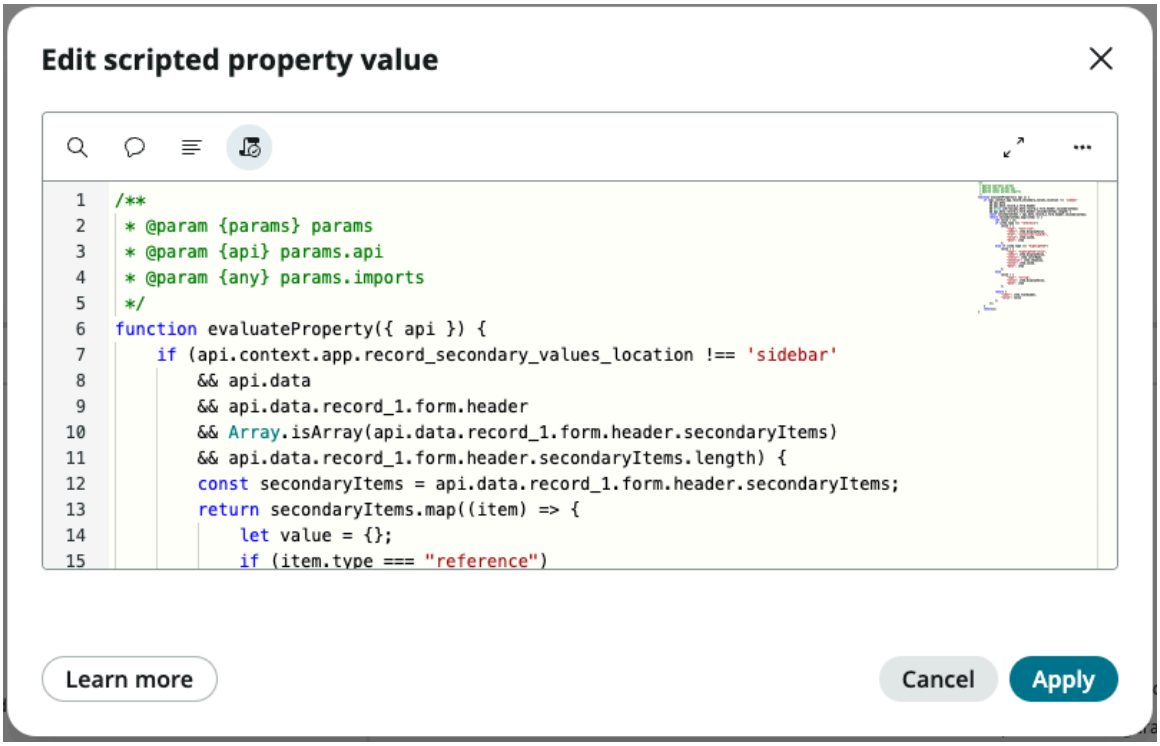


In this example, multiple controller output properties are used in a formula expression to build an object.



5. To view a scripted property value, select the lock icon next to the property.

This puts the property into edit mode and opens a modal for editing the script.



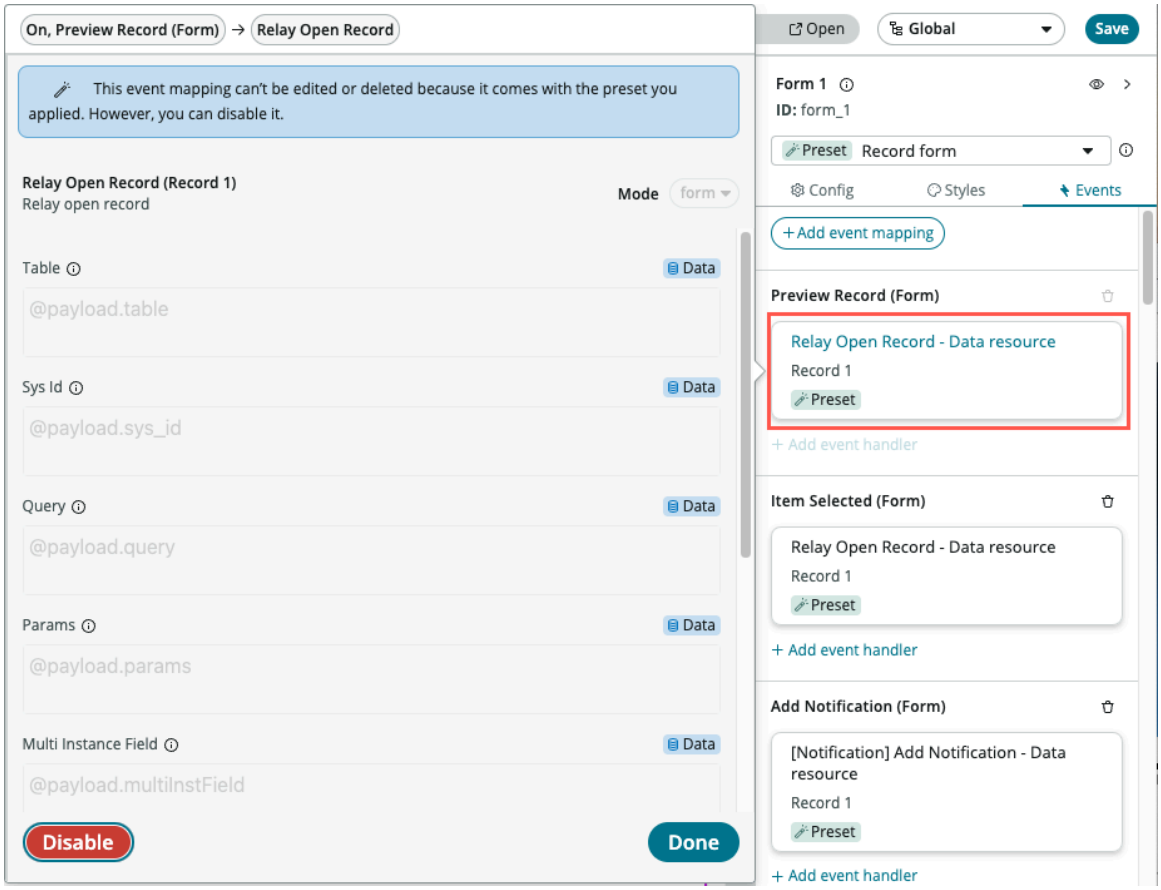
Viewing events in the Controller API

You can view handled events mapped to a component by a preset and their payloads in the Events tab in the UI Builder configuration panel. If the data mapping for an event isn't appropriate for your use case, you can add additional data handlers.

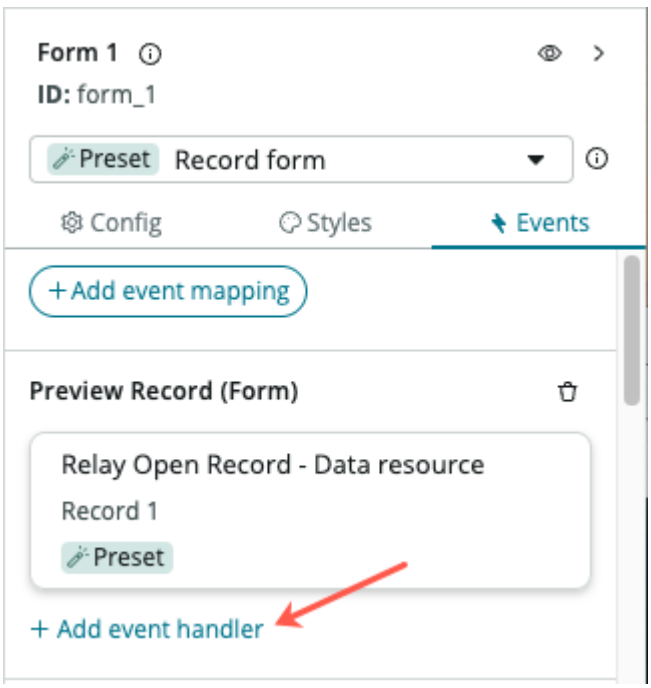
Note:
Data handler mappings provided with the preset are not editable.

1. Select the Events tab in UI Builder.
2. To view the mapping for a preset event, select the event tile.

   A modal appears showing the payload properties for the preset event. You can **Disable** the event in this modal.



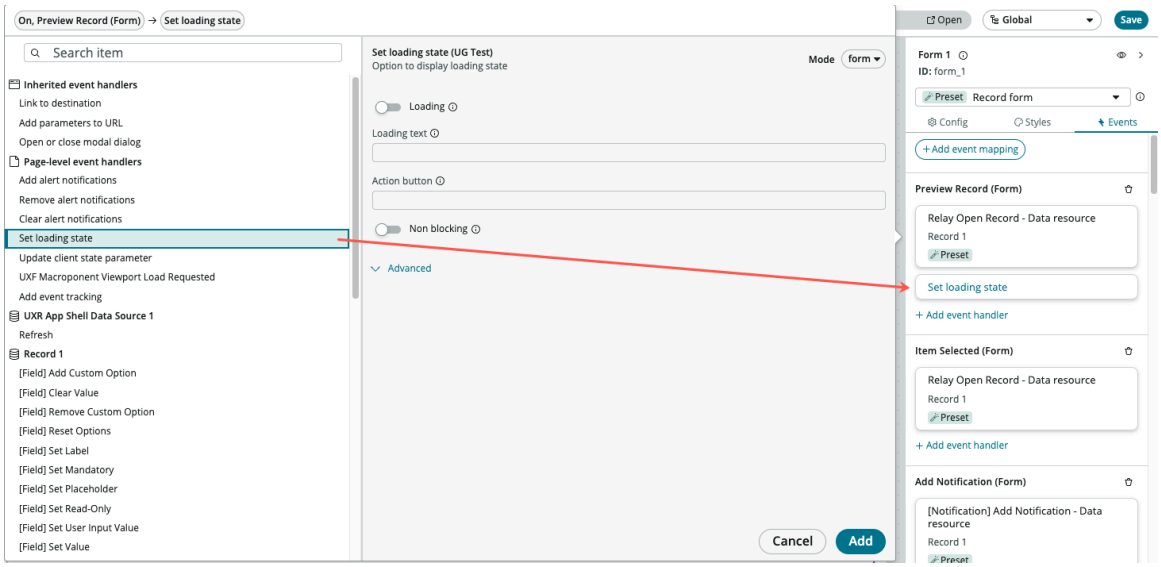
3. To add an event handler to the event, select **+ Add event handler** under the event tile.



A modal appears, showing a list of available, handled events. You can select any action from the list, including an event handler from one of the controllers listed.

4. Select an event handler from the list and select **Add**.

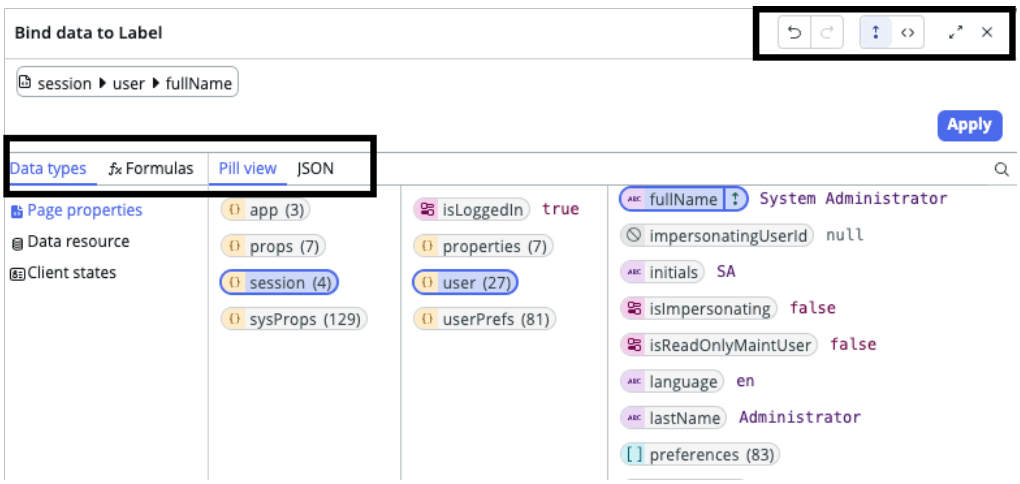
The new handler is listed under the event tile.



Connect data components

Use the data binding modal to associate data exposed by local data resources to components on your UI Builder page.

Data binding modal



Data binding modal options

Option	Description
Undo	Undo the previous change.
Redo	Redo a change that has been undone.
Bind data using drag and drop	Show Pill view.
Use script	Show JSON view.
Toggle full screen	Expand the data binding modal to full screen size. Select again to reduce the data binding modal to smaller size.

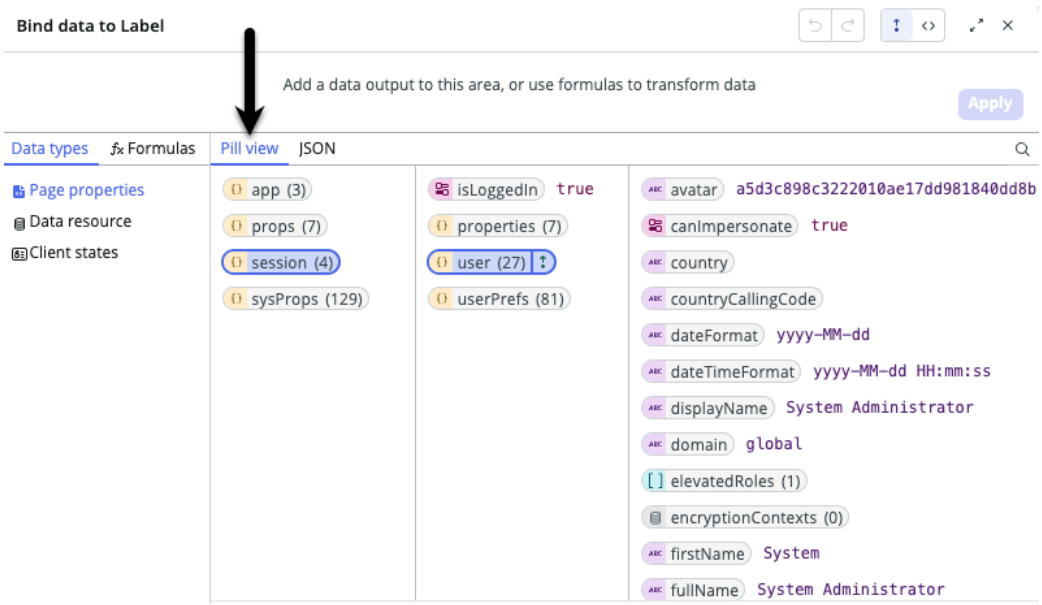
Data binding modal options (continued)

Option	Description
Close	Close the data binding modal. If you have unsaved changes, you are given the option to save.
Data types	Lists page properties, data resources, and client states.
Formulas	Lists formulas that can be used. A search field is available.
Pill view	When **Data types** is selected, pill view enables you to specify a property value or display value. You may have to select several pills to get to a specific value.
JSON	When **Data types** is selected, the JSON option enables you to edit the raw JSON code.

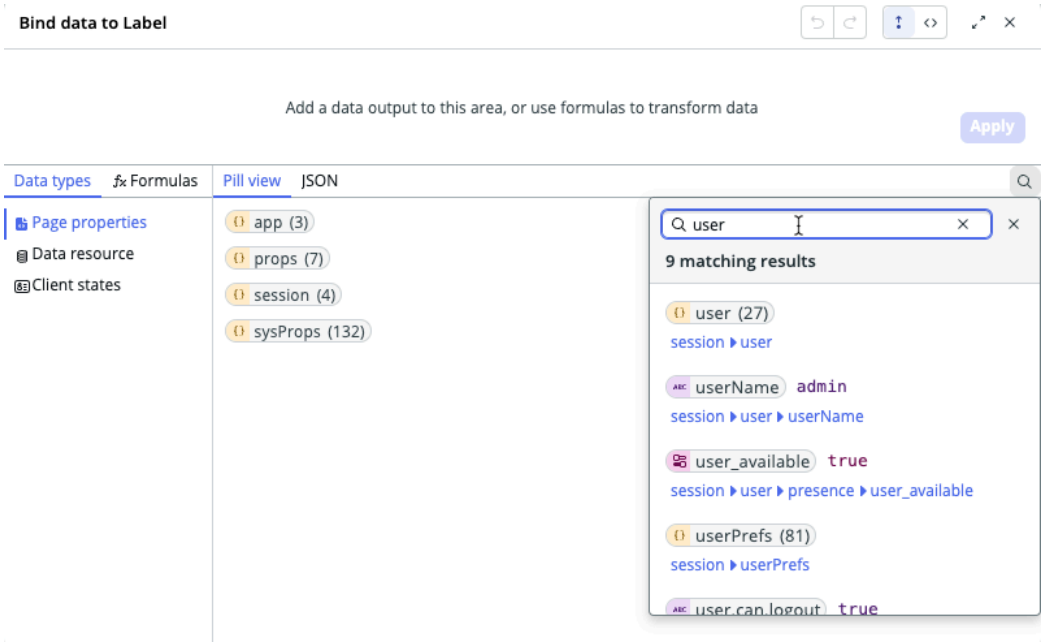
There are two methods you can use for binding data: drag and drop, and editing JSON.

Binding data with drag and drop

Use this method if you want to use a visual process.

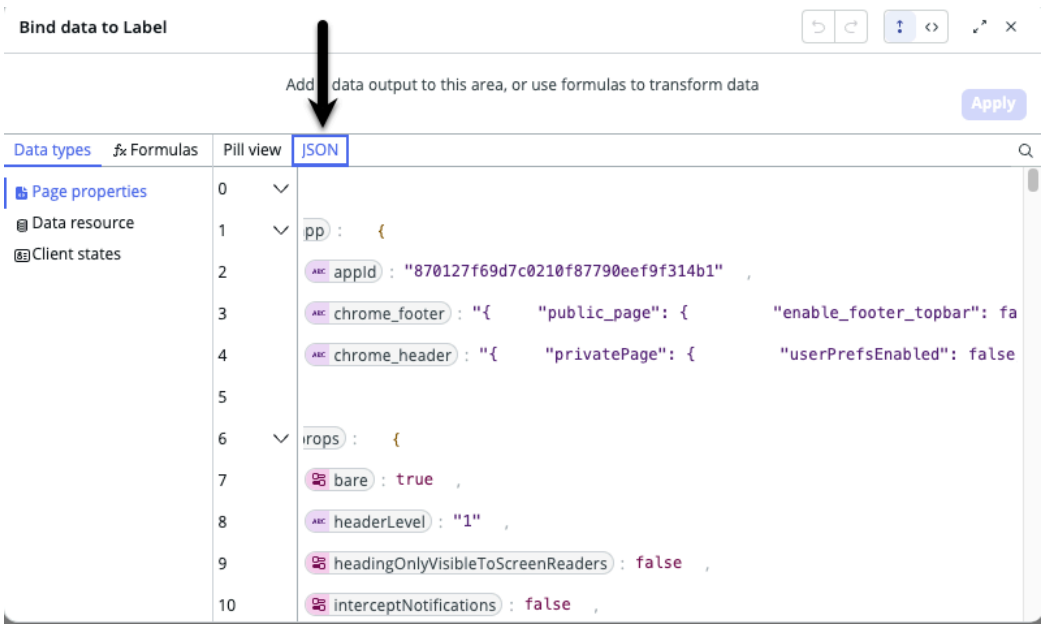


Use the search bar to find the data you would like to bind to your component.



Binding data by editing JSON

Use this method if you are comfortable working in and editing JSON code.



Connect data to your components

Bind data exposed by local data resources to components on your UI Builder page.

Before you begin

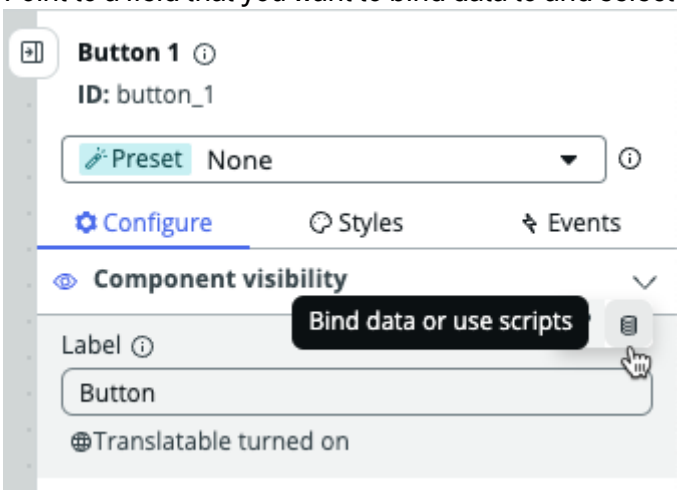
Role required: ui_builder_admin

About this task

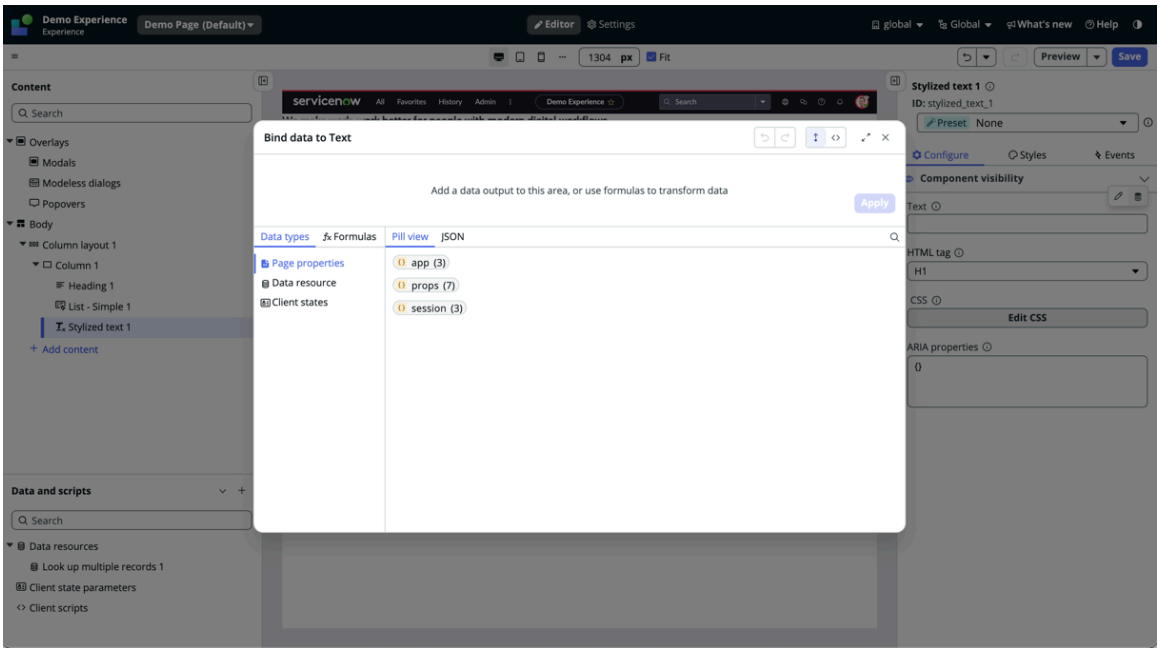
Data binding is the process of associating data with a UI element that displays the information. Before binding data to components you must add a data resource to your UI Builder page, see Add and configure data resources to a page for more information.

Procedure

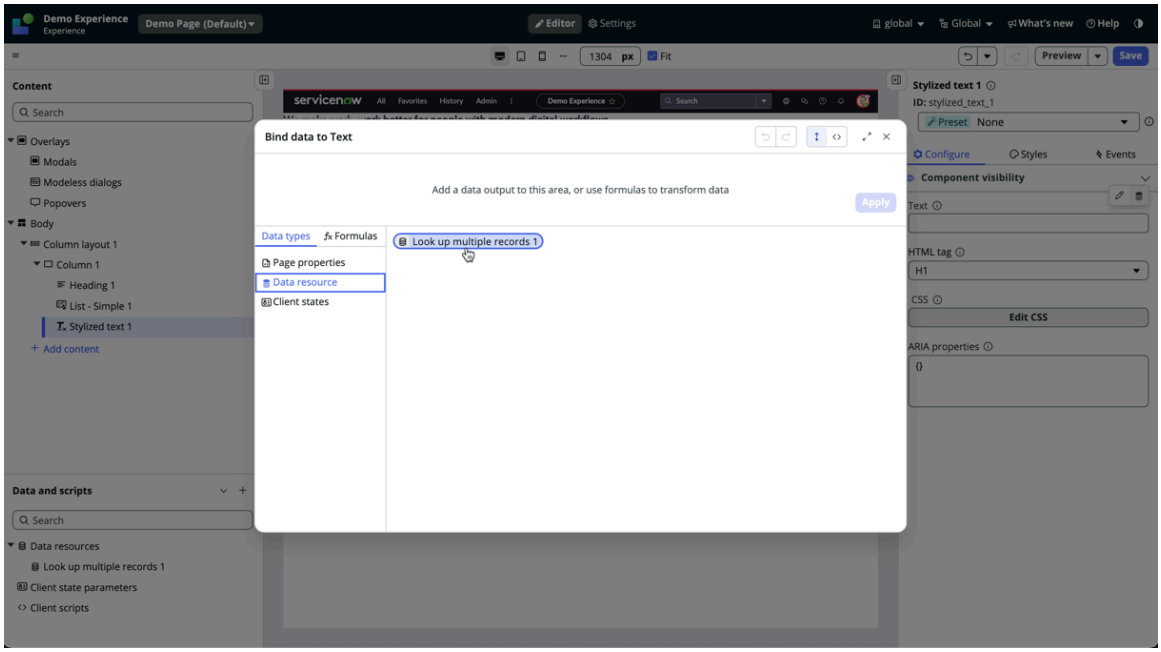
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page.
4. Add a data resource to your page.
   For more information, see Add and configure data resources to a page.
5. Add a component to your page.
   You need a component on your page before you can bind a data resource to it. For more information, see Customize UI Builder pages using components.
6. Select the **Configure** tab from the configuration panel in UI Builder.
7. Point to a field that you want to bind data to and select the dynamic data binding icon ( ).



The data binding modal appears.

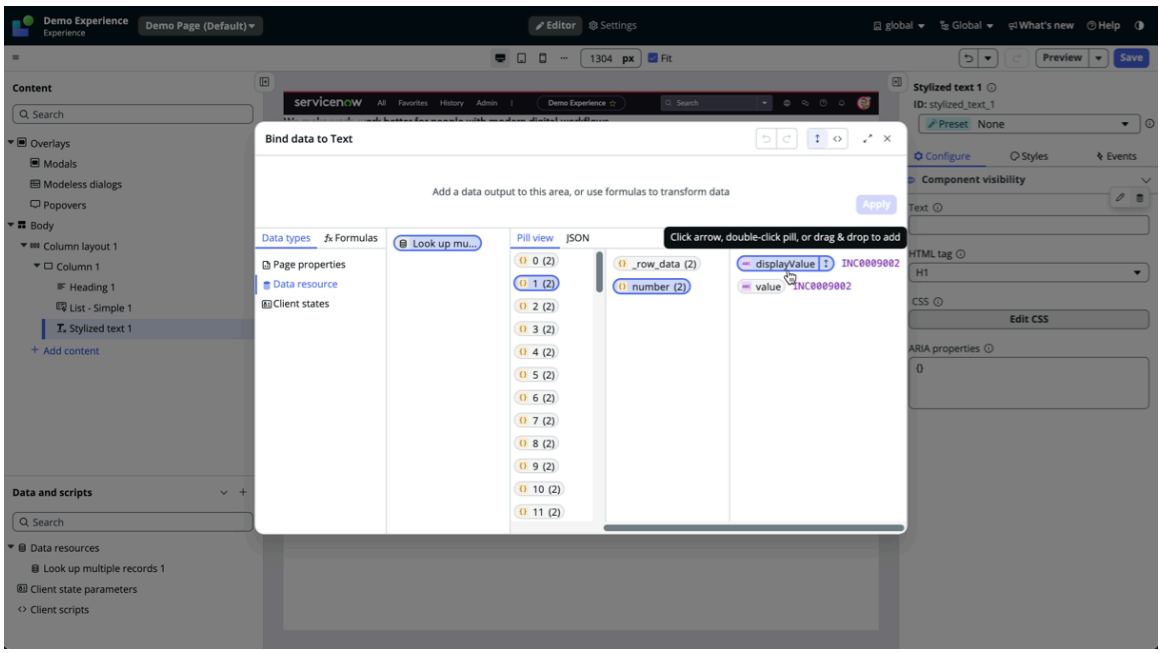


8. Select **Data resource** in the **Data types** tab.
9. Select the data resource that you want to bind to the component.



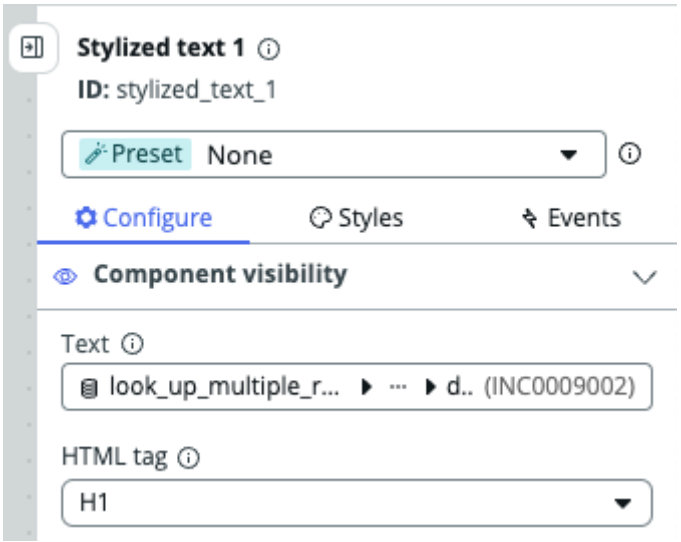
10. Select and drag the data you want to bind to the component into the field above.

You may have to select several pills to get to a specific property value or display value.



11. Select **Apply**.

The component displays the property value or display value that you've selected.



12. Select **Save** in the UI Builder header.

Connect data to your components with formulas

Bind data exposed by local data resources to components with formulas on your UI Builder page.

Before you begin

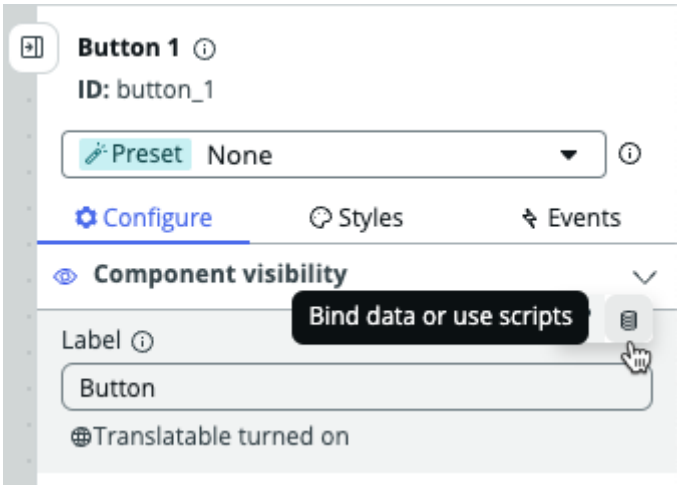
Role required: ui_builder_admin

About this task

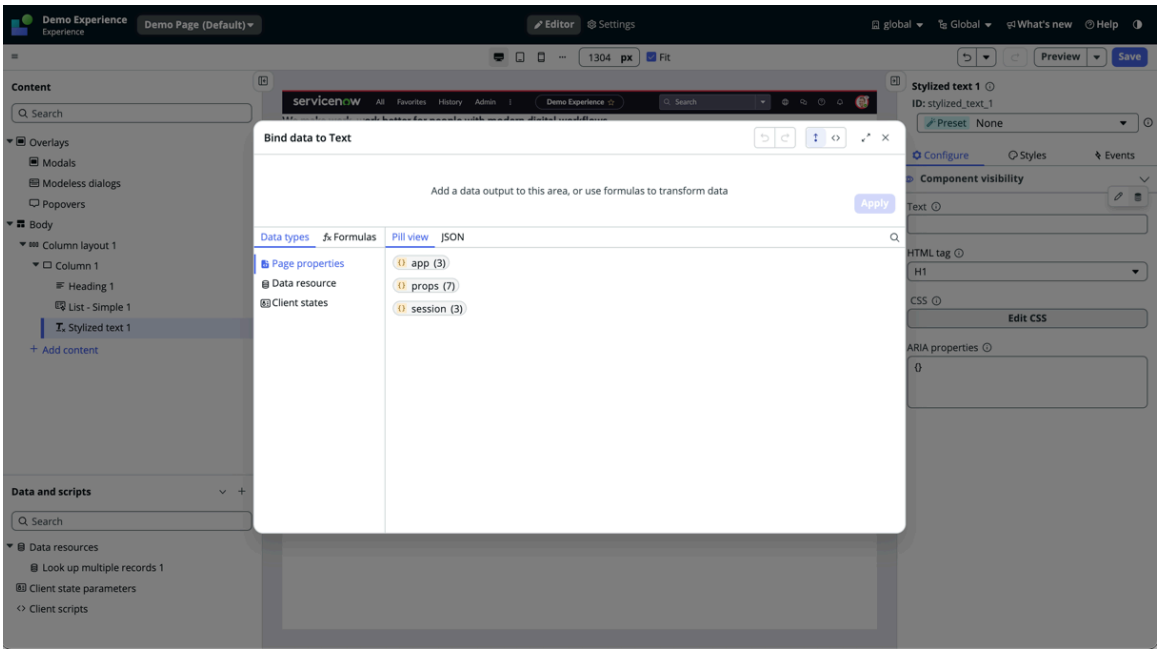
Data binding is the process of associating data with a UI element that displays the information. Before binding data to components you must add a data resource to your UI Builder page, see Add and configure data resources to a page for more information.

Procedure

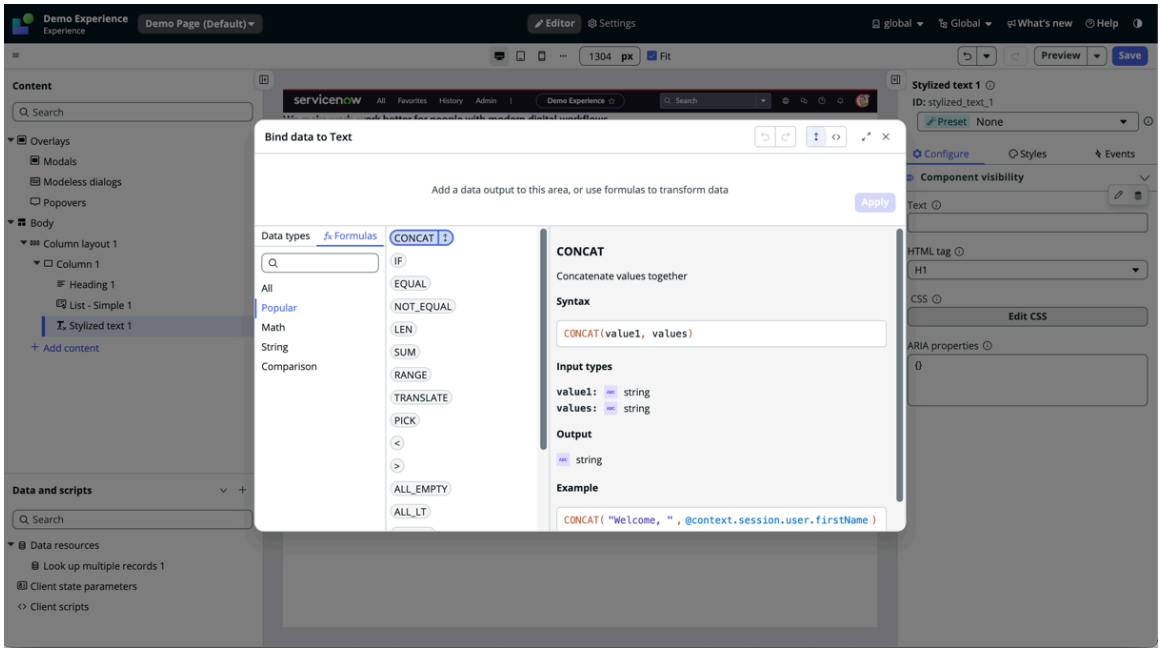
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page.
4. Add a data resource to your page.
   For more information, see Add and configure data resources to a page.
5. Add a component to your page.
   You need a component on your page before you can bind a data resource to it. For more information, see Customize UI Builder pages using components.
6. Select the **Configure** tab from the configuration panel in UI Builder.
7. Point to a field that you want to bind data to and select the dynamic data binding icon ( ).



The data binding modal appears.

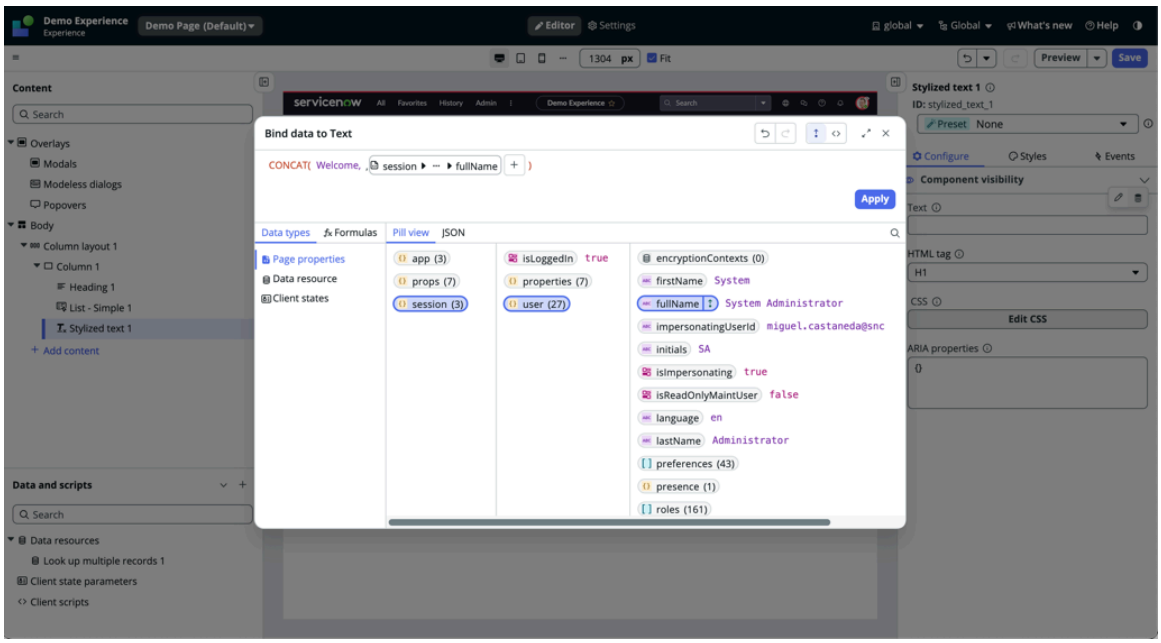


8. Select the **Formulas** tab.
9. Select and drag the formula you want to bind to the component into the field above.



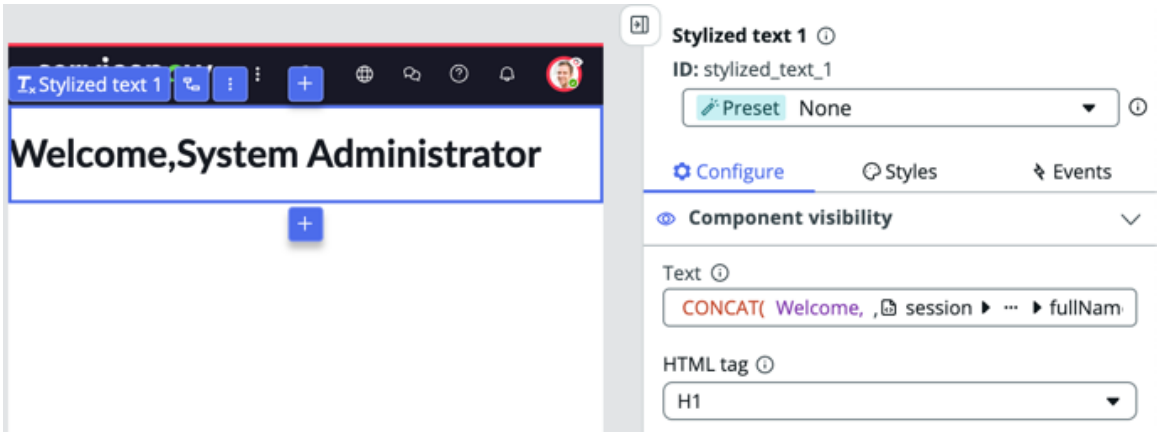
For more information about formulas, see Supported functions in the UI Builder component formula editor.

10. Fill in the fields of the formula.



11. Select **Apply**.

The component displays the property value or display value that you've selected.



12. Select **Save** in the UI Builder header.

Client state parameters

Use client state parameters to bind values to component configurations. When the client state parameter's value changes, the component updates to use the new value.

What is a client-side interaction

In web-based applications, anything that happens in the browser is client-side. Client-side interactions occur when users interact with experience components by clicking. Examples of client-side interactions include:

- Clicking a button
- Applying a filter
- Sorting a list
- Refreshing a list
- Providing inline search results
- Alerting a user to incomplete or incorrect data

Users sometimes interact with pages in order to navigate to other pages. In other cases, users want to make updates to a page's content, appearance, or data. Rather than refreshing the entire page, client-side interactions update only the affected parts of a page.

For example, a user can sort a list by clicking a column header. Sorting redraws only the list and not the entire page.

Creating client-side interactions in UI Builder involves:

- Client state parameters
- Events
- Client scripts

What are client state parameters

Client state parameters are page variables. Define and configure a client state parameter and use the value to configure components. Client state parameters centralize management of configuration values for page components. UI Builder uses client state parameters to enhance user experience and provide personalized content and services.

Example of client state parameters

In the example, a page has a client state parameter called color. Two of the three components are configured to use the color client state parameter. If color is set to orange, the components are orange. If color is set to purple, the components are purple. If color is set to green, the components are green. The client state parameter is a single place to manage a common value for the page. Without the client state parameter, each component that uses a value would need to be updated individually if that value changed.

For example, a web experience stores the primary color used by components in the color client state parameter. When the components are configured to use the client state parameter, changing the value of the client state parameter updates the components to the new value.

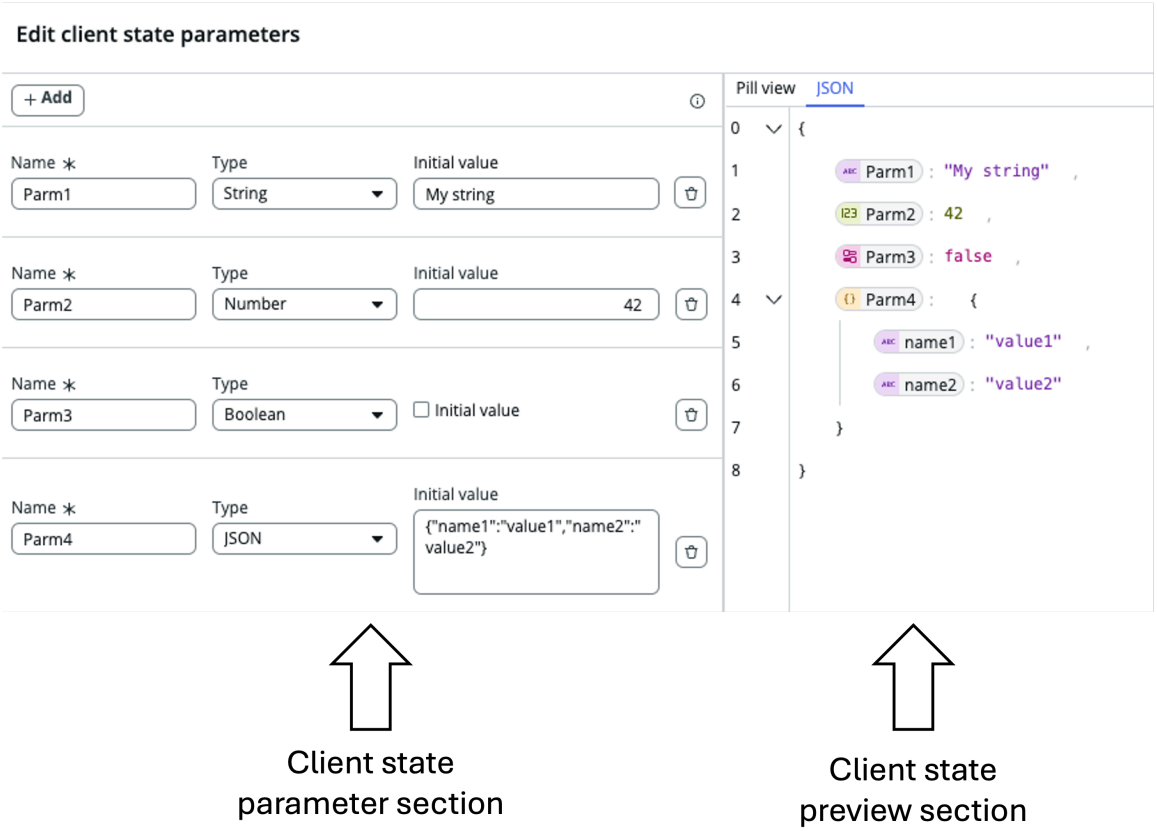
Buttons can be added to the experience to allow users to select a color for the page components. With a single click, a user can simultaneously change the color of all components on a page. Store the user's color choice in a client state parameter, then use the client state parameter to configure the page's components. When a user interaction causes the client state parameter value to change, the page's components are updated in real time.

The client state panel

The client state panel is collapsed by default. Click the client state icon in the left navigation bar to open the client state panel.

The client state panel contains two sections:

- Client state parameters: The client state parameters for the page
- Client state preview: The JSON for the page's client state parameters

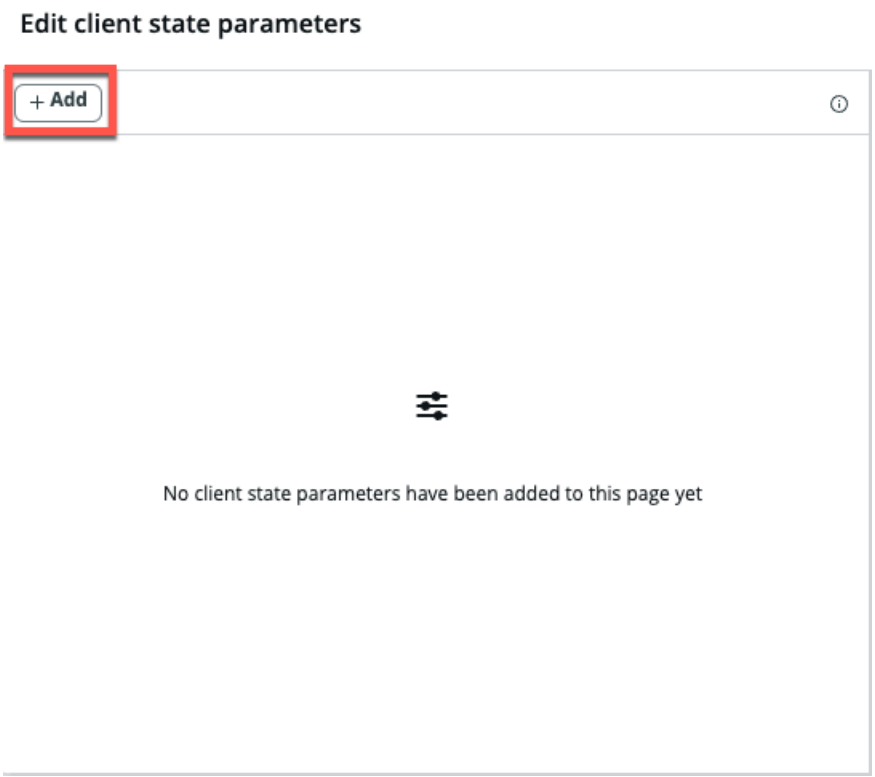


Client state parameter section

Client state preview section

Creating client state parameters

To add a client state parameter to a page, click the **+ Add** button in the client state parameters section.

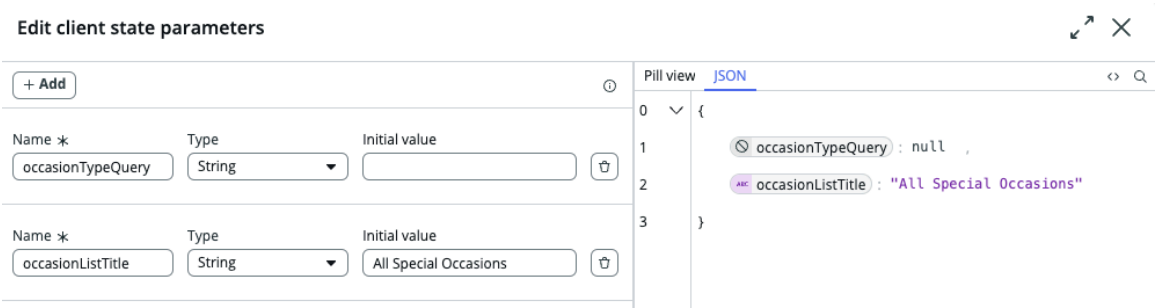


Client state parameters have three configuration fields.

- Name: The name of the parameter. Names should not include spaces
- Type: The parameter's type
  - String
  - Number
  - Boolean
  - JSON
- Initial value: The default value for the parameter

In the example, the client state panel has two client state parameters.

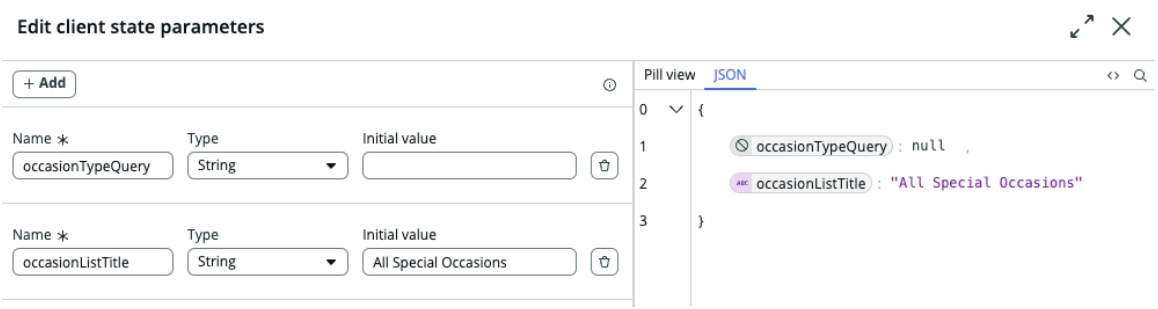
- `occasionTypeQuery` is a string with no default value
- `occasionListTitle` is a string with a default value of `All Special Occasions`



Working with client state parameters

Once you have a client state parameter, what can you do with it? To work with client state parameters, first bind the value of the client state parameter to component configurations. When the client state parameter's value changes, the component updates to use the new value. One way to change a client state parameter's value is with event handlers.

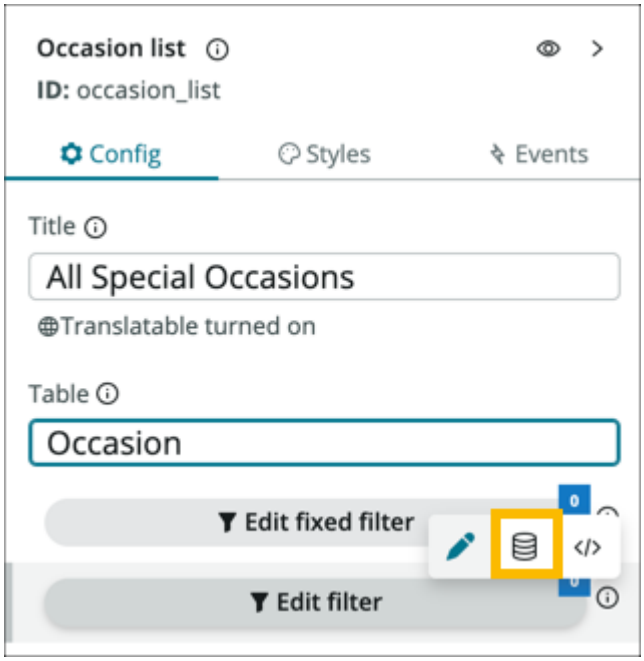
In the creating client state parameters page example, two client state parameters were added to the special occasions application: `occasionTypeQuery` and `occasionListTitle`. The `occasionTypeQuery` parameter has no value by default and `occasionListTitle` has the default value `All Special Occasions`. These client state parameters will be applied to the Occasion List component to set the list's title and filter. A button will be used to update the values of these client state parameters to dynamically update the list.



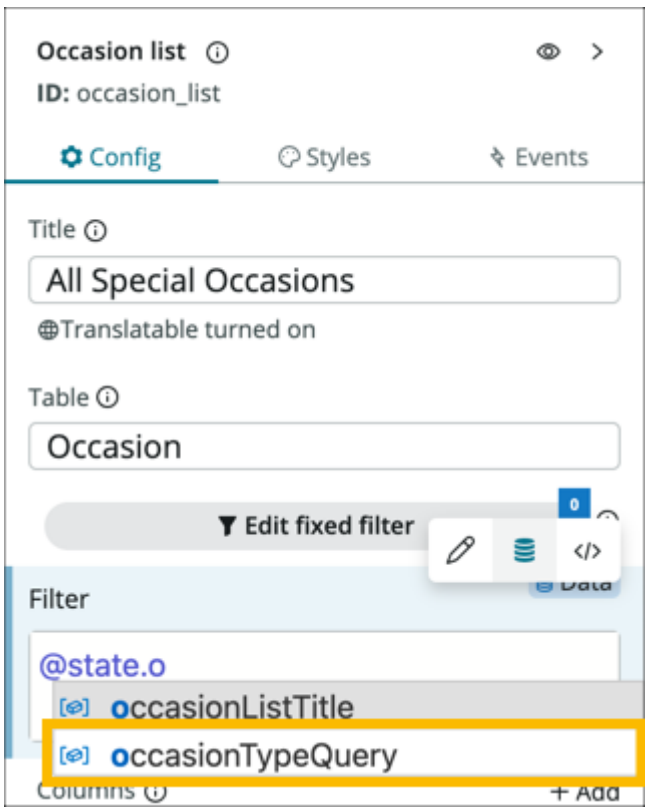
Binding client state parameters

Client state parameter values can be bound to component configuration fields the same way page context, payload, and data resources are. When configuring components, hover over a field, then select the **Dynamic data binding** button ( ) to bind a client state parameter to the

field value. In the example, the **Dynamic data binding** button is highlighted for the Occasion List component's filter field.

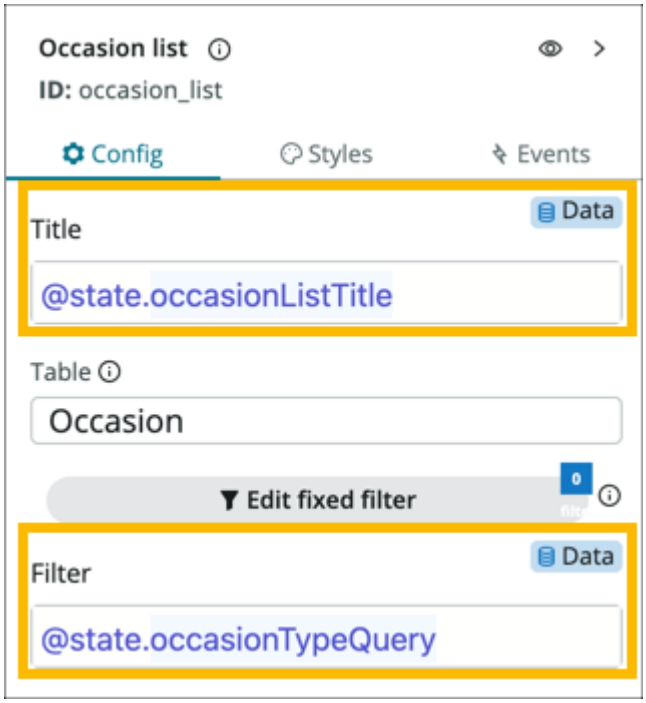


The character @ indicates data binding and the state object contains the client state parameters. Select a client state parameter from the choice list. The example shows selecting the `occasionTypeQuery` client state parameter. The default value for the `occasionTypeQuery` client state parameter is empty, so no filter will be applied by default.

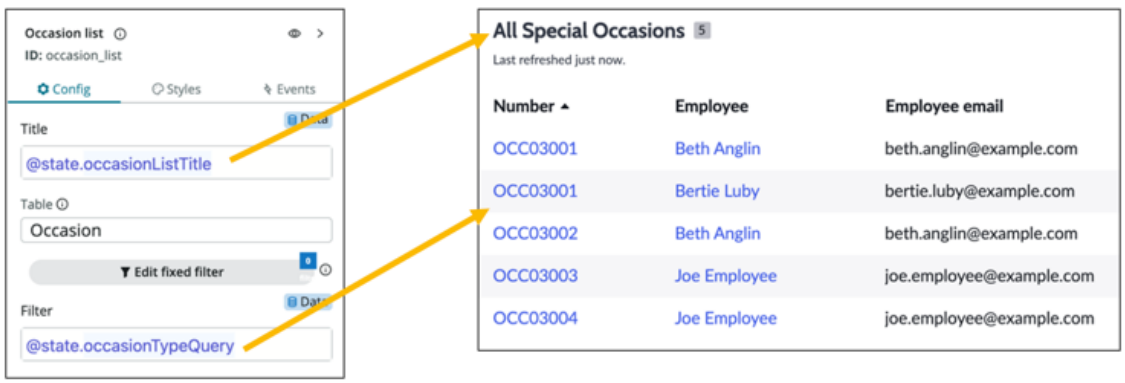


The Title for the component is set in a similar manner: select the **Dynamic data binding** button for the Title field or type @, then select the **state.occasionListTitle** client state parameter from the choice list. The default value for the `occasionListTitle` client state parameter is `All`

`Special Occansions`. The example shows both Title and Filter configured to use client state parameters.



With both client state parameters applied, no filter is applied to the list and the title is `All Special Occasions`.



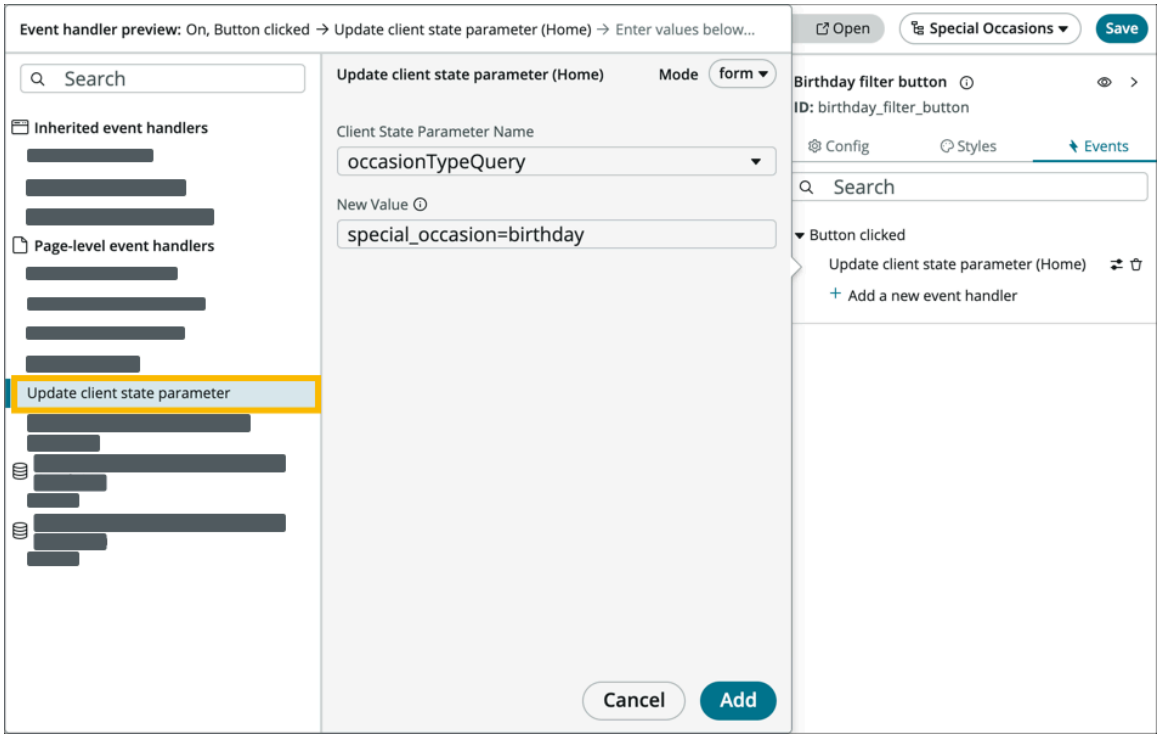
The `occasionTypeQuery` and `occasionListTitle` client state parameter values remain unchanged until a user interacts with a component that changes the value of the client state parameter.

Change client state parameter values

User the **Update client state parameter** event handler to change the value of a client state parameter in runtime. The **Update client state parameter** event handler has two properties: client state parameter to update and the new value to use for the client state parameter.

As an example, a `Birthdays` button is added above the `Occasion List` component in the `Special Occasions` application. The objective of this button is to change the `Title` of the list to `All Birthdays` and to adjust the filter of the list to only show birthdays. To achieve this objective, two event handlers are mapped to the **Button clicked** event for the button, one to change the `occasionTypeQuery` client state parameter and another to change the `occasionListTitle` client state parameter. The image shows the **Update client state**

parameter event handler configured to set the `occasionTypeQuery` client state parameter to `special_occasion=birthday`, which will filter the list to only show birthdays.



When the **Button clicked** event is mapped to event handlers to update both the `occasionTypeQuery` and `occasionListTitle` client state parameters, clicking the button updates the client state parameters, which automatically updates the `Occasion List` component with the new values.

Using client state parameters in UI Builder

Create a simple counter by adding the stylized text component and two buttons to an experience page. Use a client state parameter to implement the functionality so that when the buttons are selected the count increases or decreases.

Before you begin

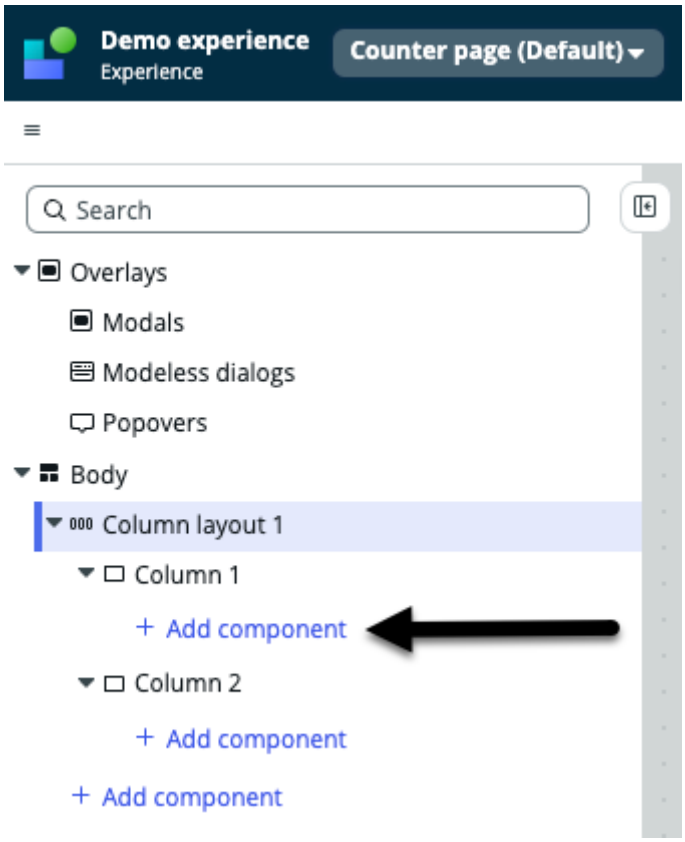
Role required: ui_builder_admin

Three minute video showing how to add a stylized text component and two button iconic components to create a simple counter application.

Procedure

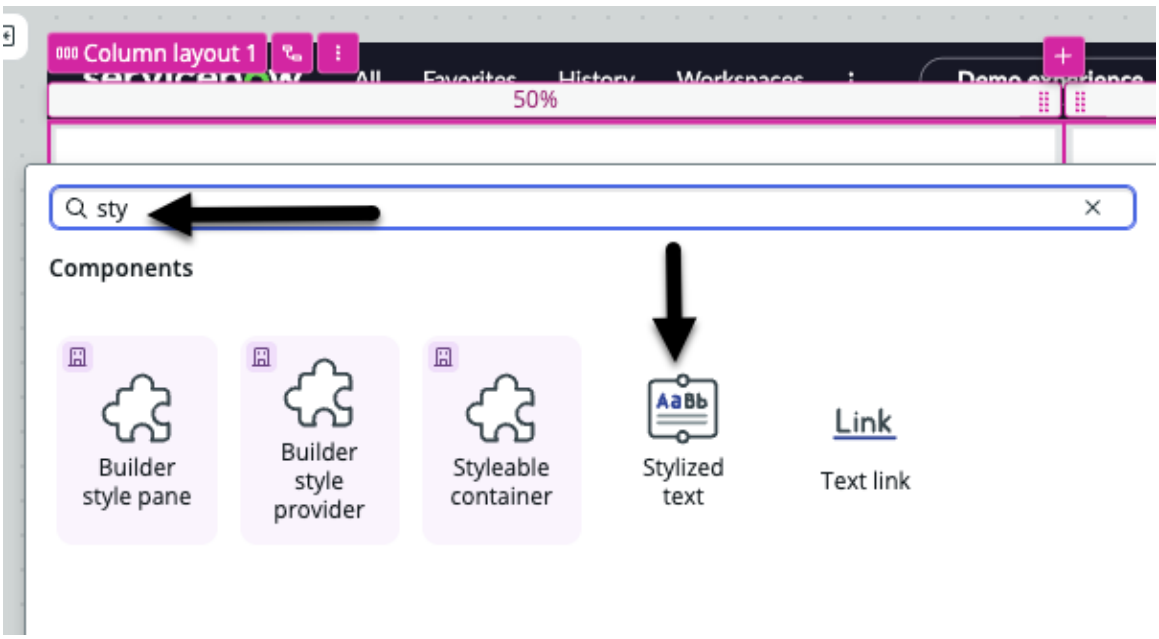
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create > Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.

3. Create a page in UI Builder or open a page.
4. Add a column layout by selecting **+ Add content** in the content tree under **Body**.
5. On the **Layouts** tab, select **Two columns**.
6. Add the first component by selecting **+ Add content** in the content tree under **Column 1**.

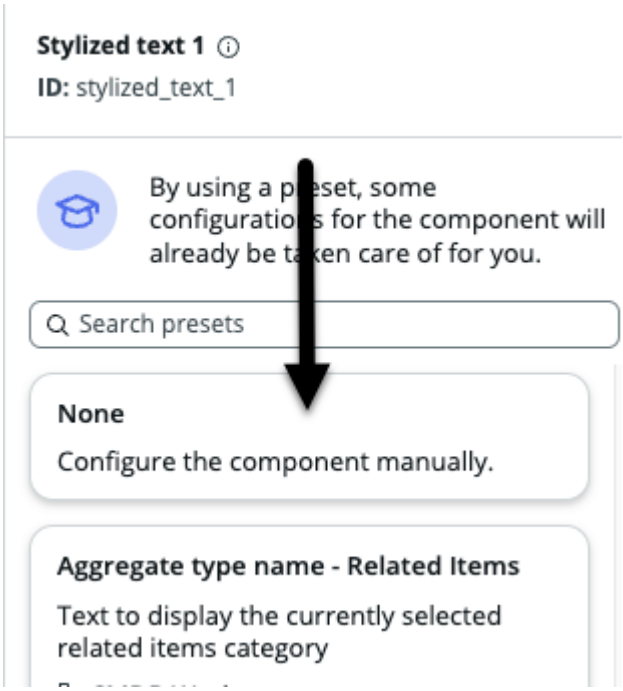


a. In **Search**, type `sty`.

b. Select the **Stylized text** component.



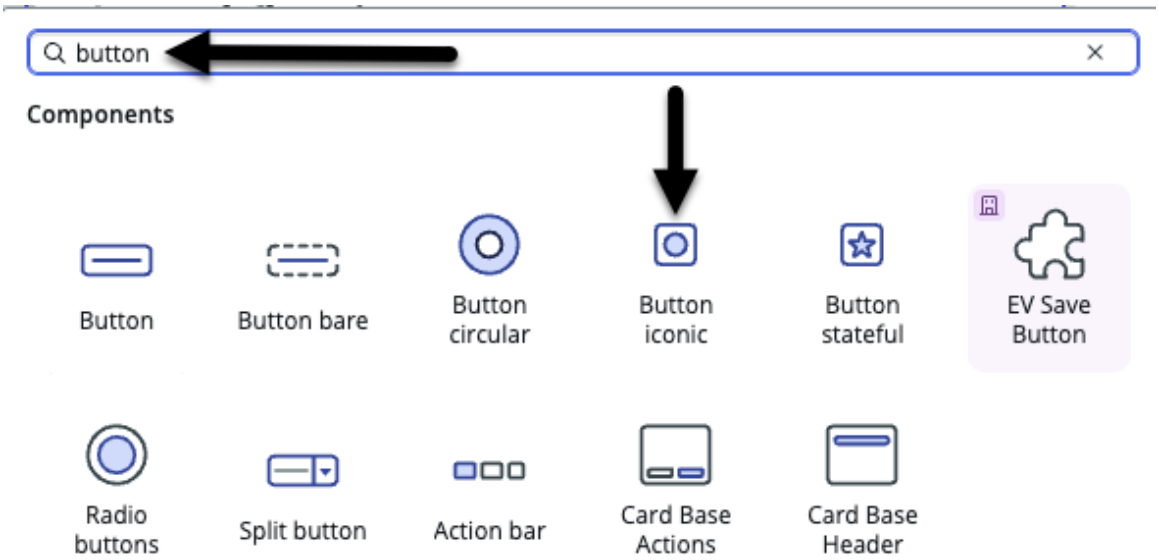
c. In the configuration panel, select **None - Configure the component manually**.



7. Add the second component by selecting **+ Add content** in the content tree under **Column 2**.

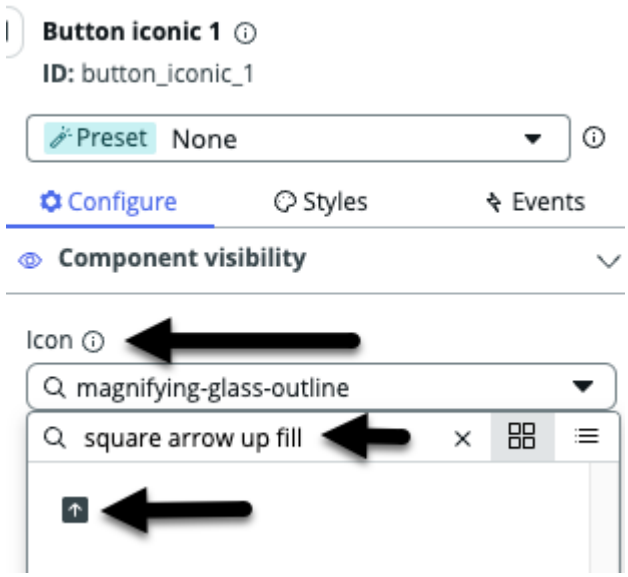
a. In **Search**, type `button`.

b. Select **Button Iconic**.

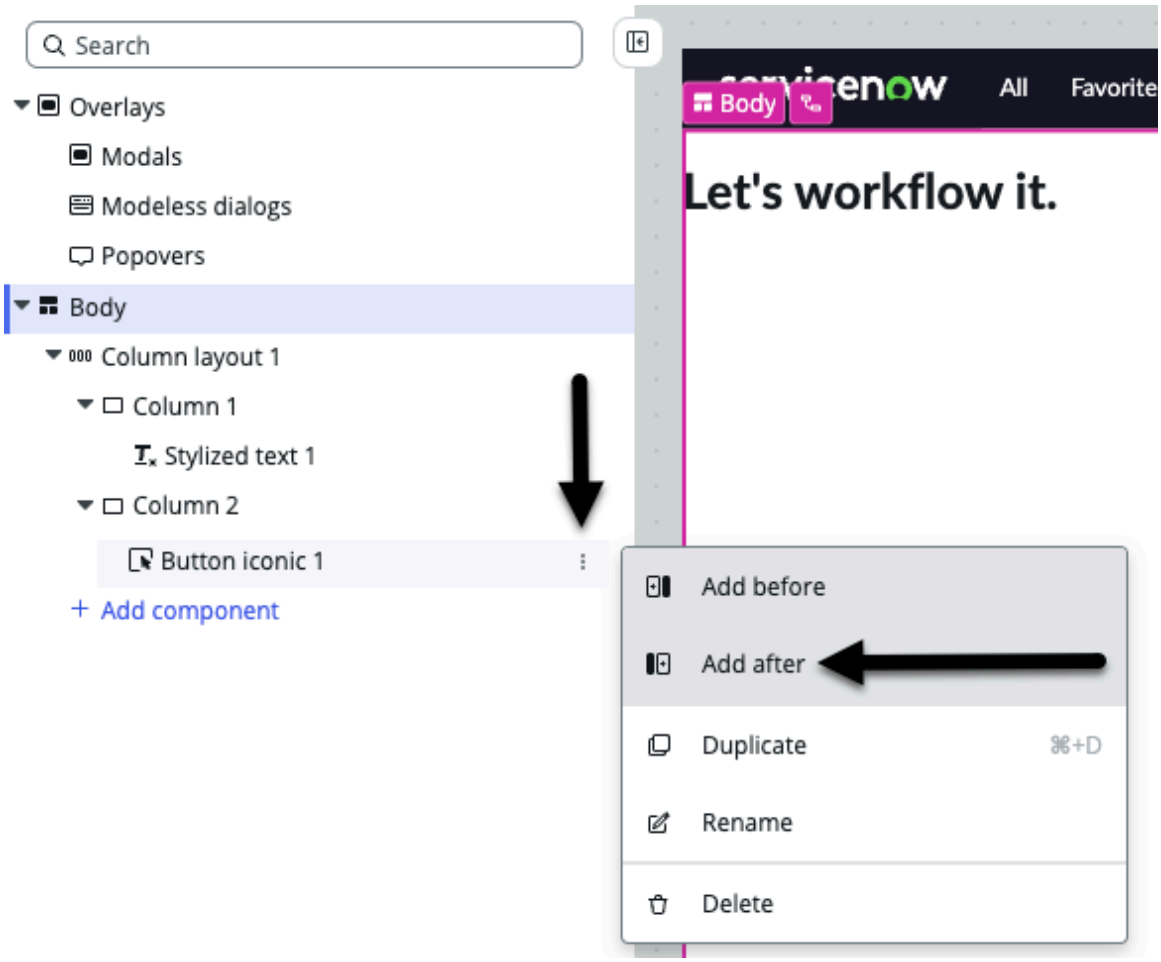


c. In the configuration panel, select **None - Configure the component manually**.

d. In **Icon**, select **Square Arrow Up Fill**.



8. Add the third component by pointing to **Button iconic 1** in the content tree, selecting the **Menu** icon, and selecting **Add after**.

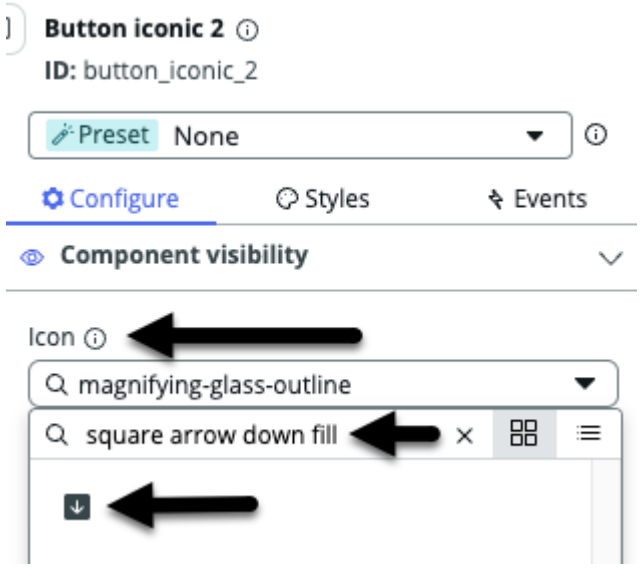


a. In **Search**, type `button` as you did in the previous step.

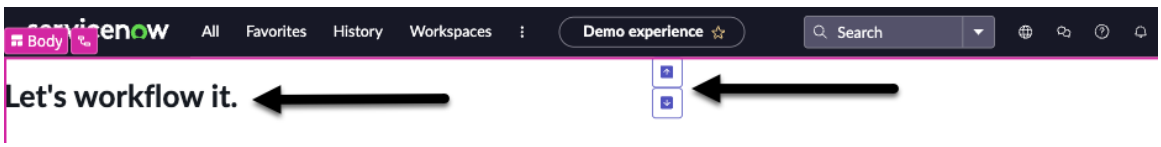
b. Select **Button Iconic** as you did in the previous step.

c. In the configuration panel, select **None - Configure the component manually** as you did in the previous step.

d. In **Icon**, select a different icon this time, the one named **Square Arrow Down Fill**.



9. Select **Save**.
The stylized text component and the two button iconic components are saved and appear on the stage.



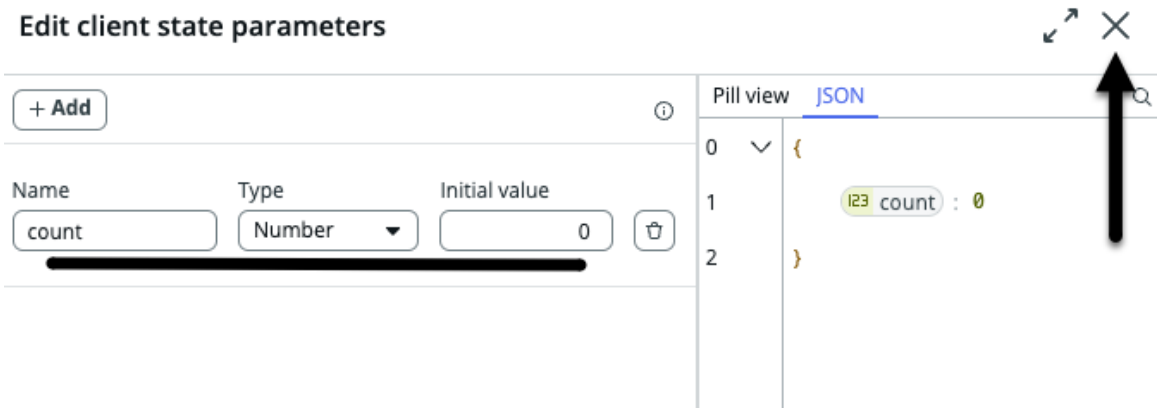
10. Add a client state parameter by going to **Data and scripts**, pointing to **Client state parameters**, and selecting the **Add new (plus)** icon.

a. Change the **Name** by entering `count`.

b. Change the **Type** to **Number**.

c. Set the **Initial value** by entering the number `0`.

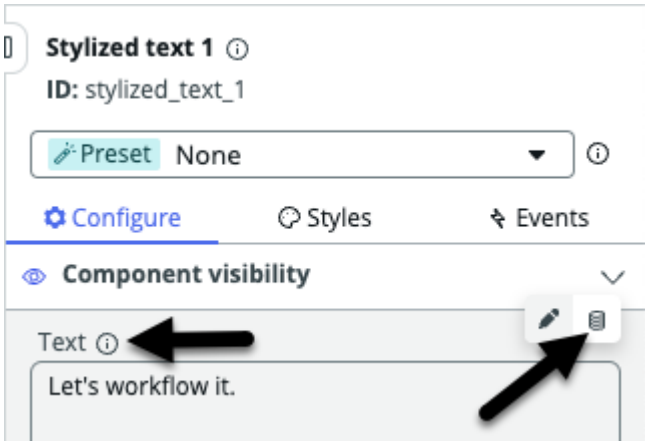
d. Select the **X** to close the window.



11. Bind the stylized text component to the client state parameter.

a. Select the stylized text component on the stage.

b. In the configuration panel, point to the **Text** field and select the **Bind data or use scripts** icon.



c. On the **Data types** tab, select **Client states**.

d. Double-click on the **count** pill.

e. Select **Apply**.

The **Text** field changes to **count**.

12. Now configure the buttons to increase or decrease the number.

a. Select the top button on the stage.

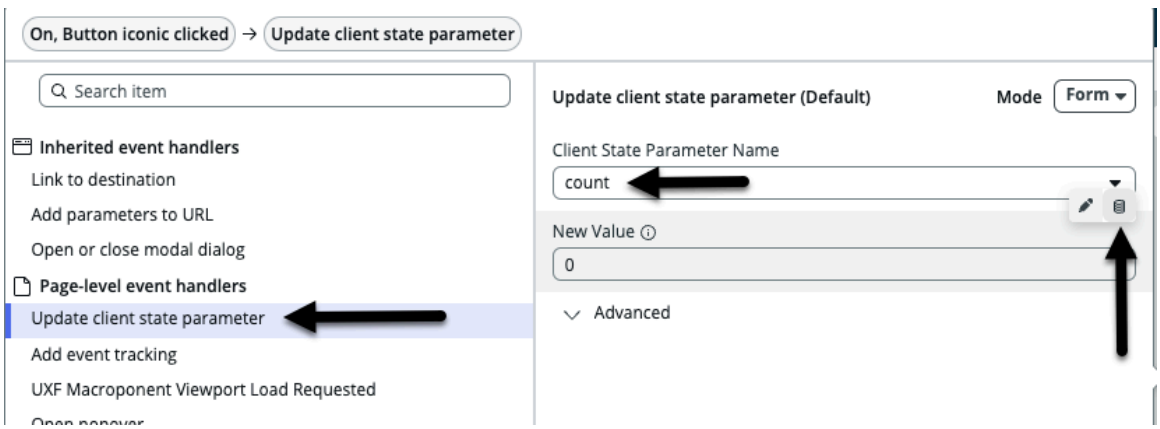
b. In the configuration panel, select the **Events** tab.

c. Select **+ Add event handler**.

d. Under **Page-level event handlers**, select **Update client state parameter**.

e. In **Client State Parameter Name**, select **count**.

f. Point to the **New Value** field and select the **Bind data or use scripts** icon.



g. On the **Data types** tab, select **Client states**.

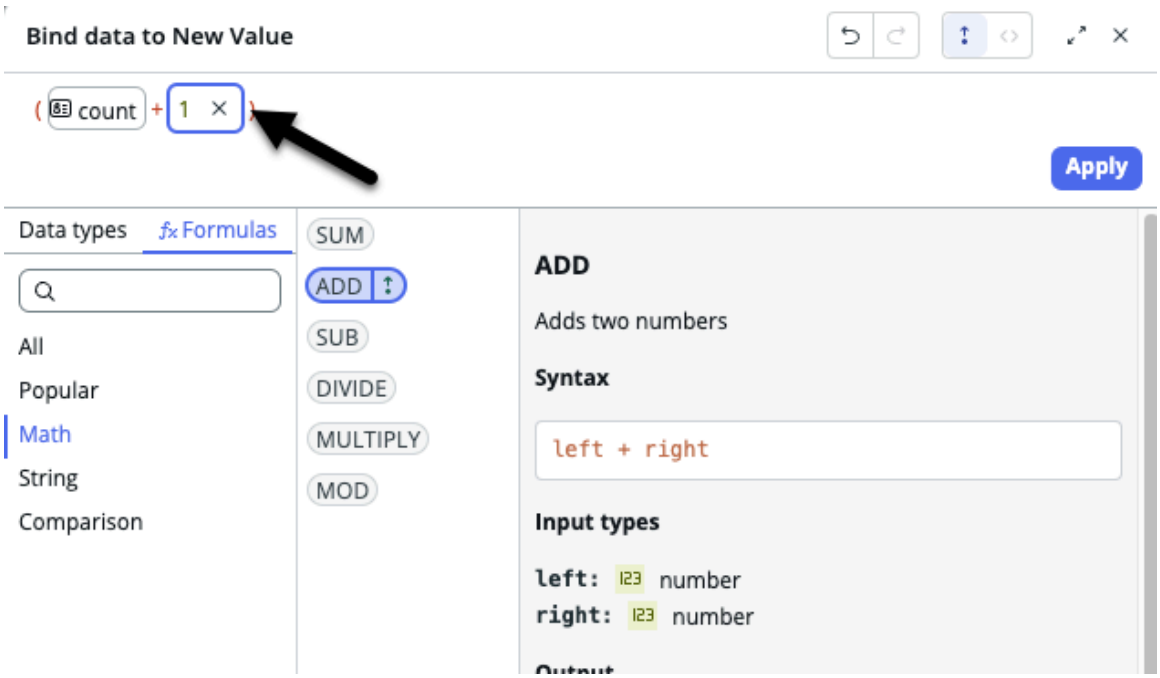
h. Double-click on the **count** pill.

i. Select the **Formulas** tab.

j. In the list, select **Math**.

k. Double-click on the **ADD** pill.

l. In the **right** pill at the top, remove the text and enter `1`.

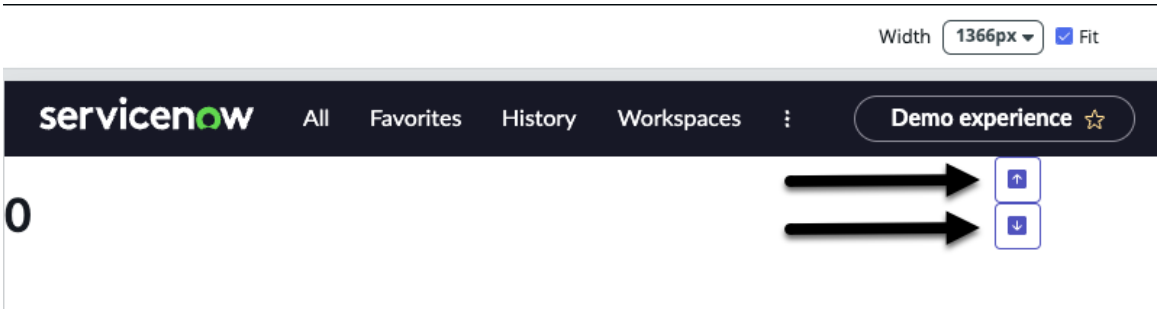


m. Select **Apply**.

n. Select **Add**.

13. Now configure the second button by following the same process that you used for the first button, but select the SUB pill instead of the ADD pill.
    a. Select the bottom button on the stage.
    b. In the configuration panel, select the **Events** tab.
    c. Select **+ Add event handler**.
    d. Under **Page-level event handlers**, select **Update client state parameter**.
    e. In **Client State Parameter Name**, select **count**.
    f. Point to the **New Value** field and select the **Bind data or use scripts** icon.
    g. On the **Data types** tab, select **Client states**.
    h. Double-click on the **count** pill.
    i. Select the **Formulas** tab.
    j. In the list, select **Math**.
    k. Double-click on the **SUB** pill.
    l. In the **right** pill at the top, remove the text and type `1`.
    m. Select **Apply**.
    n. Select **Add**.
14. Test the counter by selecting **Preview**.

15. Select the up arrow button to increase the count and the down arrow button to decrease the count.



Result

You added a button component to increase the count by one and another button component to decrease the count by one. You added the stylized text component to show the count as it increased and decreased.

For detailed, technical information about the button component, see Button Overview & API.

For detailed, technical information about the stylized text component, see Stylized Text Overview.

Update a component's state using client state parameters

Create and bind a client state parameter value to a component in UI Builder. By adding custom values to your components, these components can then be automatically updated through a script.

Before you begin

Role required: ui_builder_admin

About this task

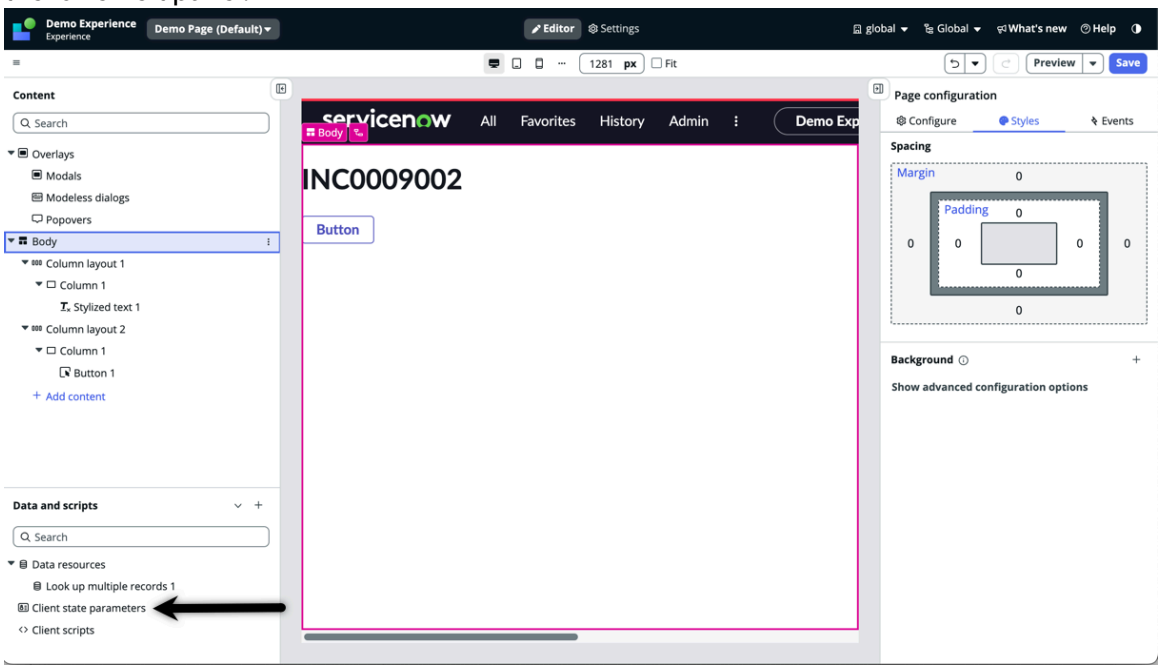
You can add two components to a page and then connect them by using client state parameters and scripts. Start by creating a client state parameter and then binding the client state parameter to one of the components, such as a label. Next, you create a client script and bind it to a second component, such as a button, using an event handler. When you create the event handler, bind the client state parameter value to it to connect the two components. If you click one component, it changes the state of the other component. Client states are useful because you can add custom values to your components that can be automatically updated through a script. Think of a client state as a bucket for storing information that is specific to the page.

For example, you can add a button and label component to your page. The button changes the value of the label, such as changing the text color.

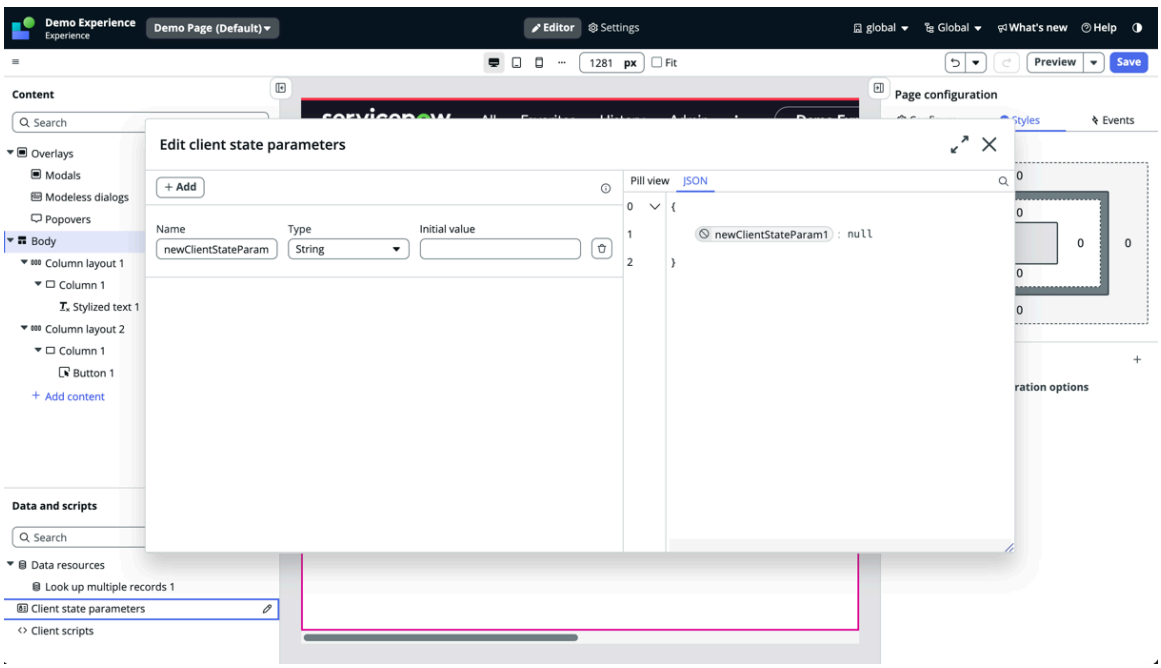
Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create or open a page.
   For more information, see Manage UI Builder pages and page variants.
4. Add two components to your page.
   For more information, see Add and configure components.

5. To define a client state parameter with an associated value, select Client state parameters in the lower-left panel.

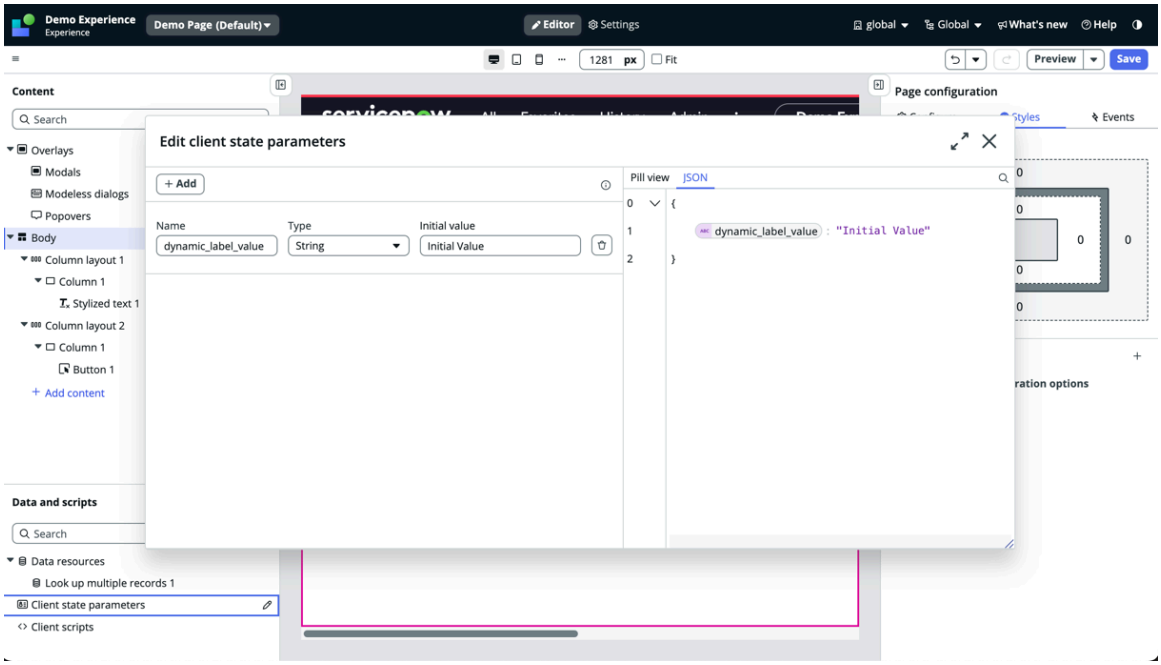


6. Select **+ Add**.



7. Enter a name for the client state name, type, and initial value.

   The client state supports Strings, Numbers, Boolean, and JSON. For example, you could enter the name as `dynamic_label_value`, the type as `String`, and the initial value as `Initial Value`.

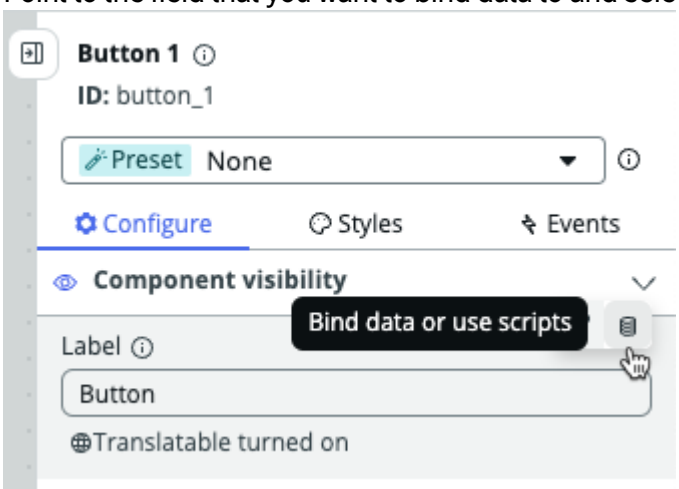


8. Bind the value of the client state parameter to your component.

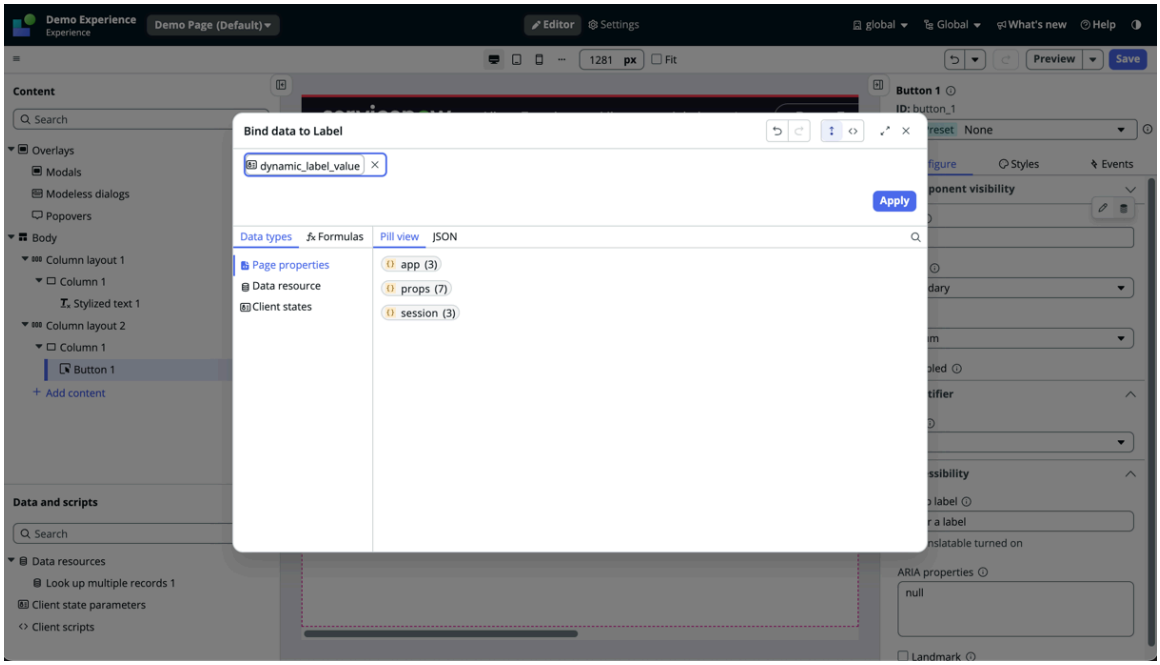
   a. Select the component that you want to bind the client state parameter to.

   b. Open the configuration panel and select the **Configure** tab.

   c. Point to the field that you want to bind data to and select the dynamic data binding icon ( ).

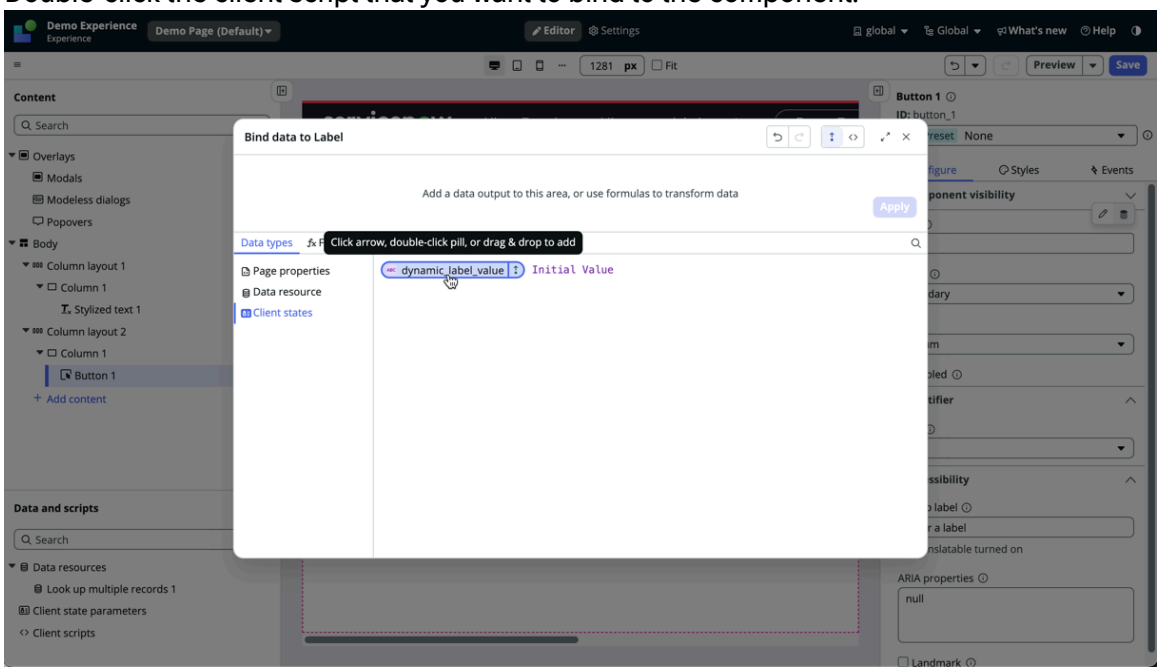


   The data binding modal appears.

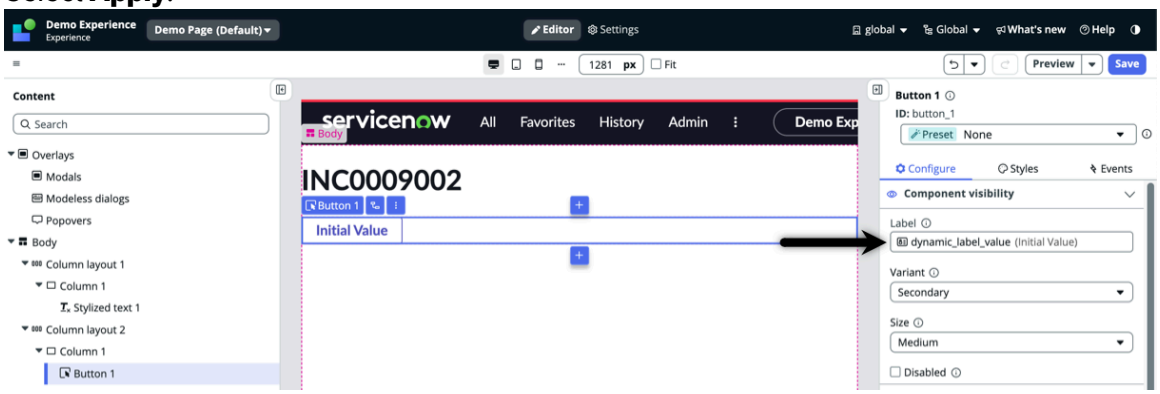


d. Select **Client states** in the **Data types** tab.

e. Double-click the client script that you want to bind to the component.



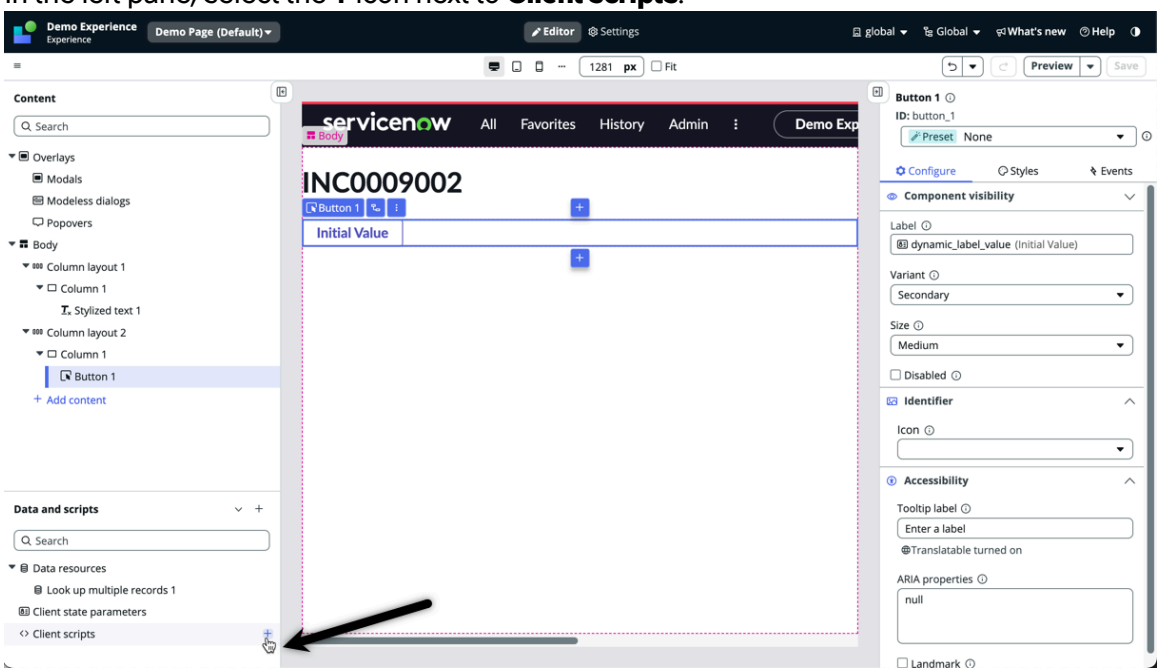
f. Select **Apply**.



9. Select **Save**.

10. Bind the client state value to one component and create a script to connect the second component to the first.

a. In the left pane, select the **+** icon next to **Client scripts**.

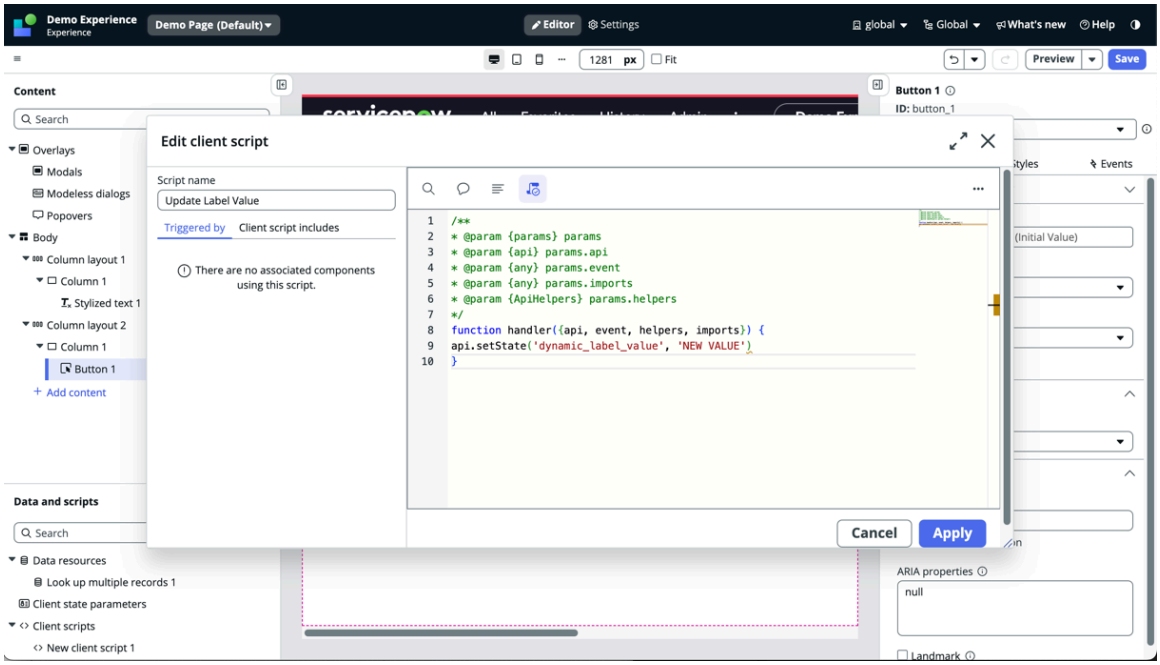


b. Enter a script name that describes the task.
For example, you could enter `Update Label Value` because that is what you want the script to do.

c. In the Now Code Editor, add your script.

Let's say that you choose an API to call, and the parameters for the API, such as a state and a value. For example, you could call the `api.setState`, and include the

`dynamic_label_value` as the first parameter, and a `NEW VALUE` as the second parameter.



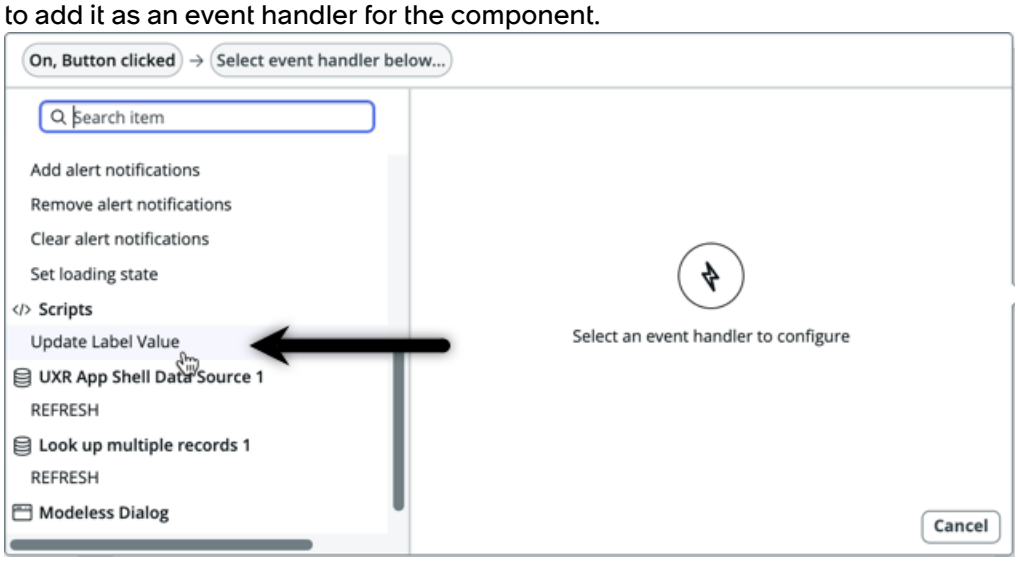
11. Add an event handler to your second component to call the new client script that you created.

a. Select the component on your page that you want to bind the script to.

b. Select the **Events** tab.

c. Select **+ Add a new event handler**.

d. Under the Scripts section of event handlers, select the script that you created, and select **+** to add it as an event handler for the component.



12. Select **Save**.

13. To preview your page and test the components to ensure they're connected, select **Preview**. When you select one component, it should change the state of the second component. For example, select the **Button** component to change the **Label value** component text from **Initial Value** to **NEW VALUE**.

Result

What you did in this procedure:

- Added two components to your page.
- Defined a client state parameter that had an initial value.
- Bound the client state parameter to the first component.
- Created a client script that updates a value that is defined by the client state parameter.
- Created an event handler for the second component to call the new client script that you just created
- Selected the second component so that it changed the state of the first component.

Define and bind client scripts to components

Add and edit client scripts in UI Builder so that you can update the client state through events. You can bind these scripts to any component by using an event handler.

Before you begin

Role required: ui_builder_admin

About this task

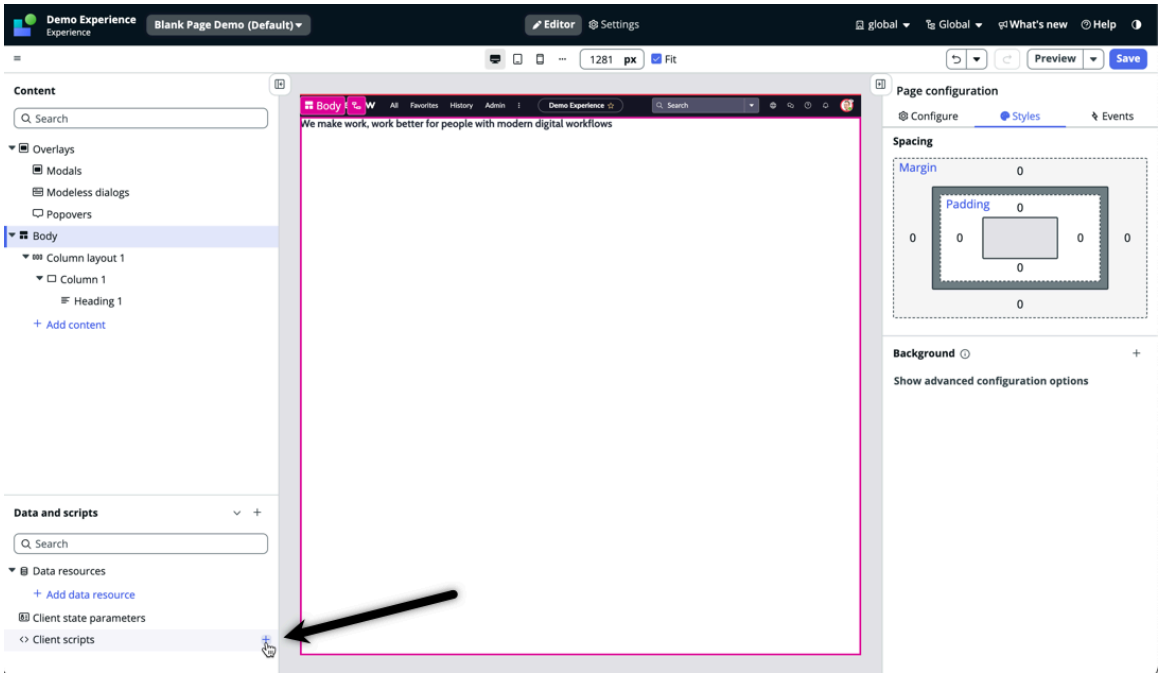
You can create JavaScript client scripts in UI Builder by using the Now Code editor. Then, you can add the client script as an event handler to update the client state, emit a handled event of your page, or execute a data resource operation. For example, you could write a script to increment a date or counter, and bind the script to a component event, like a button click. For more information about the Now Code Editor, see Edit code with the Now Code Editor (advanced feature).

With these scripts, you can do the following actions:

- Get available data, manipulate the data, and store it in the client state.
- Access data resource results.
- Execute data resource operations.
- Dispatch events.

Procedure

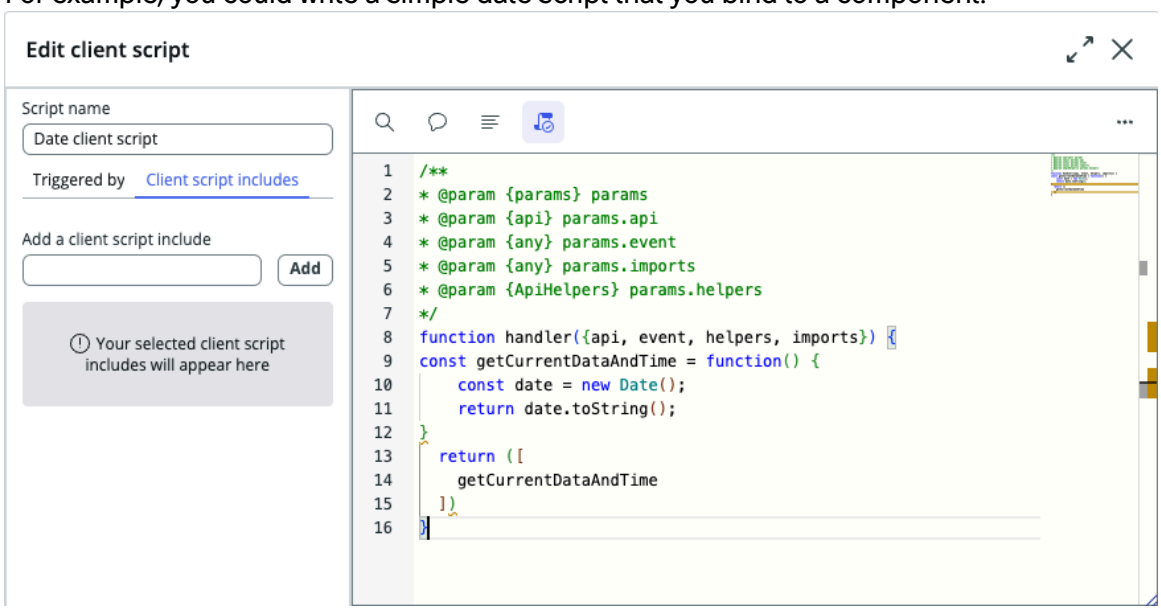
1. Navigate to **All** > **Now experience framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**.
   For more information, see Configure how users interact with your applications in UI Builder.
3. Open or create a page.
   For more information about how to create a page in UI Builder, see Create a page in UI Builder.
4. **Optional:** If you do not have any components on your page, add a component to your page.
   For example, you can add a Heading component. For more information, see Add and configure components.
5. Select **+** next to **Client scripts**.



6. Name your script.

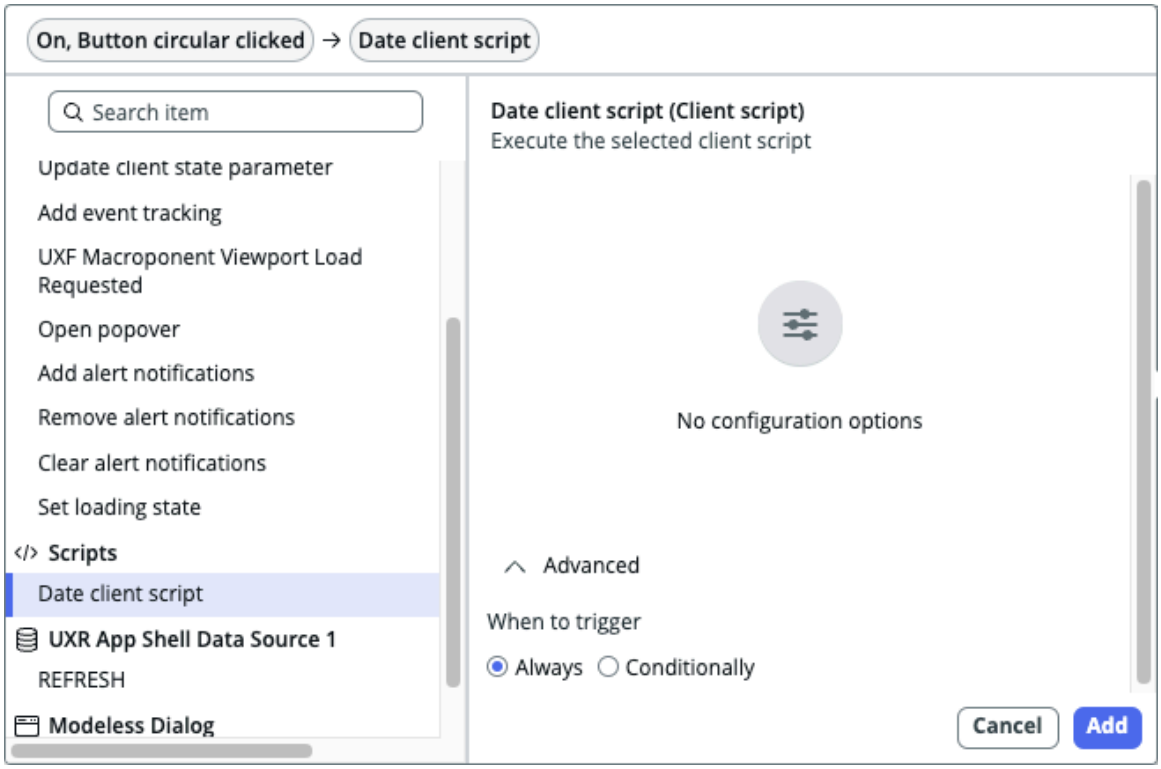
   A descriptive name helps you know what the script does. It also makes it easier to find the script later when you add it to an event handler. The following example shows a simple date client script.

7. Write your script to perform an action.
   For example, you could write a simple date script that you bind to a component.



8. **Optional:** Add a **Script include** or **Associated components**, which shows up in the `imports` parameter of your client script function.
9. Select the component on your page that you want to bind the script to and then select **Events**.
10. Select **+Add a new event handler**, select the script that you created under **Scripts**, and then select **Add**.

    The following example shows a date client script.



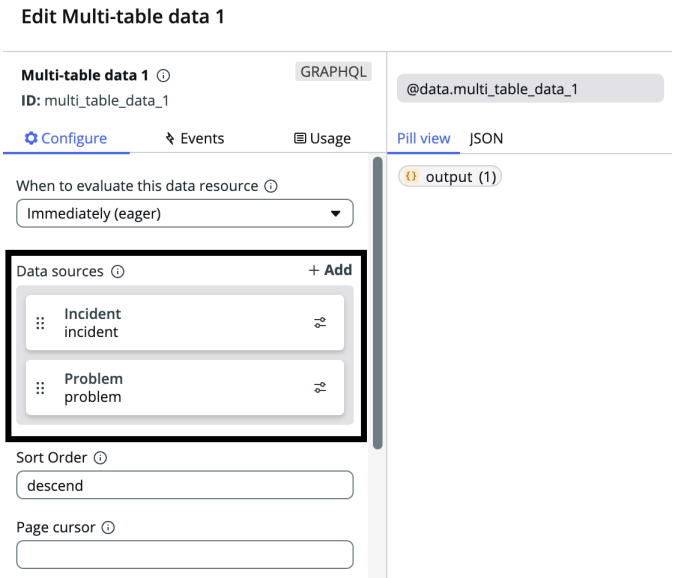
11. Select **Save**.
12. To preview your scripted component, select **Preview**.

Multi-table data configuration

Present data from multiple tables using components and control the layout and styling.

A data resource fetches the data that UI Builder uses to display information in a component. Most often, a component is mapped to data contained in one table. However, there are numerous use cases where there's a need to map a single component to information in multiple tables, for example:

- Place the card component in a repeater and use multi-table data configuration to pull data from different sources to display on the cards.
- Create a portal page that displays a list of tasks pulled from different locations.



Note:
You can configure more than one multi-table data resource on a single page to display different sets of data.

Multi-table data configuration and Entity View Action Mapper

The Entity View Action Mapper (EVAM) can also be used to configure cards and lists using different data sources. The EVAM is an option for users who understand and are comfortable working with JSON. The multi-table data configuration option gives you greater control over the presentation of data, makes it easier to configure the data, and enables you to remain in UI Builder. Use the option that you prefer. For more information about EVAM, see Work with the Entity View Action Mapper for UI Builder.

Fetch data from multiple sources

Learn how to obtain data from different sources to use in a single component.

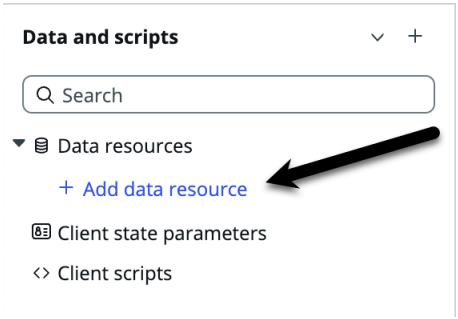
Before you begin
Role required: ui_builder_admin

About this task
Place the card base container component in a repeater and use multi-table data configuration to present all active incidents and problems created in the last two years.

Procedure

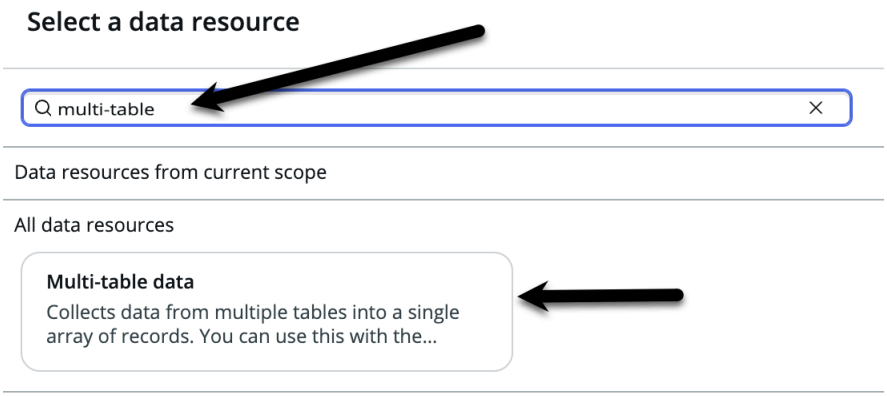
1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create > Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Create a page from scratch.
   For more information about how to create a page, see Create a page in UI Builder.
4. Add a multi-table data resource.

a. In the data shelf, select **+ Add data resource**.



b. In **Search**, enter `multi-table`.

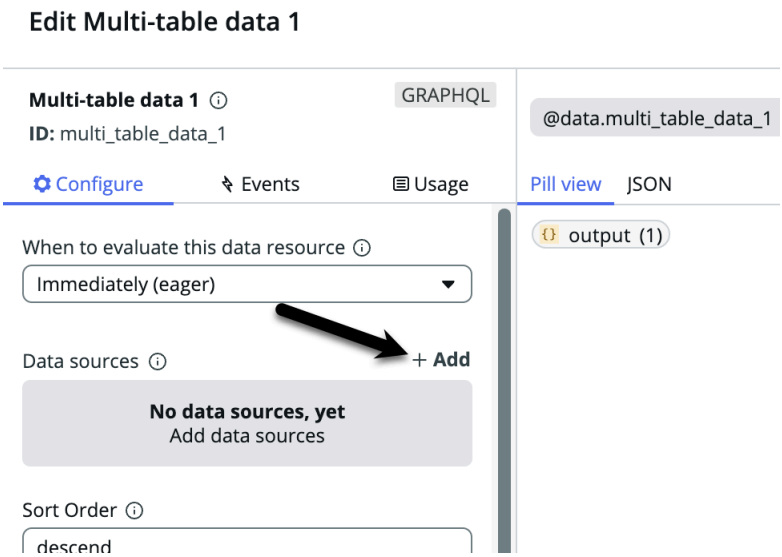
c. Select **Multi-table data**.



d. Select **Add**.

5. Configure the first data source.

a. In **Data sources**, select **+ Add**.

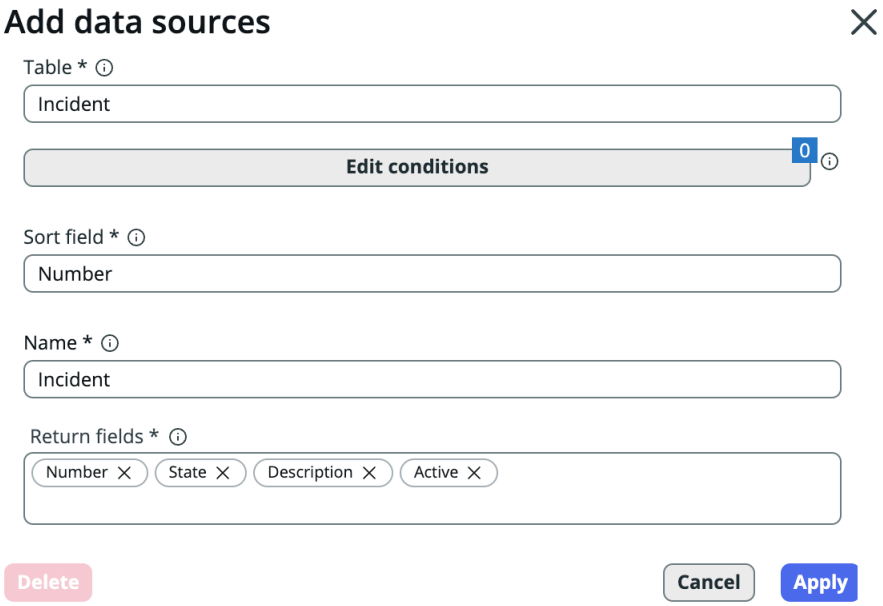


b. In **Table**, enter `incident` and select the **Incident** table.

c. In **Sort field**, enter `number` and select **Number**.

d. In **Name**, enter `Incident`.

e. In **Return fields**, add **Number**, **State**, **Description**, and **Active**.

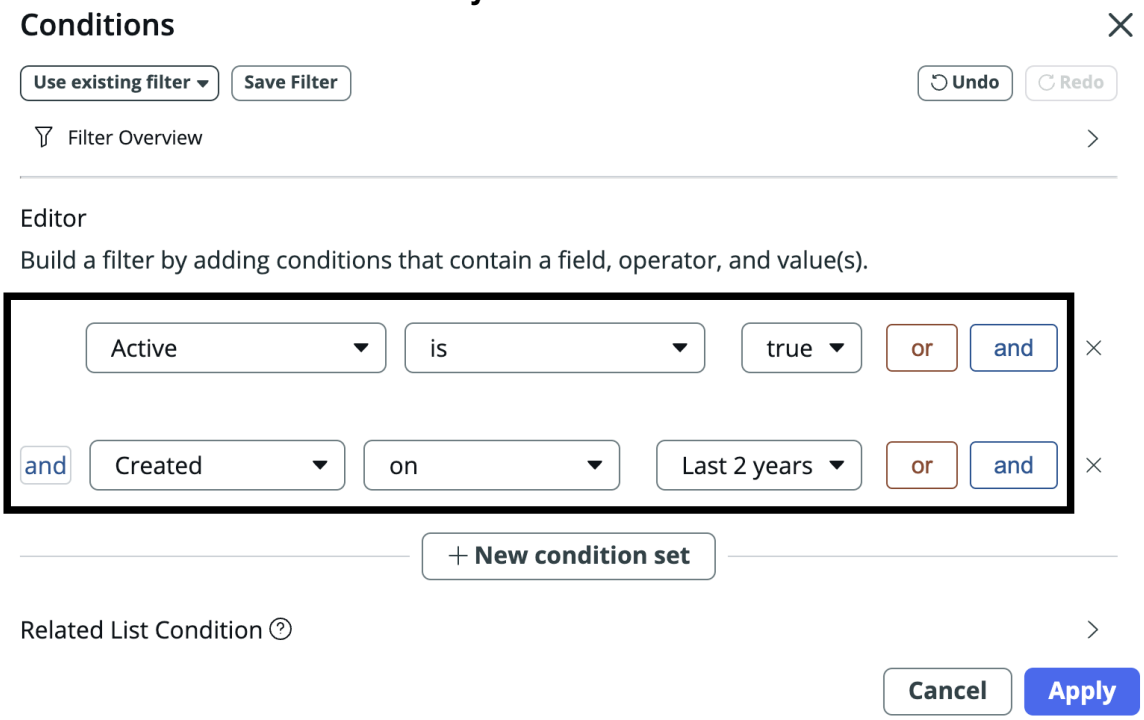


f. Select **Edit conditions**.

g. Build the condition **Active is true**.

h. Select **and**.

i. Build the condition **Created on Last 2 years**.



j. Select **Apply**.

k. Select **Apply**.

6. Configure the second data source.

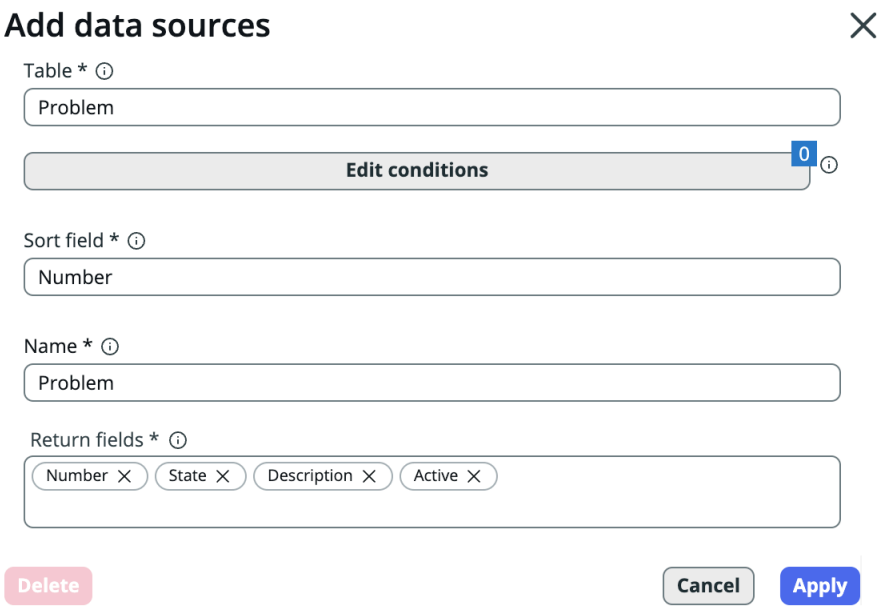
a. In **Data sources**, select **+ Add**.

b. In **Table**, enter `problem` and select the **Problem** table.

c. In **Sort field**, enter `number` and select **Number**.

d. In **Name**, enter `Problem`.

e. In **Return fields**, add **Number**, **State**, **Description**, and **Active**.
When creating a multi-table data resource, each data source should have the same return fields specified and in the same order. This helps ensure that the data displayed on the page is consistent and accurate.



f. Select **Edit conditions**.

g. Build the condition **Active is true**.

h. Select **and**.

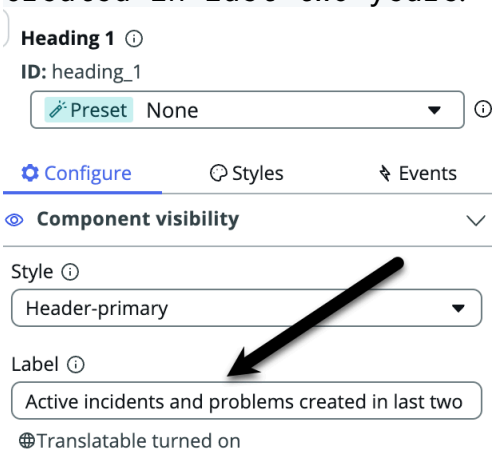
i. Build the condition **Created on Last 2 years**.

j. Select **Apply**.

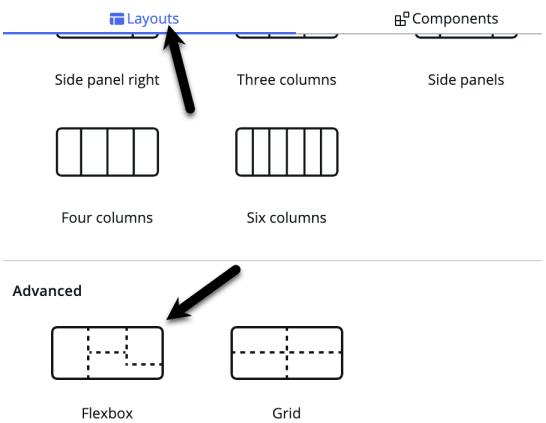
k. Select **Apply**.

l. Select the **X** to close the **Edit Multi-table data** pop-up.

7. Select **Save**.
8. Add the heading component.
   a. In the content tree, select **+ Add content** under **Body**.
   b. In **Search** enter `heading`.
   c. Select the **Heading** component.
   d. In the configuration panel, on the **Configure** tab, select **None - Configure the component manually**.
   e. In **Label**, delete the default text and enter `Active incidents and problems created in last two years`.

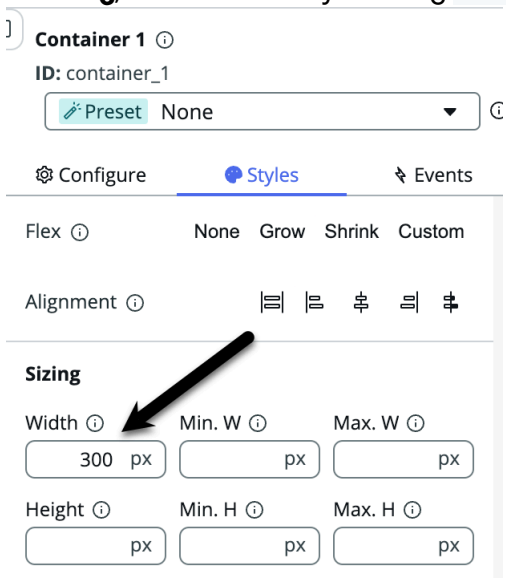


   f. Select **Save**.
9. Add a container.
   a. In the content tree, select **+ Add content** under **Heading 1**.
   b. On the **Layouts** tab, in the **Advanced** section, select **Flexbox**.



   c. In the configuration panel, on the **Configure** tab, select **None - Configure the component manually**.

d. In **Sizing**, set the **Width** by entering `300`.



e. Select **Save**.

10. Add the repeater component.

a. In the content tree, select **+ Add content** under **Container 1**.

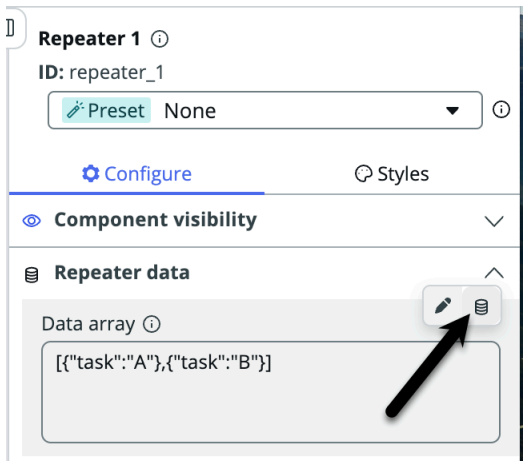
b. In **Search** enter `repeater`.

c. Select the **Repeater** component.

d. In the configuration panel, on the **Configure** tab, select **None - Configure the component manually**.

e. Select **Save**.

f. In the configuration panel, on the **Configure** tab, hover over the **Data array** field and select the bind data icon.

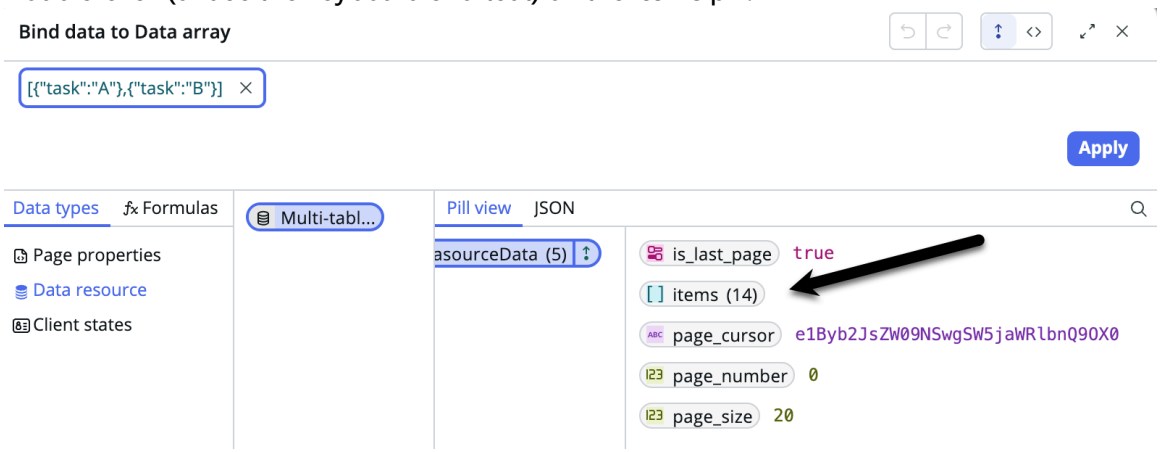


g. Under **Data types** select **Data resource**.

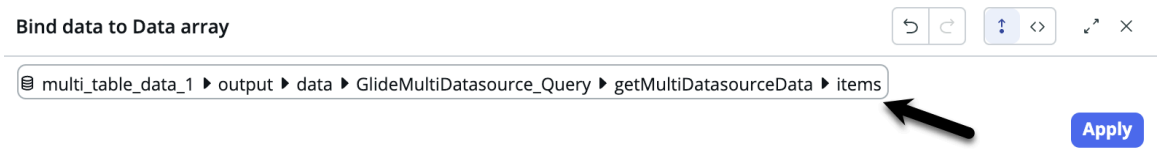
h. Select **Multi-table data 1**.

i. Select **output** > **data** > **GlideMultiDatasource_Query** > **getMultiDatasourceData**.

j. Double-click (or use the keyboard shortcut) on the **items** pill.



Check that the top section is accurate.



k. Select **Apply**.

l. Select **Save**.

11. Add the card base container component within the repeater.

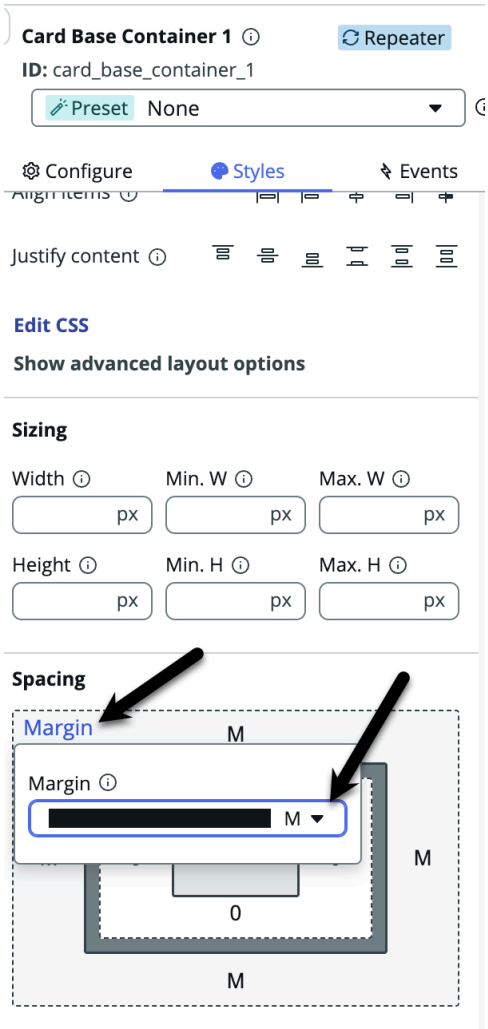
a. In the content tree, select **+ Add content** under **Repeater 1**.

b. Locate and select the **Card Base Container** component.

c. In the configuration panel, on the **Configure** tab, select **None - Configure the component manually**.

d. Select the **Styles** tab.

e. To add a little space around the cards, go to **Spacing**, select **Margin**, and then select **M** (for medium).



f. Select **Save**.

12. Add the heading component within the card base container.

a. In the content tree, select **+ Add content** under **Card Base Container 1**.

b. Locate and select the **Heading** component.

c. In the configuration panel, on the **Configure** tab, select **None - Configure the component manually**.

d. In the configuration panel, on the **Configure** tab, delete the default text in **Label**.

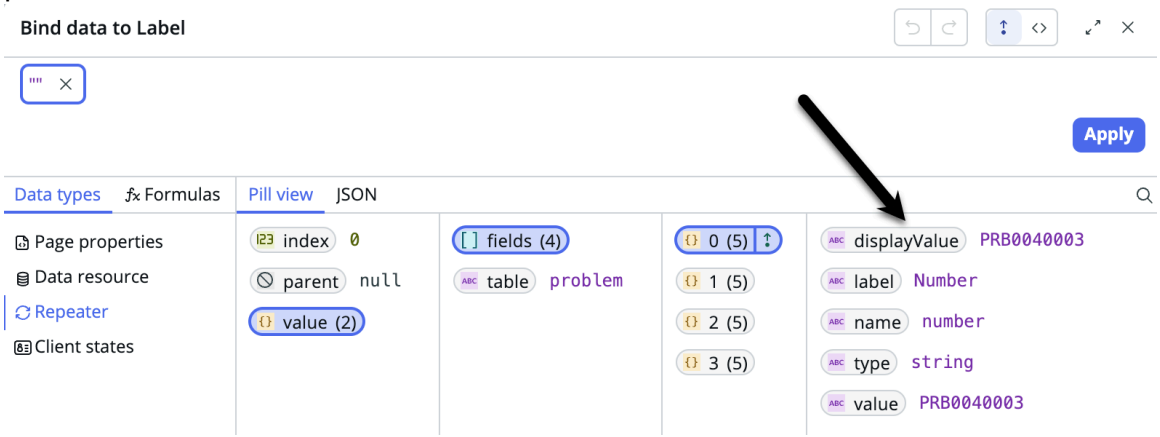
e. Point to the **Label** field and select the bind data icon.

f. Under **Data types** select **Repeater**.

g. Select **value** > **fields**.

h. Select the top pill and in the last column check for **displayValue** with an incident or problem number.

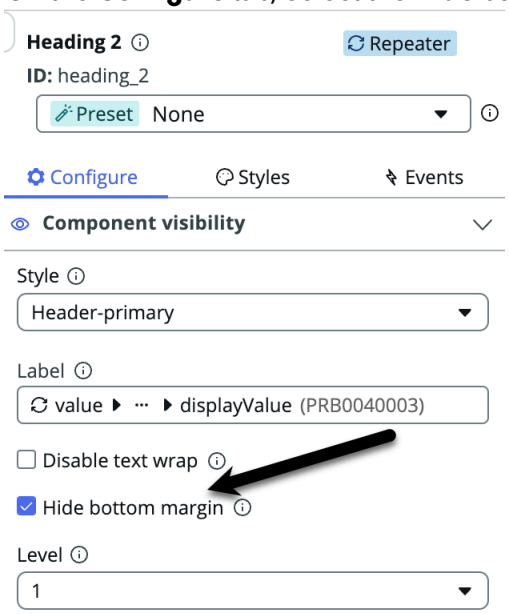
i. Double-click (or use the keyboard shortcut) on the **displayValue** pill with an incident or problem number.



Check that the top section is accurate.

j. Select **Apply**.

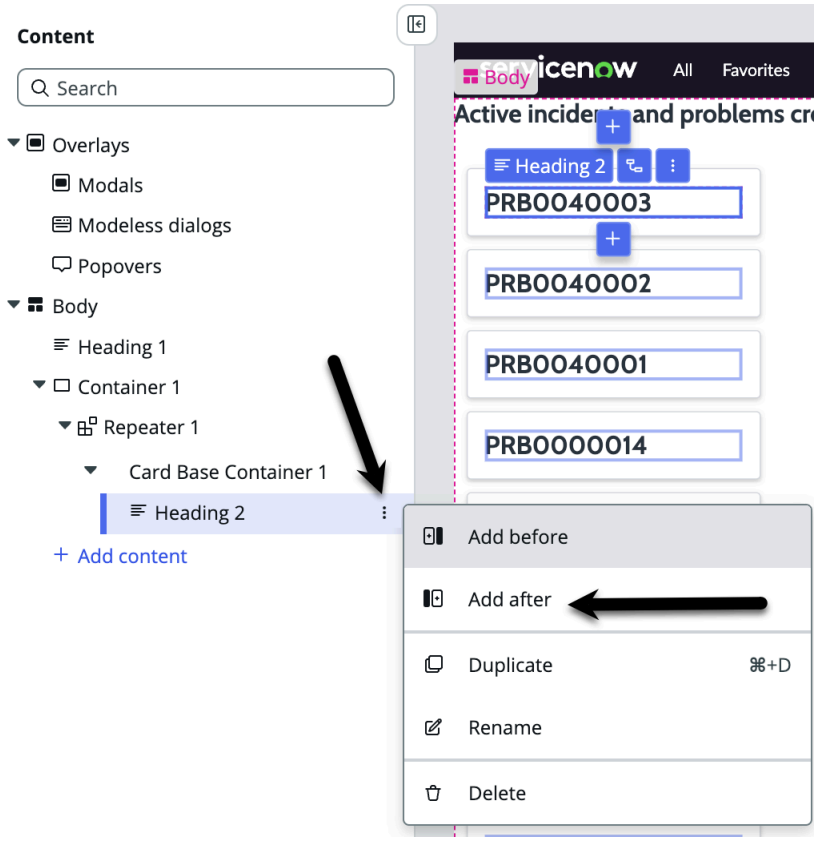
k. On the **Configure** tab, select the **Hide bottom margin** option.



l. Select **Save**.

13. Add the first stylized text component to show the description of the incident or problem.

a. In the content tree, point to **Heading 2**, select the menu (three vertical dots) icon, and select **Add after**.



b. Locate and select the **Stylized text** component.

c. In the configuration panel, on the **Configure** tab, select **None - Configure the component manually**.

d. In the configuration panel, on the **Configure** tab, delete the default text in **Text**.

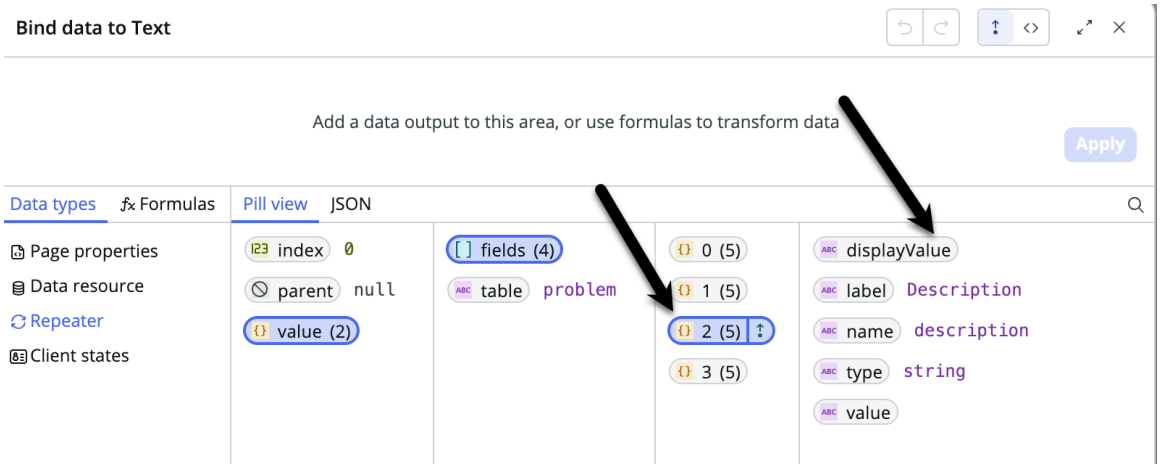
e. Point to the **Text** field and select the bind data icon.

f. Under **Data types** select **Repeater**.

g. Select **value** > **fields**.

h. Select the third pill in the list and in the last column check that the **displayValue** contains no information.

i. Double-click (or use the keyboard shortcut) on the **displayValue** pill with no value.



Check that the top section is accurate.

j. Select **Apply**.

k. To reduce the text size, go to the **Configure** tab and change the **HTML tag** to **Paragraph**.

l. Select **Save**.

14. Add the second stylized text component to show the state of the incident or problem.

a. In the content tree, point to **Stylized text 1**, select the menu (three vertical dots) icon, and select **Add after**.

b. Locate and select the **Stylized text** component.

c. In the configuration panel, on the **Configure** tab, select **None - Configure the component manually**.

d. In the configuration panel, on the **Configure** tab, delete the default text in **Text**.

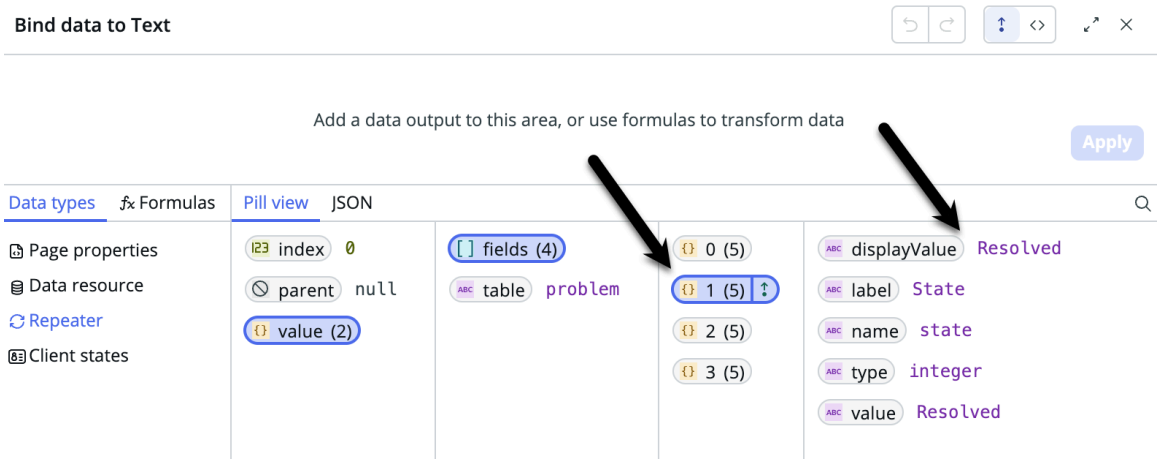
e. Point to the **Text** field and select the bind data icon.

f. Under **Data types** select **Repeater**.

g. Select **value** > **fields**.

h. Select the second pill in the list and in the last column check that the **displayValue** contains a state such as **Resolved** or **In Progress**.

i. Double-click (or use the keyboard shortcut) on the **displayValue** pill with a state value.



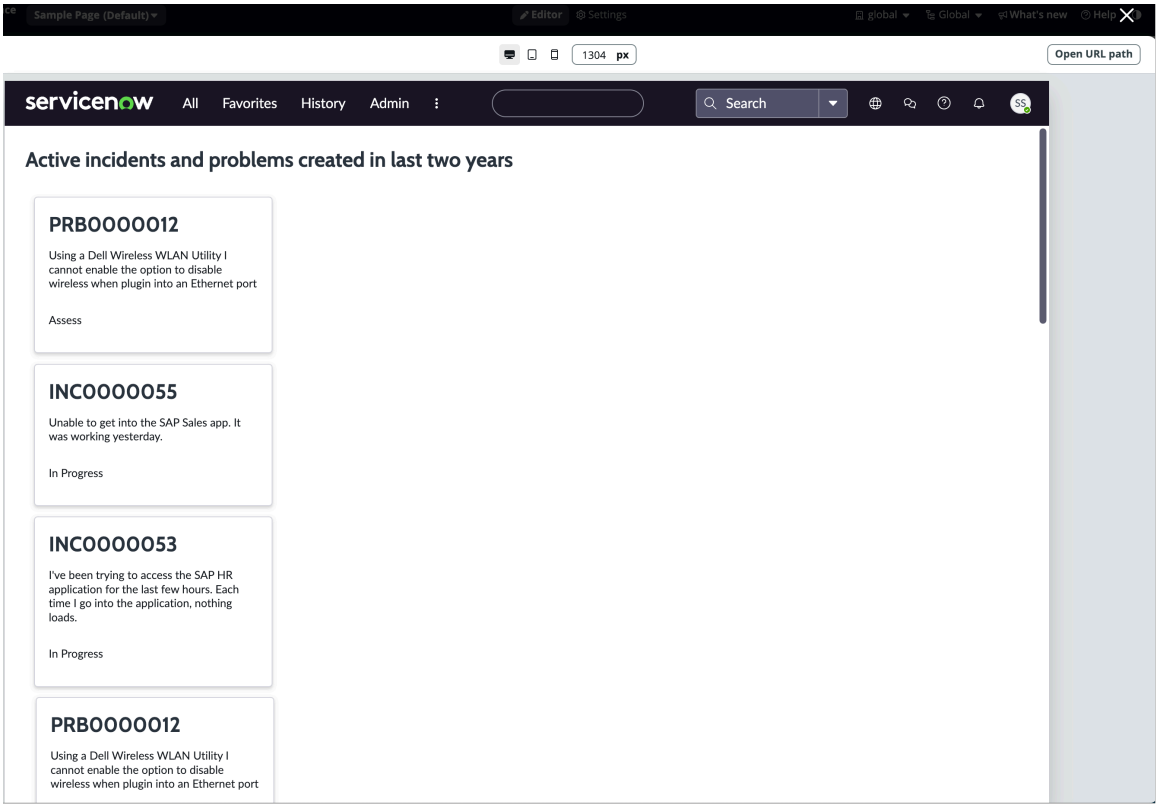
Check that the top section is accurate.

j. Select **Apply**.

k. To reduce the text size, go to the **Configure** tab and change the **HTML tag** to **Paragraph**.

l. Select **Save**.

15. Select **Preview**.
The page heading is at the top. Each card contains a heading with the incident or problem number. The cards also show the description (if one is available) of the incident or problem and the state. Data is being pulled successfully from two sources: the incident table and the problem table.



16. Select the **X** to close the preview overlay.

Work with the Entity View Action Mapper for UI Builder

With UI Builder, you can use the Entity View Action Mapper (EVAM) application to standardize how the data sources in your components are displayed in your cards and lists.

EVAM Overview

Entity View Action Mapper (EVAM) is an application that standardizes how different data sources are displayed in cards and lists. UI Builder uses EVAM data sources to show information in a component as a card grid view or as a list of information. You can add a toggle to your component to let users switch between card grid and list views.

EVAM consists of the following components:

- Entity (data source). Associated data that you intend to display, such as a community post or a user.
- View. How a card displays data and actions.
- Actions. Action that it performs on the card. For example, you can activate a user into your system.
- Map. A process that maps the data source to generic fields that are displayed on the card. You can also associate actions that trigger from the card view.

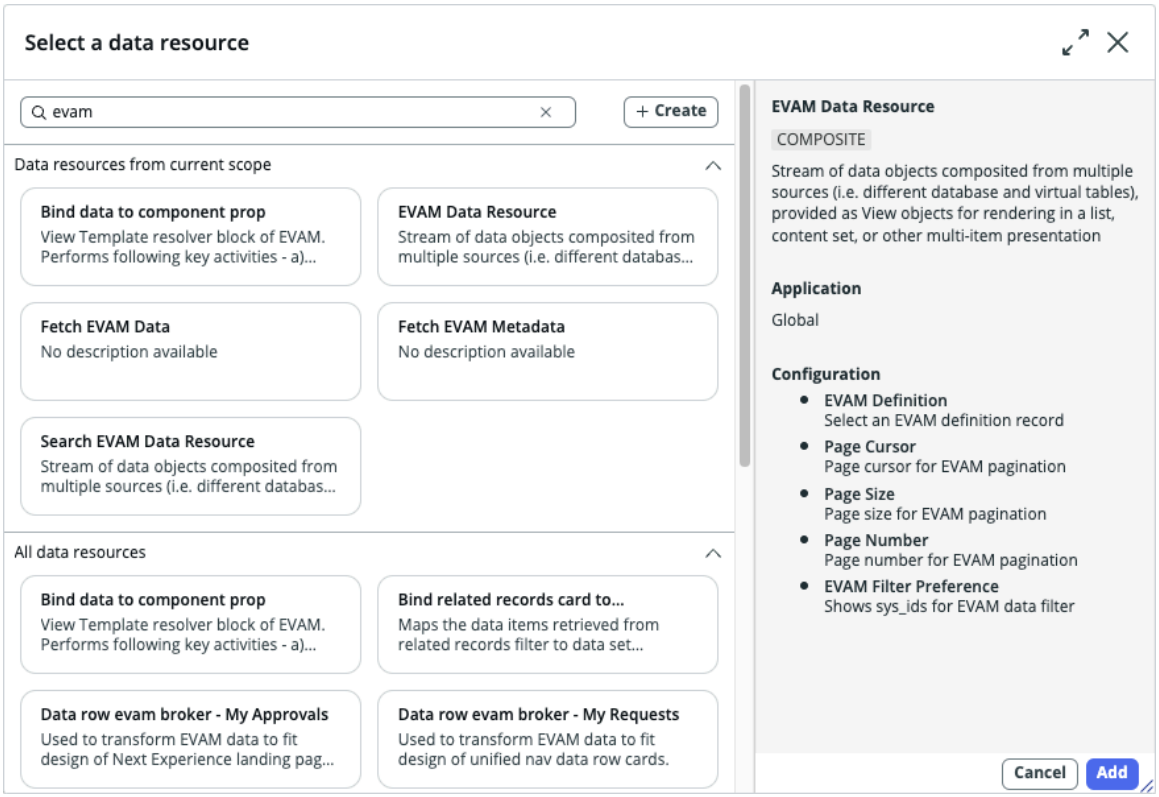
EVAM and Multi-table data configuration

The EVAM is an option for users who understand and are comfortable working with JSON. The multi-table data configuration option is an alternative that gives you greater control over the presentation of data, makes it easier to configure the data, and enables you to remain in UI Builder. Use the option that you prefer. For more information, see Multi-table data configuration.

EVAM data sources

You add EVAM data sources in UI Builder and bind them to a component.

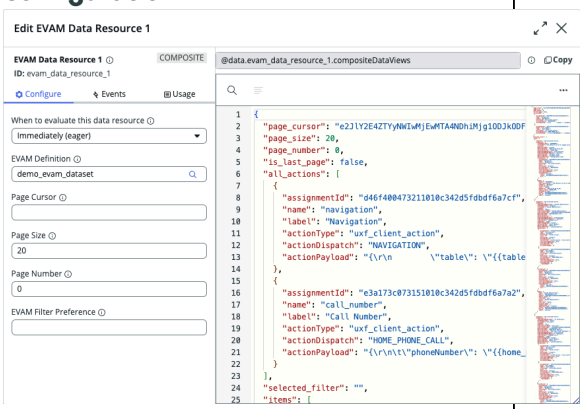
EVAM data resources



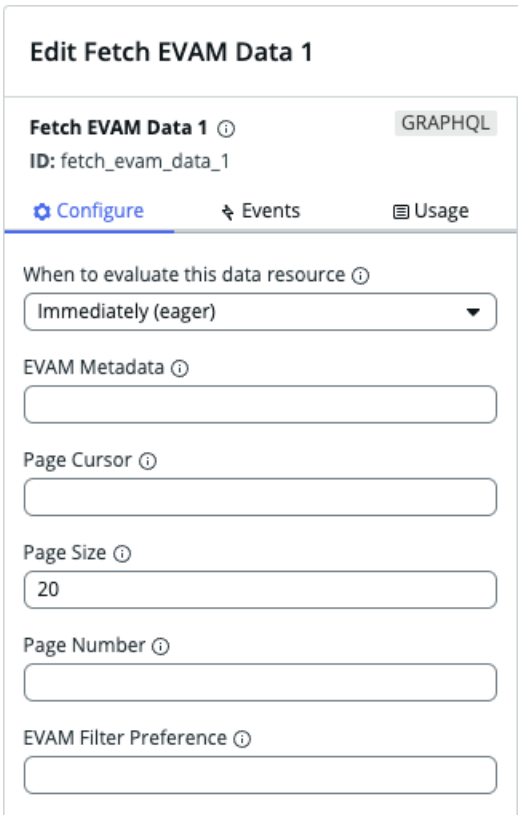
EVAM data sources

EVAM data source	Description
EVAM Data Resource	Add an EVAM definition and other information about the data source. • Type: Composite. • When to evaluate: In the **When to evaluate this data resource**, select **Immediately** to have the EVAM data resource instance evaluated on page load or select **Only when invoked** to use an event handler to evaluate the EVAM data resource. • EVAM definition: In the **EVAM Definition** field, add the EVAM definition record that is associated with the data resource. • Page cursor: In the **Page Cursor** field, add a page cursor for the EVAM definition. • Page size: In the **Page Size** field, add a size for the EVAM pagination.

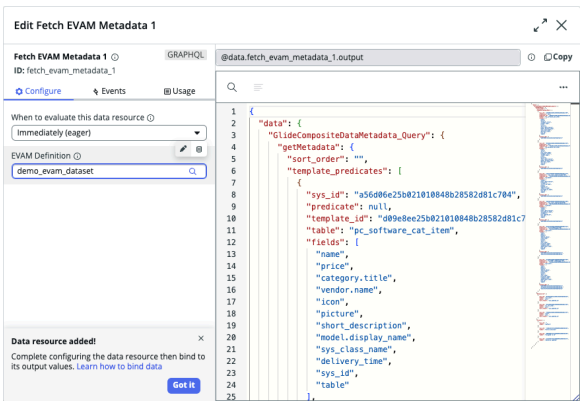
EVAM data sources (continued)

EVAM data source	Description
	• Page number: In the **Page Number** field, add the page number for the EVAM pagination. • EVAM Filter Preference: In the **EVAM Filter Preference** field, enter the sys_ids for the EVAM data filter. To add an event handler for when that data fetch is initiated, succeeded, or failed, select **Events**. In the **Now Code Editor** section, which is next to the Configuration pane, preview the EVAM definition script. **EVAM Data Resource configuration** 
Fetch EVAM Data	To add an EVAM definition and other information about the data source, select **Configuration**. • Type: GRAPHQL. • When to evaluate: To have the EVAM data resource instance evaluated on a page load, select **Immediately**. To use an event handler to evaluate the EVAM data resource, select **Only when invoked**. • EVAM definition: In the **EVAM Definition** field, enter the EVAM definition record that is associated with the data resource. • Page cursor: In the **Page Cursor** field, enter a page cursor for the EVAM definition. • Page size: In the **Page Size** field, enter a size for the EVAM pagination.

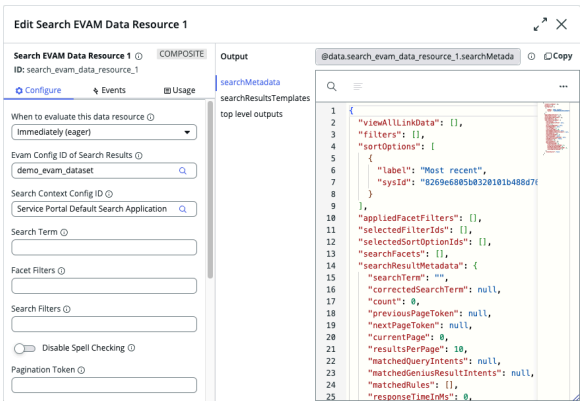
EVAM data sources (continued)

EVAM data source	Description
	• Page number: In the **Page Number** field, enter the page number for the EVAM pagination. • EVAM Filter Preference: In the **EVAM Filter Preference** field, enter the sys_ids for the EVAM data filter. To add an event handler for when that data fetch is initiated, succeeded, or failed, select **Events**. Preview the EVAM definition script in the **Now Code Editor** to the right of the Configuration pane. 
Fetch EVAM Metadata	To select when to evaluate the EVAM data resource and add an EVAM definition, select **Configuration**. • Type: GRAPHQL. • When to evaluate: To have the EVAM data resource instance evaluated on a page load, select **Immediately**. To use an event handler

EVAM data sources (continued)

EVAM data source	Description
	to evaluate the EVAM data resource, select **Only when invoked**. • EVAM definition: In the **EVAM Definition** field, enter the EVAM definition record that is associated with the data resource. To add an event handler for when that data fetch is initiated, succeeded, or failed, select **Events**. Preview the EVAM definition script in the **Now Code Editor** to the right of the Configuration pane. 
Search EVAM Data Resource	To add an EVAM definition, and other information about the data source, select **Configuration**. • Type: Composite. • When to evaluate: To have the EVAM data resource instance evaluated on a page load. select **Immediately**. To use an event handler to evaluate the EVAM data resource, select **Only when invoked**. • EVAM Config ID: In the **EVAM Config ID** field, add the **EVAM Config ID** that is associated with the data record. • Search Context Config ID: In the **Search Context Config ID** field, add the **Search Context Config ID** that you are searching for. • Search Term: In the **Search Term** field, add the search terms that you are searching for. • Facet Filters: In the **Facet filters** field, add any facet filters to help define your search. • Search Filters: In the **Search Filters** field, add any search filters to help define your search.

EVAM data sources (continued)

EVAM data source	Description
	• Disable Spell Checking: In the **Disable Spell Checking** field, select if you desire spell checking to be on or off during your search. • Pagination Token: In the **Pagination Token** field, enter a pagination token if needed. To add an event handler for when that data fetch is initiated, succeeded, or failed, select **Events**. Preview the EVAM definition script in the **Now Code Editor** to the right of the Configuration pane. 

Add an Entity View Action Mapper data resource to a page

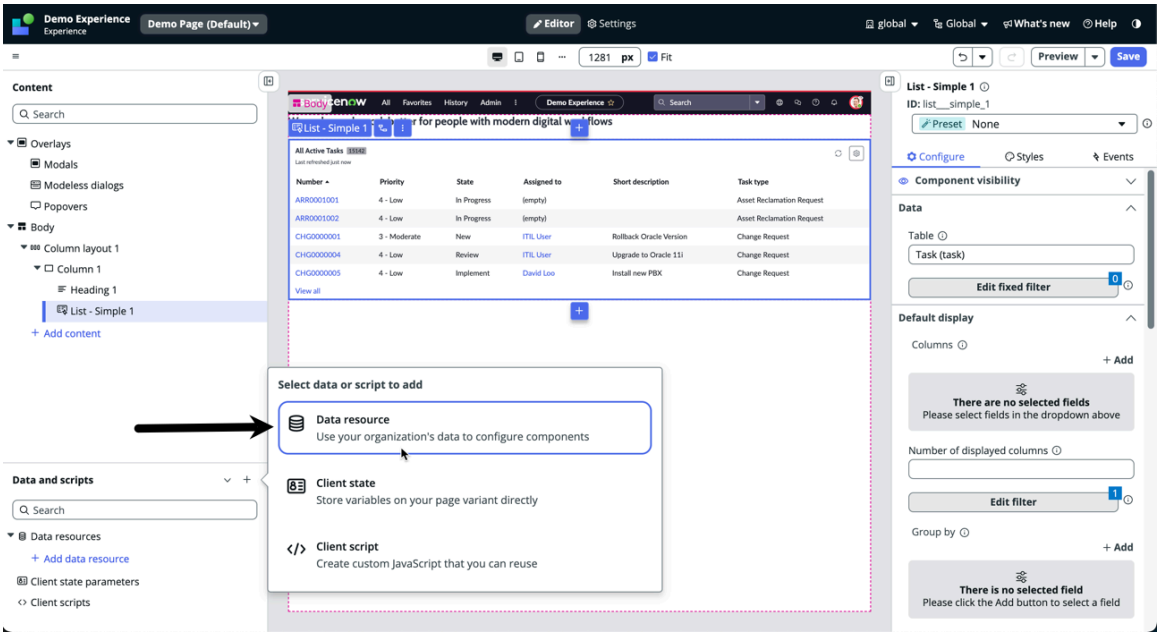
Add an Entity View Action Mapper (EVAM) data resource to your page in UI Builder so that you can standardize how the data sources in your components are displayed in your cards and lists.

Before you begin

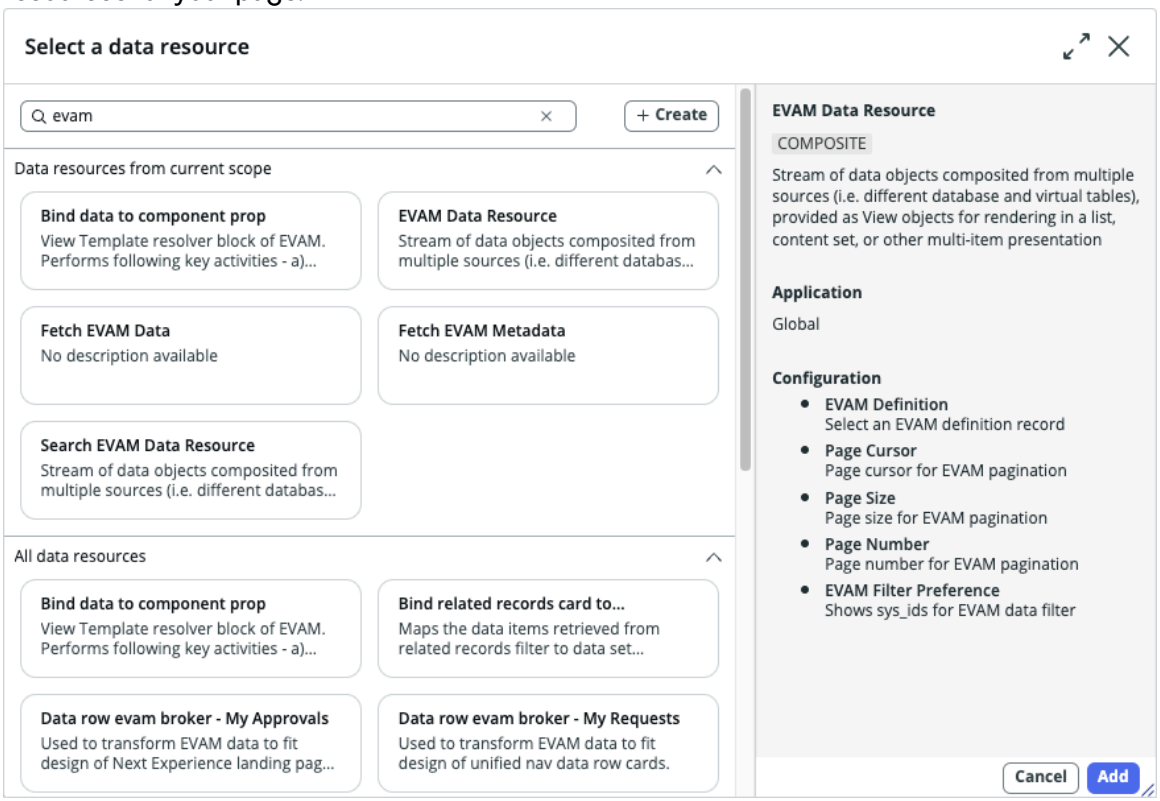
Role required: ui_builder_admin

Procedure

1. Navigate to **All** > **Now experience framework** > **UI Builder**.
2. Open an experience to work in or create an experience by selecting **Create** > **Experience**. See Configure how users interact with your applications in UI Builder for more information on creating experiences.
3. Open or create a page.
   For more information, see Manage UI Builder pages and page variants.
4. **Optional:** If you do not have any components on your page, add a **Data set** component to your page.
   For more information, see Add and configure components.
5. In the **Data and scripts** panel, select the **+** icon and then select **Data resource**.



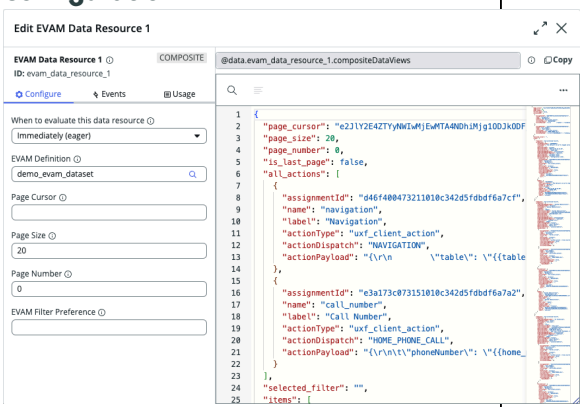
6. Enter `evam` in the search field.
7. Select the EVAM data resource that you want to add and repeat this step for all the EVAM data resources for your page.

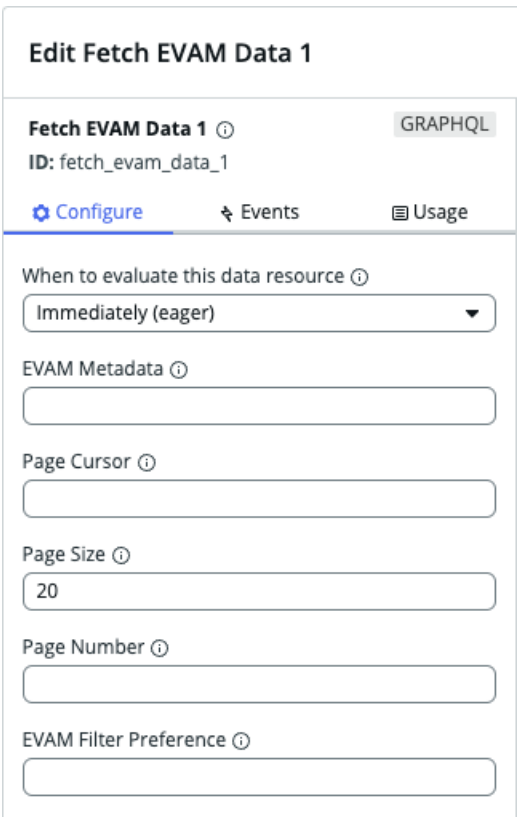


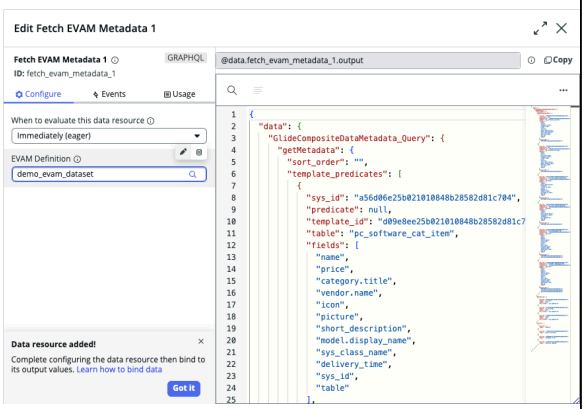
8. Configure each EVAM data resource.

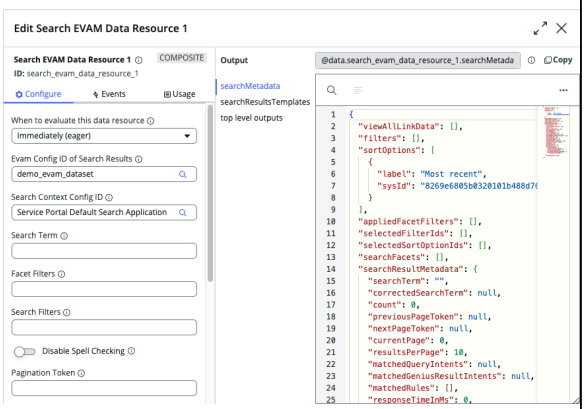
EVAM data sources

EVAM data source	Description
EVAM Data Resource	Add an EVAM definition and other information about the data source.

EVAM data source	Description
	○ Type: Composite. ○ When to evaluate: In the When to evaluate this data resource list, select **Immediately** to have the EVAM data resource instance evaluated on page load or select **Only when invoked** to use an event handler to evaluate the EVAM data resource. ○ EVAM definition: In the **EVAM Definition** field, add the EVAM definition record that is associated with the data resource. ○ Page cursor: In the **Page Cursor** field, add a page cursor for the EVAM definition. ○ Page size: In the **Page Size** field, add a size for the EVAM pagination. ○ Page number: In the **Page Number** field, add the page number for the EVAM pagination. ○ EVAM Filter Preference: In the **EVAM Filter Preference** field, add the sys_ids for the EVAM data filter. To add an event handler for when that data fetch is initiated, succeeded, or failed, select **Events**. In the Now Code Editor section, which is next to the Configuration pane, preview the EVAM definition script. **EVAM Data Resource configuration** 
Fetch EVAM Data	To add an EVAM definition and other information about the data source, select **Configuration**. ○ Type: GRAPHQL. ○ When to evaluate: To have the EVAM data resource instance evaluated on a page load, select **Immediately**. To use an

EVAM data source	Description
	event handler to evaluate the EVAM data resource, select **Only when invoked**. ○ EVAM definition: In the **EVAM Definition** field, add the EVAM definition record that is associated with the data resource. ○ Page cursor: In the **Page Cursor** field, add a page cursor for the EVAM definition. ○ Page size: In the **Page Size** field, add a size for the EVAM pagination. ○ Page number: In the **Page Number** field, add the page number for the EVAM pagination. ○ EVAM Filter Preference: In the **EVAM Filter Preference** field, add the sys_ids for the EVAM data filter. To add an event handler for when that data fetch is initiated, succeeded, or failed, select **Events**. Preview the EVAM definition script in the **Now Code Editor** to the right of the Configuration pane. 
Fetch EVAM Metadata	To select when to evaluate the EVAM data resource and add an EVAM definition, select **Configuration**.

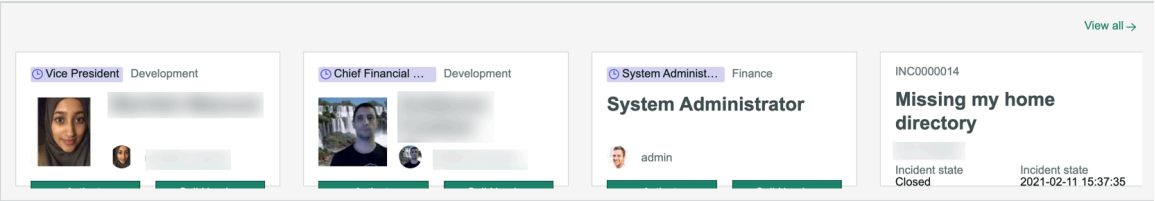
EVAM data source	Description
	○ Type: GRAPHQL. ○ When to evaluate: To have the EVAM data resource instance evaluated on a page load, select **Immediately**. To use an event handler to evaluate the EVAM data resource, select **Only when invoked**. ○ EVAM definition: In the **EVAM Definition** field, add the EVAM definition record that is associated with the data resource. To add an event handler for when that data fetch is initiated, succeeded, or failed, select **Events**. Preview the EVAM definition script in the **Now Code Editor** to the right of the Configuration pane. 
Search EVAM Data Resource	To add an EVAM definition and other information about the data source, select **Configuration**. ○ Type: Composite. ○ When to evaluate: To have the EVAM data resource instance evaluated on a page load, select **Immediately**. To use an event handler to evaluate the EVAM data resource, select **Only when invoked**. ○ EVAM Config ID: In the **EVAM Config ID** field, add the **EVAM Config ID** that is associated with the data record. ○ Search Context Config ID: In the **Search Context Config ID** field, add the Search Context Config ID that you are searching for. ○ Search Term: In the **Search Term** field, add the search terms that you are searching for ○ Facet Filters: In the **Facet filters** field, add any facet filters to help define your search.

EVAM data source	Description
	○ Search Filters: In the **Search Filters** field, add any search filters to help define your search. ○ Disable Spell Checking: In the **Disable Spell Checking** field, select if you desire spell checking to be on or off during your search. ○ Pagination Token: In the **Pagination Token** field, add a pagination token if needed. To add an event handler for when that data fetch is initiated, succeeded, or failed, select **Events**. Preview the EVAM definition script in the **Now Code Editor** to the right of the Configuration pane. 

9. Configure your data set component:
   - Bind an EVAM data resource to your component.
   - Set the **Show grid/list** toggle to on to let users choose between the grid or list view of the EVAM information on the page.
   - Set the other configuration settings for the cards on the page.
10. Select **Save**.

11. View and test your page by selecting the **Preview** button.

Finished state of the EVAM



Edit code with the Now Code Editor (advanced feature)

Now Code Editor is a rich-text editor like interface that supports Cascading Style Sheets (CSS), Hypertext Markup Language (HTML), JavaScript, Extensible Markup Language (XML), and JavaScript Object Notation (JSON). Use Now Code Editor to modify UI configuration, data resource configuration, styles, events, client-side and server-side scripts in Next Experience UI Builder components.

Now Code Editor supports the following features:

- Basic editing
- Debugging
- Command palette
- Code formatting
- Syntax checking and highlighting
- Auto-suggestions
- Script macros for common code

Basic editing

Action	Description
Format code	Applies proper indentation to the script. Keyboard shortcut: • Windows: Shift+Alt+F • Mac: Shift+Option+F
Highlight syntax	Highlights the syntax of the code.
Check syntax	Checks for formatting errors and highlights syntax errors. • Windows: Shift+Alt+C • Mac: Shit+Option+C
Show suggestions	Displays a list of valid elements at the insertion point such as:

Action	Description
	• Class names • Function names • Object names • Variable names Select and click an entry to add it to the script. Keyboard shortcut: • Windows: Control+Space • Mac: Control+Space You can also enable or turn off **Syntax highlighting** from the **Settings** menu.
Toggle comments	Comments one or more lines of code with two consecutive forward-slashes //. Keyboard shortcut: • Windows: Control+/ • Mac: Command+/
Show minimap	Displays the minimap of the code snippet. You can display or hide the minimap option, from the **Settings** menu.
Enable word wrap	Enables word wrap function in the editor area. You can toggle the **Enable word wrap** option from the **Settings** menu.
Show command palette	Displays a list of available commands for the common operations. You can execute editor commands, find and replace text, fold and unfold code blocks, toggle comments and many more tasks using the same interactive window. Keyboard shortcut • Windows: F1 • Mac: F1
Expand editor or collapse editor	Expands or collapses the editor. Keyboard shortcut • Windows: Control+M • Mac: Control+M

Debugging

To launch Script Debugger, click the Script Debugger icon in the toolbar.

Note:
You can add a breakpoint, conditional breakpoint, or logpoint, only when debugging is enabled and selected language is JavaScript.

Task	Do this
Add breakpoint	Right-click beside a line number in the ruler area and select **Add breakpoint**.
Add conditional breakpoint	1. Right-click beside a line number in the ruler area and select **Add conditional breakpoint**. 2. Enter a break condition in the editor.
Add logpoint	Right-click beside a line number in the ruler area and select **Add logpoint**.
Compare text in Diff mode	Use the side-by-side view icon and inline view icon to toggle between views.

Code editor macros

for

- Description: Inserts a standard for loop with an example array.
- Output:

```
for (var i=0; i< myArray.length; i++) {
 //myArray[i];

}
```

method

- Description: Inserts an empty JavaScript function template.
- Output:

```
/*-------------------------------------------------
-------------
   * Description:
   * Parameters:
   * Returns:

 -------------------------------------------------
-----------*/
   : function() {

   },
```

info

- Description: Inserts a *GlideSystem* information message.
- Output:

```
gs.addInfoMessage(gs.getMessage(""));
```

doc

- Description: Inserts a comment block for describing a function or parameters.
- Output:

```
/**

* Description:

* Parameters:

* Returns:
*/
```

vargror

- Description: Inserts a *GlideRecord* query for two values with an OR condition.
- Output:

```
var gr = new GlideRecord('');

var qc = gr.addQuery('field', 'value1');

qc.addOrCondition('field', 'value2');
gr.query();

while (gr.next()) {


}
```

vargr

- Description: Inserts a standard *GlideRecord* query for a single value.
- Output:

```
var gr = new GlideRecord("");
gr.addQuery("name", "value");
gr.query();
if (gr.next()) {

}
```

Testing and debugging applications

Verify the application meets your business requirements. Your testing should cover record operations (such as create, read, update, and delete), user interface elements (such as views and UI policies), runtime operations (such as business rules), and event script actions.

Testing and debugging on Core UI

Automated Test Framework

Create and run automated tests to confirm that your instance works after making a change. Review failed test results to identify the changes that caused the failure and the changes that you should review.

Debugging scripts

Debug scripts using session logs and ServiceNow AI Platform debugging tools, such as a walk-through script debugger and error messages that display in the UI.

Script Debugger and Session Log

The Script Debugger enables users with the script_debugger role to debug server-side JavaScript, while the Session Log enables you to view and download required logs.

Script Tracer and debugging scripts

Use the Script Tracer to filter your debugging search and quickly narrow down script problems. Finding specific lines of scripts, rather than doing a wide search, helps save time and improves productivity.

Related ServiceNow applications and features

Test Management 2.0

The ServiceNow® Test Management 2.0 application streamlines the management of testing processes to help you deliver software products more efficiently and with fewer errors. You can create multiple versions of a test and integrate with Agile Development 2.0.

Impersonate a user

Administrators can impersonate other authenticated users for testing purposes and view impersonation logs. When impersonating another user, the administrator has access to exactly what that user can access in the system, including the same menus and modules.

Test your apps with the ATF

The Automated Test Framework (ATF) enables you to create and run automated tests to confirm that your instance works after making a change. For example, after an upgrade, during application development, or when deploying instance configurations with update sets. Review failed test results to identify the changes that caused the failure and the changes that you should review.

Note:
By default, the system property to run automated tests is disabled to prevent you from accidentally running them on a production system. Run tests only on development, test, and other non-production instances to avoid data corruption and outage.

For developer training, see Using the Automated Test Framework on the ServiceNow Developer Site.

Benefits

Automated Test Framework provides these benefits for change managers and developers.

- Reduce upgrade and development time by replacing manual testing with automated testing.
- Design tests once and reuse them in different contexts and with different test data sets.

- Keep test instances clean by rolling back test data and changes made after each test run.
- Create test suites to organize and run tests in batches.
- Schedule test suite runs.
- Enable non-technical test designers to create tests of standard ServiceNow AI Platform functionality.
- Reduce test design time by copying quick start tests and test suites.
- Create custom test steps to expand test coverage.

Automated Test Framework records and components

The Automated Test Framework consists of these records and components.

Test

A test is a logical grouping of related automated test steps that verify some functionality or feature. Each test is a record in the Test [sys_atf_test] table. Test designers typically create a test to verify one feature or a group of related features. For example, the **CSM: Create Product Case** test validates the creation of Product Case records. Each test has a related list of test steps and test results.

Test suite

A test suite is a collection of tests that run in a specific order. Test designers typically create a test suite to test an application or a group of related features. For example, the **CSM: Case Management** test suite validates the functionality of the Customer Service Management application. Test designers can schedule running test suites and starting any required client test runners.

Quick start test

A quick start test is a test or test suite installed with the demo data of an application. Use quick start tests as templates to build your own tests and test suites. See this quick start test overview video for more information.

Test step

A test step combines a step configuration with the runtime test data necessary to run a step. The test step always specifies the order in which it runs in the test. Test steps have their own related list of step results. Each test step is a record in the Test Step [sys_atf_step] table that specifies a test action, the step configuration, and an execution order. Test designers add test steps to tests to verify functionality. For example, the first test step of the **CSM: Create Product Case** test is to impersonate the demo user John Jason who is authorized for Case Management.

Step configuration

A step configuration is a specific test action the Automated Test Framework can run. Step configurations do not contain any runtime test data and can only be run when test designers add them as part of a test step. Each step configuration is a record in the Test Step Config [sys_atf_step_config] table that specifies the input variables used to run the step configuration and the output variables available to other step configurations. For example, the **Impersonate** step configuration allows a test to impersonate another user.

The Automated Test Framework provides a default list of step configurations for most use cases and allows test designers to create their own custom step configurations.

Step variable

A step variable stores step-specific input and output values. For example, the **Open a New Form** step configuration has variables to specify the table and form view names. Use step variables to specify a particular test step target or to pass information to other test steps.

Test result

A test result stores the output of a test or test suite run. Each test result is a record in the Test Result [sys_atf_test_result] table that specifies the duration of the test run, the status of the test, and screenshots where available. Use test results to identify failing or non-running tests, and use the test logs to see more information about test results. By default, the system deletes test and test suite results 30 days after creation unless you enable the option to retain the test result indefinitely.

Step result

A step result stores the output of a test step run. Each step result is a record in the Step Result [sys_atf_test_result_step] table that specifies the status of the test step, a summary of the output, and a complete log of the output generated by the test step. Use step results to identify failures and functionality needing review.

Assert type

The **Assert type** field specifies the conditions that must be met for a test to pass. Test designers can use assertions to specify whether the results of an operation are expected or unexpected. For example, suppose you want to test that a record cannot be updated. In this case, you would add a Record Insert test step and set the **Assert type** field to **Record was not inserted**. The test passes when the record insert fails.

Some test step categories that include steps with an **Assert type** field include:

- Server category: Assert which CRUD operations cause a test to pass or fail.
- Custom UI category: Assert which component states cause a test to pass or fail, and whether visible text causes a test to pass or fail.
- Forms in Service Portal category: Assert whether a form canceled in the browser due to validation errors, or a form successfully submitted to the server causes a test to pass.

Client test runners

A client test runner is a browser tab that runs client-side test steps within a ServiceNow AI Platform user interface. Client test runners require a browser tab to function. If no client test runner is available when you run a test, the system prompts you to open one. Testers can manually start a client test runner or select an existing client test runner. Test designers can schedule starting client test runners when they schedule running a test suite.

User roles

Assign roles to define Automated Test Framework permissions.

Role Title [name]	Role Description
atf_test_admin	Create or edit Automated Test Framework properties. Has permission to:

Role Title [name]	Role Description
	• View the tests page • Create/edit/delete tests • Create/edit/delete test steps • View the step config page • View the test runner page • View the test suite results, test results and result items pages • Execute user tests • View, create, edit, delete and execute test suites • Create/edit step config records • Create/edit Automated Test Framework properties
atf_test_designer	View Automated Test Framework properties only (cannot create or edit properties). Has permission to: • View the tests page • Create/edit/delete tests • Create/edit/delete test steps • View the step config page • View the test runner page • View the test suite results, test results and result items pages • Execute user tests • View, create, edit, delete and execute test suites • View Automated Test Framework properties
atf_ws_designer	View or set basic authentication profiles needed for REST endpoints that require authentication. See REST category for more information.

ATF doesn't support these elements of Next Experience, but support for these features is planned for future releases:

- Pages built with UI Builder, including pages with lists and form components.
- Landing pages

Note:
ATF still supports the Core UI, including Classic Environment (such as classic lists and forms).

Getting started with the Automated Test Framework

If you are new to the Automated Test Framework, read this overview to learn what the framework can do. Next, follow the tutorial to create and run a test that uses the most basic of ATF features. After you feel comfortable with the basics, explore more advanced features provided by the ATF.

ATF features provide flexibility in how you test your instance.

Test step configuration categories

Category	Description
Service Catalog in Service Portal	Perform end-to-end testing for a catalog item in the Service Portal. • Open a record producer, catalog item, or order guide. • Set variable values and catalog item quantity. • Validate variable values, states, price, and items included in an order guide. • Navigate in an order guide. • Open and toggle catalog items in an order guide. • Add an item or an order guide to a shopping cart. • Order a catalog item or an order guide. • Submit a record producer.
Application Navigator	Create tests to check navigation features. • Verify that application menus are listed in the left navigation bar. • Verify that application modules are listed in the left navigation bar. • Navigate to a module as if a user clicked the module in the left navigation bar.
Custom UI	Create simple tests that mimic user actions with no scripting. • Set component values. • Assert that specified text is or is not on a page. • Validate component values. • Click components. • Validate the states of components (read-only or not read-only).
Form	Create tests of forms. • Open a new form or an existing record. • Set field values. • Validate field values or field states (such as mandatory, not mandatory, read only, not read only, visible, and not visible). • Validate whether a UI action is visible.

Category	Description
	• Click a button on a modal page. • Click a UI action. • Submit a form.
Service Catalog	Perform end-to-end testing for a catalog item. • Open a catalog item or a record producer. • Search for a catalog item. • Set variable values and catalog item quantity. • Validate variable values, states, and price. • Add an item to a shopping cart. • Order a catalog item. • Submit a record producer.
Forms in Service Portal	Create tests of forms in the Service Portal. • Open a form. • Set field values. • Validate field values or field states (such as mandatory, not mandatory, read only, not read only, visible, and not visible). • Validate whether a UI action is visible. • Click a UI action. • Submit a form.
REST	Create and send an Inbound REST request and verify the response. • Test any REST endpoint on the instance. • Use a REST request to create records, as well as retrieve, update, or delete records created in a previous test step or that already existed on the instance. • Verify the response status code, response headers, response time, and response payload.
Server	Perform more complex operations, including the following:

Category	Description
	• Perform unit tests using JavaScript, including tests using the Jasmine test framework. • Test business rules, script includes, and other scripts. • Create tests that operate on data that you define.

Output variables

Many test steps return output variables whose values you can use as inputs to a later step. For example, you can use output variables to accomplish the following tasks:

- Perform a server-side assert on a record that you previously inserted.
- Create a record as one user, and then reopen its form as a different user.

Custom test step configurations

In addition to the steps built into the Automated Test Framework, you can create custom test step configurations. These custom steps can take input variables and return output variables that you define.

Note:
You can only define custom test steps that run on the server. The Automated Test Framework does not support creating custom step configurations that run on the browser.

Data preservation

The Automated Test Framework automatically tracks and deletes any data created by running tests, and automatically rolls back changes after testing.

Test suites

Test suites enable you to execute a batch of tests in a specified order. In addition, test suites can be hierarchical, with suites nested within other suites. You can associate test suites with schedules that determine when the system runs the test suites.

Build and run your first automated test

Follow these step-by-step instructions to create and run your first automated test. This test creates a new user record.

- The Automated Test Framework (com.glide.automated_testing_framework) plugin must be activated. It is activated by default on zBoot or upgrade.
- If necessary, enable test execution. For instructions, see Enable or disable executing Automated Test Framework tests.

Note:
By default, the system property that is used to run automated tests is disabled to prevent you from accidentally running these tests on a production system. To avoid data corruption or an outage, run tests only on development, test, and other non-production instances.

- Role required: admin

Create new test

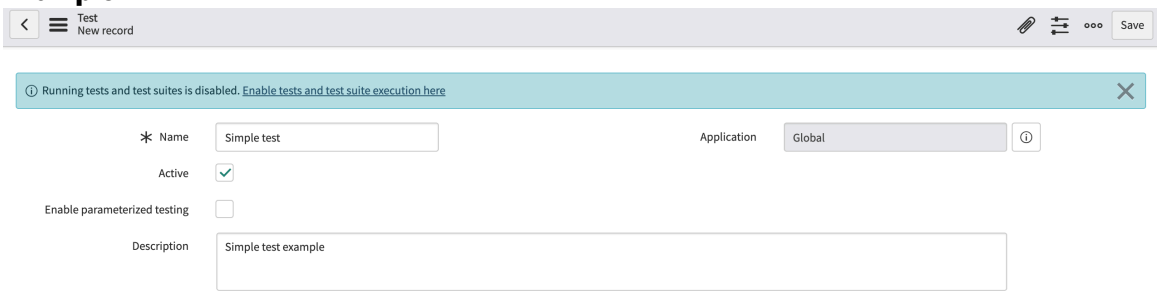
Create a new automated test record.

Before you begin
Role required: atf_test_admin or admin

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click **New**.
3. On the Test new record form, enter a name for your test in the **Name** field.
   The system identifies this test by this name whenever it displays a list of tests (for example, under the Tests module).

Example



4. In the **Description** field, enter a description for your test.
5. Click **Save**.

Result
The system creates a new test record and returns to the list of tests. For more information about creating new automated tests, see Create a new automated test.

Add the first step to the new test

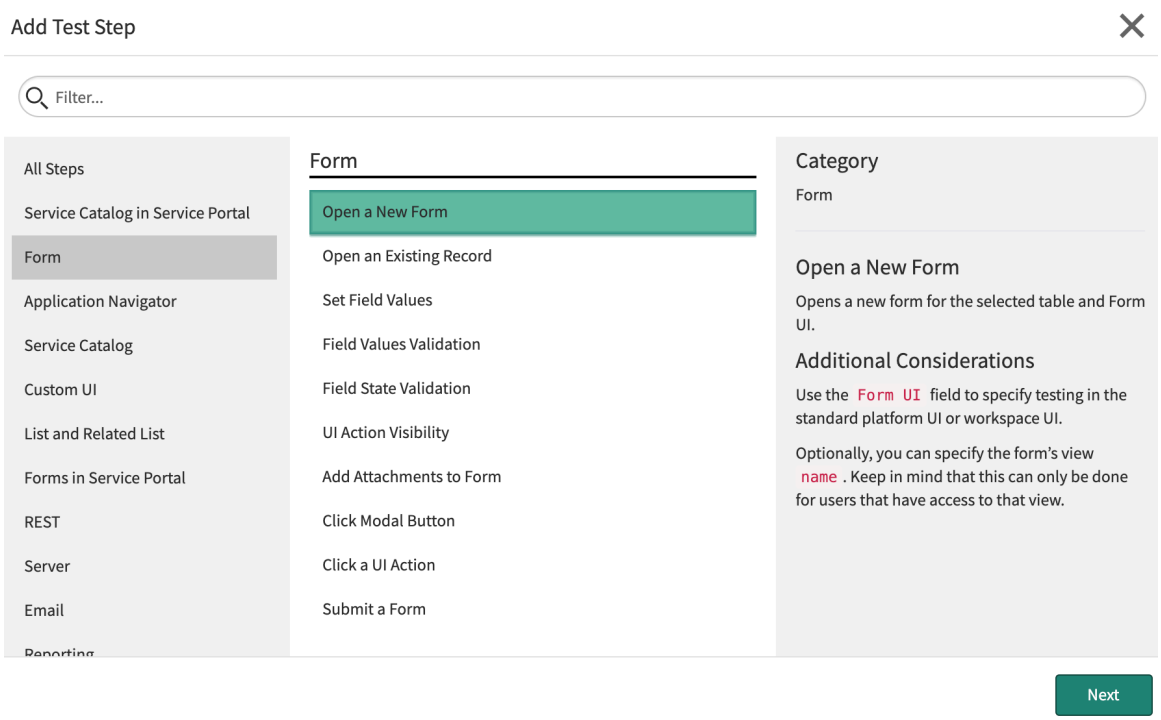
Add the first of three steps to the automated test.

Before you begin
Role required: atf_test_admin or admin

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click the test that should contain the new test steps.
3. In the Test Steps related list at the bottom of the Test form, click **Add Test Step**.
4. In the middle column, click the row for the step type **Open a new form**, then select **Next**.

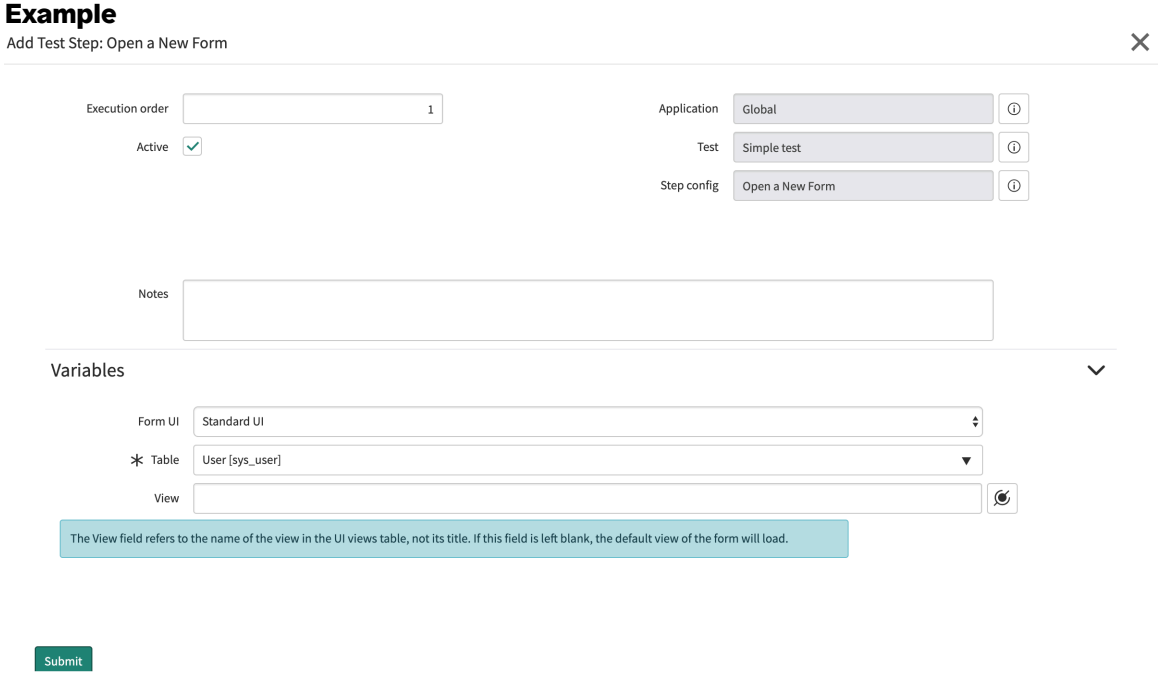
Example



The system displays the **Add Test Step** form for the Open a new form step.

5. From the **Table** field, select the **User [sys_user]** table and click **Submit**.

Example



The system creates the new step and returns to the test record.

6. Click **Update**.

Result

The system creates a new test record and returns to the list of tests. For more information about adding steps to automated tests, see Add steps to an automated test.

What to do next

For some ideas on how to continue learning about the Automated Test Framework, see Next steps with the Automated Test Framework.

Add the second step to your automated test

Add the second of three steps to the automated test.

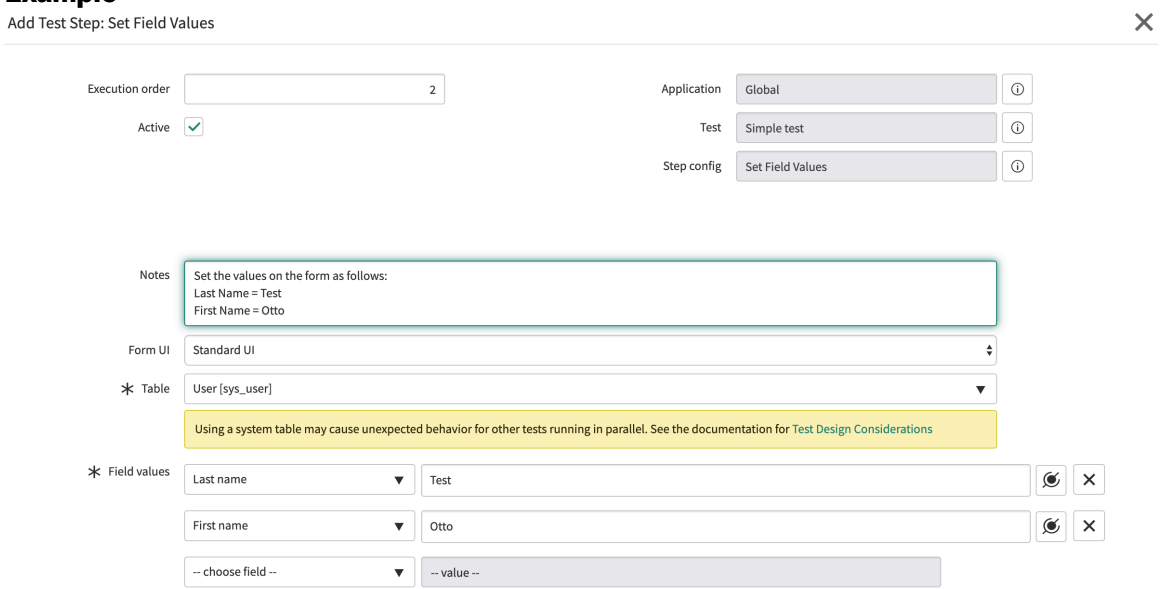
Before you begin

Role required: atf_test_admin or admin

Procedure

1. Click the test that should contain the new test steps.
2. In the Test Steps related list at the bottom of the Test form, click **Add Test Step**.
3. In the middle column, click **Set Field Values**, then click **Next**.
   The system displays the **Add Test Step** form for the Set values step.
4. In the field values section, set **Last name** to `Test` and **First name** to `Otto` (or other names of your choice).

Example



5. Click **Submit**.

Result

The system creates the new step and returns to the test record. For more information about adding steps to automated tests, see Add steps to an automated test.

Add the third step to your test

Add the last of three steps to the automated test.

Before you begin

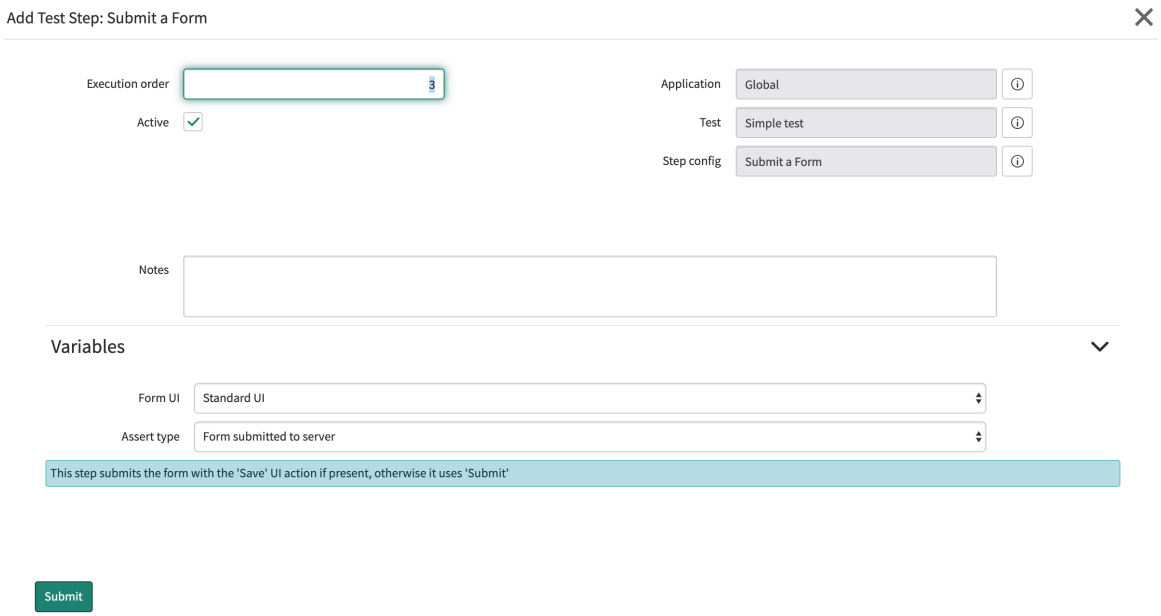
Role required: atf_test_admin or admin

Procedure

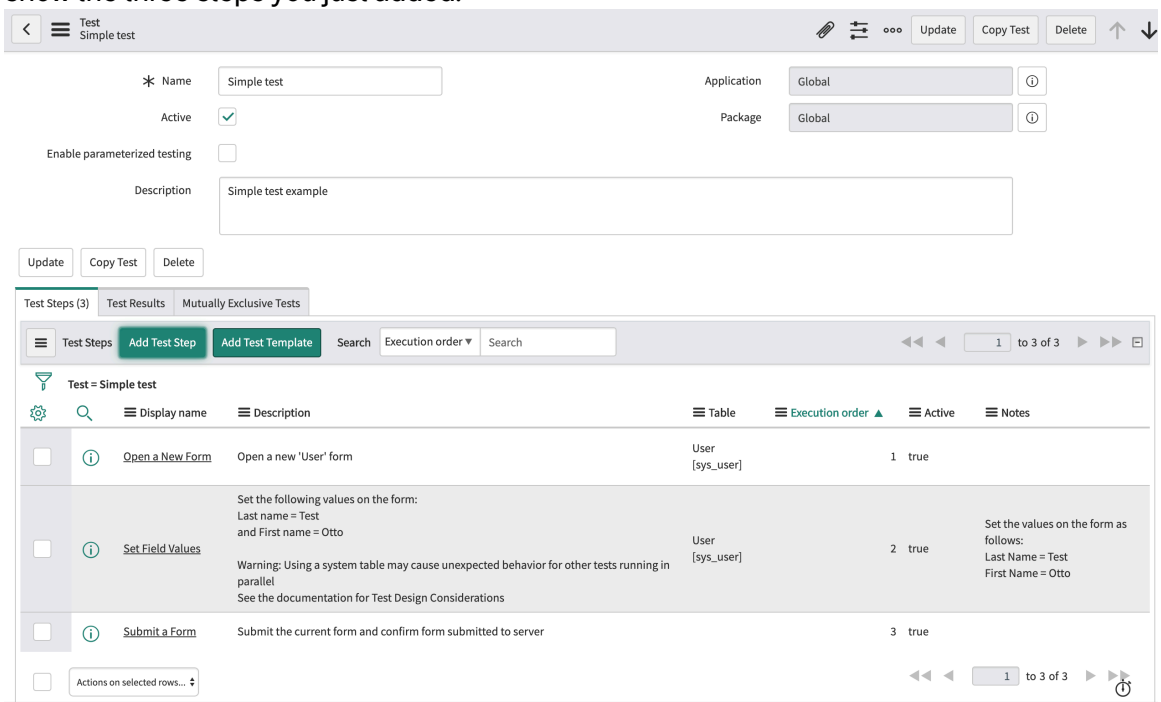
1. Click the test that should contain the new test steps.
2. In the Test Steps related list at the bottom of the Test form, click **Add Test Step**.

3. In the middle column, click **Submit a Form**, then click **Next**.
   The system displays the **Add Test Step** form for the Submit a form step.
4. Leave all values as set by default and click **Submit**.

Example



The system creates the new step and returns to the test record. The test record should now show the three steps you just added.



5. Click **Update**.

Result

The system returns to the list of tests. For more information about adding steps to automated tests, see Add steps to an automated test.

Run your test

After adding test steps, run and monitor the progress of the automated test.

Before you begin
Role required: atf_test_admin or admin

Procedure

1. If necessary to view the Tests list, click **Tests**.
2. Click the row containing the test you just created.
   The system displays the **Test** form.
3. Click **Run Test**.
   Because this test includes a form step (any step involving a UI), the system displays a dialog box asking you to choose among any currently-running test clients or start a new test runner. To continue, select **start a new test runner** and click **Run Test**.

Result
The system displays the Run Test progress dialog. For more information about running automated tests, see Run an automated test.

Monitor test progress and view test results

Monitor the progress of the automated test and view its test results.

Before you begin
Role required: atf_test_admin or admin

About this task
If needed, you can cancel a test even while it's running. For more information, see Cancelling automated tests and test suites.

Procedure

1. Monitor the progress of the test in the Run Test progress dialog.

   Note:
   If your test creates data, the system rolls back that data after all steps in the test complete.

2. When complete, click **Go to Results** on the Run Test progress dialog to display the **Test Results** list, where you can view and analyze the results.

Next steps with the Automated Test Framework

After you feel comfortable creating and running simple tests, explore the more advanced features of the Automated Test Framework.

Batch tests together with test suites
If you commonly run a set of tests together, you can group them using test suites. To learn about test suites, see Building and running automated test suites.

Pass data from one test step to another with input and output variables

You can pass data from one test step to another using input variables and output variables. For more information, see Passing data from one automated test step to another.

Reuse common sequences of steps with templates

If you frequently add the same sequence of steps to your tests, save time by creating a template. To learn more about templates, see Add a predefined list of steps (template) to an automated test

Domain separation and Automated Test Framework

Domain separation is supported in the Automated Test Framework. Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Support level: Standard*

The support level is Standard but has some exceptions or special conditions.

- Includes all aspects of **Basic** level support.
- Business logic: The service provider (SP) creates or modifies processes per customer. The use cases reflect proper use of the application by multiple SP customers in a single instance.
- The instance owner must be able to configure minimum viable product (MVP) business logic and data parameters. This configuration is done per tenant, as expected for the specific application.

Sample use case: An admin must be able to make comments required when a record closes for one tenant, but not for another.

For more information on support levels, see Application support for domain separation.

ATF use case

Automated test framework design and runtime access are solely for the owner of the instance to validate domain-specific processes. By designing tests for my customers we confirm results per tenant.

Testing

When testing domain separation during ATF test steps, you must set the domain first. This should be part of the first impersonation step of each of the ATF test steps when they are dependent on a domain being set. To learn more about domain separation recommended practices, see Domain separation recommended practices for service providers

Related topics

Domain separation for service providers

Headless Browser for Automated Test Framework

Improve your UI testing by automating the creation of browsers to process Automated Test Framework (ATF) User Interface (UI) tests. This feature is known as a headless browser, and helps you test UI functionality without having to manually open a browser to the Client Test Runner, which is what processes the UI tests in your local browser.

Important:
Headless Browser is a legacy feature of ATF; it is recommended to instead use the ATF Test Generator and Cloud Runner application. Cloud Runner offers you an easy setup and a seamless user experience. You can install Cloud Runner from the store app.

If you are an on-premise instance user, you must continue using the Headless Browser option. Cloud Runner is not currently available for on-premise instances.

Background

ServiceNow customers use Automated Test Framework (ATF) to test applications and instances. When developing in the ServiceNow platform, ATF ensures that your changes have both the desired behavior and don't break existing features.

Since the ServiceNow Orlando release, customers can automate the testing and deployment of their applications via the Continuous Integration/Continuous Delivery (CI/CD) API.

Overview

The automation provided by the ServiceNow Headless Browser for ATF skips the step of manually opening a browser during testing. There are several sequential procedures to follow in the one-time setup. Below are the instructions for both Linux and Windows setup.

See Headless Browser for ATF for more information on the Support Model.

Headless Browser setup for Linux

The ServiceNow® Headless Browser for Automated Test Framework provides automation so you can skip having to manually open a browser during testing. The Headless Browser setup is available in both Linux and Microsoft Windows. This topic covers the setup for Linux.

Important:
Headless Browser is a legacy feature of ATF; it is recommended to instead use the ATF Test Generator and Cloud Runner application. Cloud Runner offers you an easy setup and a seamless user experience. You can install Cloud Runner from the store app.

If you are an on-premise instance user, you must continue using the Headless Browser option. Cloud Runner is not currently available for on-premise instances.

There are several sequential procedures to follow in the one-time setup.

Prerequisites

Note:
MFA must be disabled in your instance to use the Headless Browser option.

Role required: admin on your ServiceNow instance and local administrator on the host machine.

- Install the Docker application
- Install Java Runtime Environment (JRE) 1.8 - required for keytool utility

  Note:
  The version must be Java 1.8, or errors result in your ServiceNow instance when trying to validate the certificate created with the keytool utility. To learn more, see Create encryption keys using the Java KeyStore keytool.

- Install OpenSSL
- Two-way communication
  - There must be two-way communication between the instance URL and your server.
  - Find the IP address of your server and get your hostname. You can use one or both of them, but you need at least one. Make sure the address or hostname are visible from your ServiceNow instance.

- See My IP Address to find the IP address ranges of your instance. You can then configure your server to allow inbound access to all of the indicated IP addresses/ranges.
- Use Port 2376 or your own default port for this procedure. Make sure your server's firewall rules allow inbound requests on this port from the IP addresses you get from the instance.

Note:
If you don't have a hostname and are connecting via the IP address, you can enter the IP address and put "localhost" in the Hostname environment variable.

Steps to set up the Headless Browser for Linux

Follow these six steps (plus verification) to set up the Headless Browser for Linux.

Generate certificates for Headless Browser setup for Linux

Generate TLS/SSL certificates to secure the Docker REST API and authenticate HTTP requests.

Before you begin
Complete the prerequisites listed in the Headless Browser setup for Linux topic.

Role required: admin on your ServiceNow instance and local administrator on the host machine.

About this task

Warning:
Get certificate authority keys from a trusted certificate authority.

By default when exposing the Docker API, requests are not authenticated, which can leave your host machine vulnerable to attack. Docker API, however, does support TLS authentication where requests are verified against public private keys provided in the HTTPS encryption. In this step you create those keys for the server and the client.

Tip:
To make remembering these easier, enter the following commands in your Linux terminal.
Note: Do not add these environment variables to your terminal profile. For security reasons, they should only exist for the duration of the current session.

- `export PASSWORD="<password to generate the certificates with>"`
- `export SERVERIP="<this server's IP address>"`
- `export HOSTNAME="<hostname of this server>"`

To learn more, see Use TLS (HTTPS) to protect the Docker daemon socket.

Procedure

1. Open a command line.
2. Generate a self-signed certificate authority key or retrieve a keypair from a trusted certificate authority.
   The following commands are an example. Note that your configuration might vary.
   - `openssl genrsa -aes256 -passout pass:$PASSWORD -out ca-key.pem 4096`
   - `openssl req -passin pass:$PASSWORD -new -x509 -days 365 -key ca-key.pem -sha256 -out ca.pem`

- `chmod 0400 ca-key.pem`
- `chmod 0444 ca.pem`

3. Generate the server keypair using the certificate authority key.
   The following commands are an example. Note that your configuration might vary.
   - `openssl genrsa -out server-key.pem 4096`
   - `openssl req -subj "/CN=$HOSTNAME" -new -key server-key.pem -out server.csr`
   - `echo "subjectAltName = DNS:$HOSTNAME,IP:$SERVERIP,IP:127.0.0.1" > extfile.cnf`
   - `openssl x509 -passin pass:$PASSWORD -req -days 365 -in server.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -out server-cert.pem -extfile extfile.cnf`
4. Create the client keypair using the certificate authority key.
   The following commands are an example. Note that your configuration might vary.
   - `openssl genrsa -out client-key.pem 4096`
   - `openssl req -subj "/CN=$HOSTNAME" -new -key client-key.pem -out client.csr`
   - `echo "extendedKeyUsage = clientAuth" > extfile.cnf`
   - `openssl x509 -passin pass:$PASSWORD -req -days 365 -in client.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -out client-cert.pem -extfile extfile.cnf`

   Now you've created all your encryption keys.
5. Import the CA Public Key and Client Keypair to a java keystore.
   The following commands are an example. Note that your configuration might vary.

   Create the keystore file and create a password for it (and save for later use): `keytool -genkey -keyalg RSA -alias dse -keystore my.keystore`

   Delete a default entry from the keystore file: `keytool -delete -alias dse -keystore my.keystore`

   Import the CA public key to the keystore: `keytool -import -keystore my.keystore -trustcacerts -alias ca -file ca.pem`

   Import the client keypair.

   Note:
   You are creating a new pkcs12 keystore file and importing the keypair to it. Then copy the contents to your original keystore file.

   - `openssl pkcs12 -export -name clientkeypair -in client-cert.pem -inkey client-key.pem -out clientkeypair.p12`
   - `keytool -importkeystore -destkeystore my.keystore -srckeystore clientkeypair.p12 -srcstoretype pkcs12 -alias clientkeypair`

   Now that you have added all of the certificates to the keystore file, save the file **my.keystore** for later, as it will be uploaded to the ServiceNow instance. In addition, be sure to remember the password that you entered when prompted to create the keystore file; you will need to enter that into a form in the ServiceNow instance.

Configure Docker for Headless Browser setup in Linux

Complete Step 2 in the Linux setup for the ServiceNow® Headless Browser for ATF: Configure Docker Server to authenticate all requests.

Before you begin

Complete Step 1: Generate certificates for Headless Browser setup for Linux

This task: After creating your client and server keys as directed in Step 1, now you configure the Docker Server to authenticate all requests using those keys, and expose the Docker API on Port 2376.

Role required: admin on your ServiceNow instance and local administrator on the host machine.

Procedure

1. Configure Docker to use the certificates that you generated in Step 1.
   a. Find or create the /etc/docker/daemon.json file. You can run: `touch /etc/docker/daemon.json`
   b. Edit the daemon.json file and add the following JSON values. Be sure to replace with the correct paths to your certificates:

```
{
"debug": true,
"tlscacert": "<path to your certificates>/ca.pem",
"tlscert": "<path to your certificates>/server-cert.pem",
"tlskey": "<path to your certificates>/server-key.pem",
"tlsverify": true
}
```

   To learn more, see Configure the Docker daemon.
2. Configure Docker to expose the remote API on a port (Port 2376 is suggested).
   You can configure Docker to accept remote connections with the docker.service systemd unit file for Linux distributions using **systemd**, such as recent versions of RedHat, CentOS, Ubuntu and SLES, or with the daemon.json file, which is suggested for Linux distributions that do not use systemd.

   If using **systemd (systemctl)**:

   a. Use the command `sudo systemctl edit docker.service` to open an override file for docker.service in a text editor.
   b. Add or modify the following lines, substituting your own port if not using 2376.

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2376
```

   c. Save the file.
   d. Reload the systemctl configuration.

```
sudo systemctl daemon-reload
```

   e. Restart Docker

```
sudo systemctl restart docker.service
```

3. To enable Docker access for the current user via a command line, copy certificates to the user's Docker home directory:
   - `mkdir -pv ~/.docker`
   - `cp ca.pem ~/.docker`
   - `cp client-key.pem ~/.docker/key.pem`
   - `cp client-cert.pem ~/.docker/cert.pem`
4. Set DOCKER_HOST and DOCKER_TLS_VERIFY environment variables for your user:
   - `echo "export DOCKER_HOST=tcp://${SERVERIP}:2376 DOCKER_TLS_VERIFY=1" >> ~/.bash_profile`
   - `source ~/.bash_profile`

   To learn more, see Manage Docker as a non-root user.

Create the Docker image and containers for Headless Browser setup in Linux

Pull the Docker image from the Public Registry.

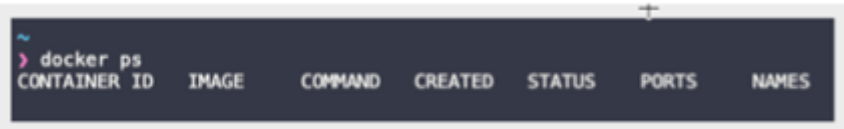
Before you begin

Complete Step 2: Configure Docker for Headless Browser setup in Linux

Role required: admin on your ServiceNow instance and local administrator on the host machine.

Procedure

1. In the command line, run `docker ps` to verify that Docker is working.
   Your results should look like this:



2. Pull the latest Docker image from the GitHub repo ATF Headless Runner.
   This command is an example, as these images are subject to change.
   `docker pull ghcr.io/servicenow/atf-headless-runner:lin-1.0.0`

> **Note:**
> The instance-to-image compatibility makes sure that the automation script inside the Docker image is compatible with the instance code. Elements such as the user interface might change over time to support new features or upgrades.

Add secrets to Docker for Headless Browser setup in Linux

Create a Docker secret, which stores the password of the ServiceNow user who will log into the instance to execute the tests. Docker Secrets is a feature of ServiceNow® for securely storing the passwords that will be used in containers.

Before you begin

Complete Step 3: Create the Docker image and containers for Headless Browser setup in Linux

About this task

Inside of the Docker container is an automation script that opens a web browser, logs into the instance, and opens the client test runner page. In order to log into the ServiceNow instance, you will need a user password. In this procedure you use a Docker feature called **Docker Secrets**

so you can securely store passwords. When you run containers, the password is automatically available to log in to your instance.

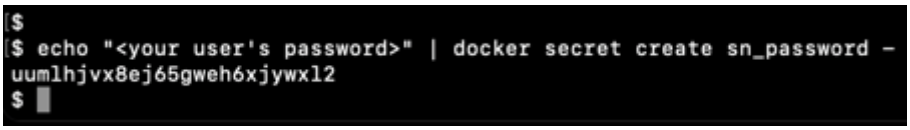
Role required: admin on your ServiceNow instance and local administrator on the host machine.

Procedure

1. At the command line, enter `docker swarm init`
2. Enter `echo "<your user's password>" | docker secret create sn_password -`

   Note:
   Replace `<your user's password>` with the user's ServiceNow password.

   Your results should look like this:



Result

The result is your secret ID, which you must save for later use. The secret ID will be added to the ServiceNow instance in the **sys_property** `sn_atf.headless.secret_id`.

Set up instance for Headless Browser in Linux

Step 5 in the Linux setup for the ServiceNow® Headless Browser for ATF: Set up your instance so it can support the Headless Browser.

Before you begin

Complete Step 5: Add secrets to Docker for Headless Browser setup in Linux

Role required: admin on your ServiceNow instance and local administrator on the host machine.

This task: Set up the instance so that it successfully communicates and authenticates with the host machine.

Procedure

1. Create user:
   a. Navigate to **User Administration** > **Users** and select **New**.
   b. Create a user - User ID, which can be whatever you want, as well as a user name and password. The password should be the same as the one you created for your Docker Secret container.
   c. Add the role **atf_test_designer** for this user. (To learn more, see the User Roles section in the Test your apps with the ATF topic.)
2. Create certificate:
   a. Navigate to **System Definition** > **Certificates** to open the **sys_ certificate** table. Create a new certificate (the name can be whatever you prefer):
      - **Type**: Java Key Store
      - **Password**: Password for the keystore that you created in Step 1 (Generate certificates for Headless Browser setup for Linux).
   b. Select the Attach (paper clip) icon to attach the keystore file you saved earlier to this record.

c. Select **Submit**.

d. Click **Validate certificate** and confirm that the success message displays.

3. Create protocol profile:

a. Navigate to **System Security** > **Protocol Profiles** to open the **sys_protocol_profile** table. Create a new protocol profile record:

Option	Description
Protocol	One word, lowercase: “docker”
Default port	2376 (or the one you chose to use)
Keystore	Docker host keystore

b. Select **Submit**.

4. Create a Docker spoke connection:

a. Navigate to **Connections & Credentials** > **Connection & Credential Aliases** to open the **sys_alias** table.

b. Select the alias with the name **Docker**.

c. Under the Connections related list, select **New**.

d. Fill in these fields:

- **Name**: Any text you prefer
- **Credential field**: Leave blank

e. Select the **URL Builder** check box.

f. Select the **Mutual authentication** check box.

g. In the **Protocol profile** field, select the protocol profile you created earlier.

h. In the **Host** field, add the IP address or Host name of your server.

i. Select **Submit**.

The Connection URL is automatically created by the system.

5. Modify properties:

> **Warning:**
> By default, the *com.glide.communications.trustmanager_trust_all* property is set to **false**. The ServiceNow AI Platform trusts only certificates that it can verify against the JVM certificate store. Self-signed and enterprise-signed certificates are not trusted.

You need to do this only when using self-signed certificates. To learn more, see Generate certificates for Headless Browser setup for Linux.

- `com.glide.communications.httpclient.verify_hostname`: false
- `com.glide.communications.trustmanager_trust_all`: true

Configure ATF for Headless Browser in Linux

Step 6 in the Linux setup for the ServiceNow® Headless Browser for ATF: Configure ATF with properties.

Before you begin

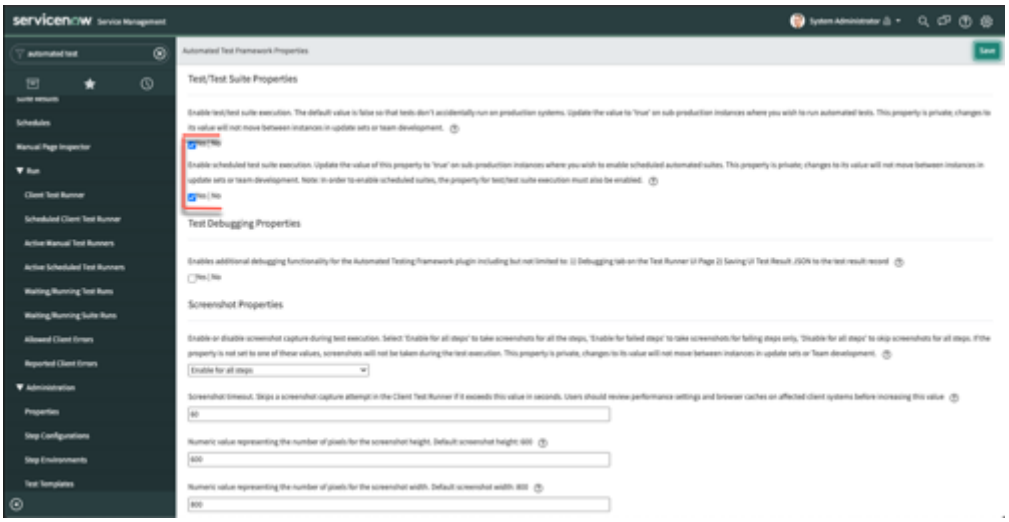
Complete Step 5: Set up instance for Headless Browser in Linux

Role required: admin on your ServiceNow instance and local administrator on the host machine.

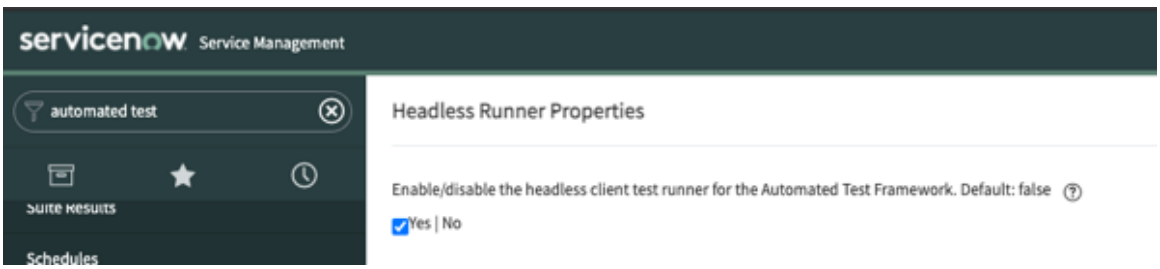
This task: Now that you've set up your connection and authentication, configure ATF with several properties so that it can start containers successfully on the host machine.

Procedure

1. In your instance, navigate to **ATF** > **Administration** > **Properties**.
2. Enable the top two properties: **Enable test/test suite execution** and **Enable scheduled test suite execution**.



3. Scroll down to the **Headless Runner Properties** section.
4. Enable the top check box.



5. Enter the following values in the Headless Runner Properties form:

Form Label	Property Name	Value to Imput
The user account used to login from the headless client test runner and begin the tests	sn_atf.headless.username	Username of the integration user that you created in Add secrets to Docker for Headless Browser setup in Linux

Form Label	Property Name	Value to Imput
The Docker secret ID that has the password stored of the user account	sn_atf.headless.secret_id	Docker "Secret ID" that you obtained earlier, as well as the Docker secret name that you created in Step 2 of the Add secrets to Docker for Headless Browser setup in Linux procedure.
The Docker image that is used for headless Client Test Runner	sn_atf.headless.docker_image_name	Image name with tag that you downloaded: `ghcr.io/servicenow/atf-headless-runner:<tagname>`

Note:
If you forgot the secret ID, go to the Docker host and run the command `docker secret list`

6. Leave the rest of the fields as they are.
7. Select **Save**.

Verify Headless Browser procedures in Linux

Step 7, the final step in the Linux setup for the ServiceNow® Headless Browser for ATF: Verify that your Headless Browser setup procedures are successful.

Before you begin
Role required: admin

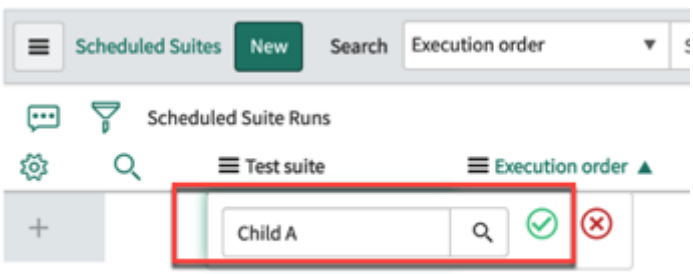
Complete Step 6: Configure ATF for Headless Browser in Linux

Role required: admin on your ServiceNow instance and local administrator on the host machine.

This task: Verify that Steps 1-6 in your Headless Browser setup procedures are successful.

Procedure

1. Go to **ATF Schedules** and next to the Suite Schedules link, select **New**.
2. In the Run box, select **On Demand**.
3. Select **Save**.
4. In the Scheduled Suite related list, add a new record for your test suite by double-clicking the **+** sign and choose your browser name of **Chrome** or **Firefox**.
   Use **Child A** as a good example to run.



5. Set the OS name to Linux.
6. Select **Execute Now**.
7. Verify that your test passed.
   When your verification is successful, any suites with UI tests run by scheduled suites or via CICD now automatically create headless test runners without the need to manually open the "scheduled client test runner" page.

Headless Browser setup for Microsoft Windows

The ServiceNow® Headless Browser for Automated Test Framework (ATF) provides automation so you can skip having to manually open a browser during testing. The Headless Browser setup is available in both Linux and Microsoft Windows. This topic covers the setup for Windows.

Important:
Headless Browser is a legacy feature of ATF; it is recommended to instead use the ATF Test Generator and Cloud Runner application. Cloud Runner offers you an easy setup and a seamless user experience. You can install Cloud Runner from the store app.

If you are an on-premise instance user, you must continue using the Headless Browser option. Cloud Runner is not currently available for on-premise instances.

There are several sequential procedures to follow in the one-time setup. Below are the instructions for the Microsoft Windows setup.

Prerequisites

Note:
MFA must be disabled in your instance to use the Headless Browser option.

Role required: admin on your ServiceNow instance and local administrator on the host machine.

Warning:
The only supported version of Microsoft Windows as a host is Windows Server 2019 v10.0.17763.737. No other versions are supported. If you are unable to meet these requirements, a Linux host is recommended.

- Make sure that the following programs are installed on your Windows server:
  - Docker: Install Docker for Headless Browser setup for Microsoft Windows
  - Java keytool: Chocolatey tool for javaruntime
  - OpenSSL: Chocolatey tool for openssl
- Two-way communication
  - There must be two-way communication between the instance URL and your server.
  - Find the IP address of your server and get your hostname. You can use one or both of them, but you need at least one. Make sure the address or hostname are visible from your ServiceNow instance.
  - See My IP Address to find the IP address ranges of your instance. You can then configure your server to allow inbound access to all of the indicated IP addresses/ranges.
  - Use Port 2376 or your own default port for this procedure. Make sure your server's firewall rules allow inbound requests on this port from the IP addresses you get from the instance.

Note:
If you don't have a hostname and are connecting via the IP address, you can enter the IP address and put "localhost" in the Hostname environment variable.

- To learn more, see Use TLS (HTTPS) to protect the Docker daemon socket.

Steps to set up the Headless Browser for Windows

Follow these seven steps (plus verification) to set up the Headless Browser for Windows.

Install Docker for Headless Browser setup for Microsoft Windows

Step 1 in the Windows setup for the ServiceNow® Headless Browser for Automated Test Framework: Install Docker.

Before you begin
Role required: admin

- See Prerequisites in the Headless Browser setup for Microsoft Windows topic.

- **Tip:**
  To make remembering these easier, set the following environment variables:
  - set PASSWORD=<password>
  - set SERVERIP=<server ip>
  - set HOSTNAME=<hostname>

Warning:
The only supported version of Microsoft Windows as a host is Windows Server 2019 v10.0.17763.737. No other versions are supported. If you are unable to meet these requirements, a Linux host is recommended.

Procedure

1. Open an elevated PowerShell session and install the Docker-Microsoft PackageManagement Provider from the PowerShell Gallery.
   PowerShell

   ```
   Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
   ```

   If you're prompted to install the NuGet provider, type `Y` to install it as well.

2. Use the PackageManagement PowerShell module to install the latest version of Docker.
   PowerShell

   ```
   Install-Package -Name docker -ProviderName DockerMsftProvider
   ```

   When PowerShell asks you whether to trust the package source 'DockerDefault', type `A` to continue the installation.

3. After the installation completes, restart the computer.
   PowerShell

   ```
   Restart-Computer -Force
   ```

Generate certificates for Headless Browser setup for Microsoft Windows

Generate TLS/SSL certificates to secure the Docker REST API and authenticate HTTP requests.

Before you begin
Role required: admin on your ServiceNow instance and local administrator on the host machine.

> **Warning:**
> The only version of Microsoft Windows that ServiceNow supports as a host is Windows Server 2019 v10.0.17763.737. No other versions are supported. If you are unable to meet these requirements, a Linux host is recommended.

- Complete the prerequisites listed in the Headless Browser setup for Microsoft Windows topic. Make sure that the following programs are installed on your Windows server:
  - Docker: Docker application for Windows
  - Java keytool: Chocolatey tool for javaruntime
  - OpenSSL: Chocolatey tool for openssl

Note the following requirements:

- Two-way communication

  > **Warning:**
  > Be sure to get certificate authority keys from a trusted certificate authority.

  - There must be two-way communication between the instance URL and your server.
  - Find the IP address of your server and get your hostname. You can use one or both of them, but you need at least one.

    > **Note:**
    > If you don't have a hostname and are connecting via the IP address, you can enter the IP address and put "localhost" in the Hostname environment variable.

  > **Tip:**
  > To make remembering these easier, set the following environment variables:
  >
  > - `export PASSWORD="<password to generate the certificates with>"`
  > - `export SERVERIP="<this server's IP address>"`
  > - `export HOSTNAME="<hostname of this server>"`

- Port: Use Port 2376 or your own default port for this procedure.

  > **Note:**
  > Make sure your firewall rules allow inbound requests on this port

- To learn more, see Use TLS (HTTPS) to protect the Docker daemon socket.

About this task
By default when exposing the Docker API, requests are not authenticated, which can leave your host machine vulnerable to attack. Docker API, however, does support TLS authentication where requests are verified against public private keys provided in the HTTPS encryption. In this step you create those keys for the server and the client.

Procedure

1. Open a command line.
2. Generate a self-signed certificate authority key or retrieve a keypair from a trusted certificate authority.
   The following commands are an example. Note that your configuration might vary.

```
openssl genrsa -aes256 -passout pass:%PASSWORD% -out ca-key.pem
 4096
openssl req -passin pass:%PASSWORD% -new -x509 -days 365 -key
 ca-key.pem -sha256 -out ca.pem
```

3. Generate the server keypair using the certificate authority key.
   The following commands are an example. Note that your configuration might vary.

```
openssl genrsa -out server-key.pem 4096
openssl req -subj /CN=%HOSTNAME% -new -key server-key.pem -out
 server.csr
echo extendedKeyUsage = clientAuth > extfile.cnf
openssl x509 -passin pass:%PASSWORD% -req -days 365 -in
 server.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -out
 server-cert.pem -extfile extfile.cnf
```

4. Create the client keypair using the certificate authority key.
   The following commands are an example. Note that your configuration might vary.

```
openssl genrsa -out client-key.pem 4096
openssl req -subj /CN=%HOSTNAME% -new -key client-key.pem -out
 client.csr
echo subjectAltName = DNS:%HOSTNAME%,IP:%SERVERIP%,IP:127.0.0.1
 > extfile.cnf
openssl x509 -passin pass:%PASSWORD% -req -days 365 -in
 client.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -out
 client-cert.pem -extfile extfile.cnf
```

5. Import the CA Public Key and Client Keypair to a Java keystore.
   This instructions are an example Your configuration might vary.

   Create the keystore file and create a password for it (and save for later use): `keytool -genkey -keyalg RSA -alias dse -keystore my.keystore`

   Delete a default entry from the keystore file: `keytool -delete -alias dse -keystore my.keystore`

   Note:
   This entry is auto-generated, s.o is not needed.

   Import the CA public key to the keystore: `keytool -import -keystore my.keystore -trustcacerts -alias ca -file ca.pem`

   Import the client keypair.

   Note:
   You are creating a new pkcs12 keystore file and importing the keypair to it. Then copy the contents to your original keystore file.

- `openssl pkcs12 -export -name clientkeypair -in client-cert.pem -inkey client-key.pem -out clientkeypair.p12`
- `keytool -importkeystore -destkeystore my.keystore -srckeystore clientkeypair.p12 -srcstoretype pkcs12 -alias clientkeypair`

Now that you have added all of the certificates to the keystore file, save the file **my.keystore** for later, as it will be uploaded to the ServiceNow instance. In addition, be sure to remember the password that you entered when prompted to create the keystore file; you will need to enter that into a form in the ServiceNow instance.

Configure Docker for Headless Browser setup in Microsoft Windows

Configure Docker Server to authenticate all requests.

Before you begin

Complete Step 2: Generate certificates for Headless Browser setup for Microsoft Windows

Role required: admin on your ServiceNow instance and local administrator on the host machine.

About this task

After creating your client and server keys, now you configure the Docker Server to authenticate all requests using those keys, and expose the Docker API on Port 2376.

Procedure

1. Configure Docker to use the certificates you generated in Step 2.
2. Find or create the `C:\ProgramData\docker\config\daemon.json` file.
3. Add the following properties to the daemon.json file.
   Be sure to replace items in these commands with the correct paths to your certificates:

   Note:
   The double slashes are important to copy exactly.

   ```
   {
   "tlscacert": "C:\\Users\\Administrator\\certs\\ca.pem",
   "tlscert": "C:\\Users\\Administrator\\certs\\server-cert.pem",
   "tlscert": "C:\\Users\\Administrator\\certs\\server-cert.pem",
   "tlskey": "C:\\Users\\Administrator\\certs\\server-key.pem",
   "tlsverify": true,
   "hosts": ["tcp://0.0.0.0:2376", "npipe://"]
   }
   ```

   To learn more, see Configure the Docker daemon.
4. In administrator PowerShell, run `restart-service *docker*`

Create the Docker image and containers for Headless Browser setup in Microsoft Windows

Pull the Docker image from the Public Registry.

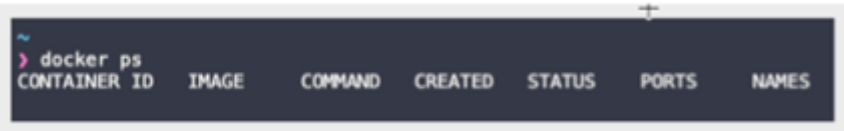
Before you begin

Complete Step 3: Configure Docker for Headless Browser setup in Microsoft Windows

Role required: admin on your ServiceNow instance and local administrator on the host machine.

Procedure

1. In an administrator command line, run `docker ps` to verify that Docker is working.
   Your results should look like this:



2. Pull the latest Docker image from the GitHub repo ATF Headless Runner.
   This command is an example, as these images are subject to change:
   `docker pull ghcr.io/servicenow/atf-headless-runner:win-1.0.0`

> **Note:**
> The instance-to-image compatibility makes sure that the automation script inside the Docker image is compatible with the instance code. Elements such as the user interface might change over time to support new features or upgrades.

Add secrets to Docker for Headless Browser setup in Microsoft Windows

Create a Docker secret that stores the password of the ServiceNow user who will log into the instance to execute the tests. Docker Secrets is a feature of ServiceNow® for securely storing the passwords that will be used in containers.

Before you begin

Complete Step 4: Create the Docker image and containers for Headless Browser setup in Microsoft Windows

Role required: Go to your ServiceNow instance and create a new user to be used by the Docker container to log in. Give this user the roles of **admin** or **atf_test_admin**. Save the password to be used in Step 2 of the following procedure.

About this task

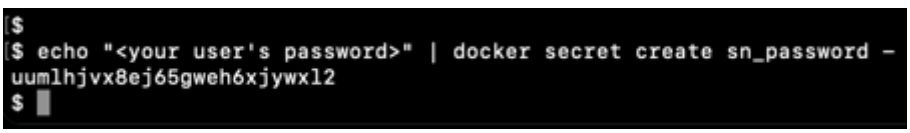
Inside the Docker container is an automation script that opens a web browser, logs into the instance, and opens the Client Test Runner page. In order to log into the ServiceNow instance, you will need a user password. In this step you use a Docker feature called Docker Secrets so you can securely store passwords. Then when you run containers, the password is automatically available to log in to your instance.

Procedure

1. In the admin command line, enter `docker swarm init`
2. In the admin command line, enter `echo <your user's password> | docker secret create sn_password -`

> **Note:**
> Replace `<your user's password>` with the user's ServiceNow password.

Your results should look like this:



Result

The result is your secret ID, which you must save for later use. The secret ID will be added to the ServiceNow instance in the sys_property `sn_atf.headless.secret_id`.

Set up instance for Headless Browser in Microsoft Windows

Step 6 in the Microsoft Windows setup for the ServiceNow® Headless Browser for ATF: Set up your instance so it can support the Headless Browser.

Before you begin

Complete Step 5: Add secrets to Docker for Headless Browser setup in Microsoft Windows.

Role required: admin on your ServiceNow instance and local administrator on the host machine.

This task: Now that you have configured the host machine, you will set up the instance so that it successfully communicates and authenticates with the host machine.

Procedure

1. Create user:
   a. Navigate to **User Administration** > **Users** and select **New**.
   b. Create a user - User ID, which can be whatever you want, as well as a user name and password. The password should be the same as the one you created for your Docker Secret container.
   c. Add the role **atf_test_designer** for this user. (To learn more, see the User Roles section in the Test your apps with the ATF topic.
2. Create certificate:
   a. Navigate to **System Definition** > **Certificates** to open the **sys_ certificate** table. Create a new certificate (the name can be whatever you prefer):
      - **Type**: Java Key Store
      - **Password**: Password for the keystore that you created in Step 2 (Generate certificates for Headless Browser setup for Microsoft Windows).
   b. Select the Attach (paper clip) icon to attach the keystore file you saved earlier to this record.
   c. Select **Submit**.
   d. Click **Validate certificate** and confirm that the success message displays.
3. Create protocol profile:
   a. Navigate to **System Security** > **Protocol Profiles** to open the **sys_protocol_profile** table. Create a new protocol profile record:

Option	Description
Protocol	One word, lowercase: “docker”
Default port	2376 (or the one you chose to use)
Keystore	Docker host keystore

   b. Select **Submit**.
4. Create a Docker spoke connection:

a. Navigate to **Connections & Credentials** > **Connection & Credential Aliases** to open the **sys_alias** table.

b. Select the alias with the name **Docker**.

c. Under the Connections related list, select **New**.

d. Fill in these fields:

- **Name**: Any text you prefer
- **Credential field**: Leave blank

e. Select the **URL Builder** check box.

f. Select the **Mutual authentication** check box.

g. In the **Protocol profile** field, select the protocol profile you created earlier.

h. In the Host field, add the IP address or Host name of your server.

i. Select **Submit**.

The Connection URL is automatically created by the system.

5. Modify properties:
   - `com.glide.communications.httpclient.verify_hostname`: false
   - `com.glide.communications.trustmanager_trust_all`: true

> **Warning:**
> By default, the *`com.glide.communications.trustmanager_trust_all`* property is set to **false**. The ServiceNow AI Platform trusts only certificates that it can verify against the JVM certificate store. Self-signed and enterprise-signed certificates are not trusted.

You need to do this only when using self-signed certificates. To learn more, see Generate certificates for Headless Browser setup for Microsoft Windows.

Configure Automated Test Framework (ATF) for Headless Browser in Microsoft Windows

Step 7 in the Microsoft Windows setup for the ServiceNow® Headless Browser for ATF: Configure ATF with properties.

Before you begin

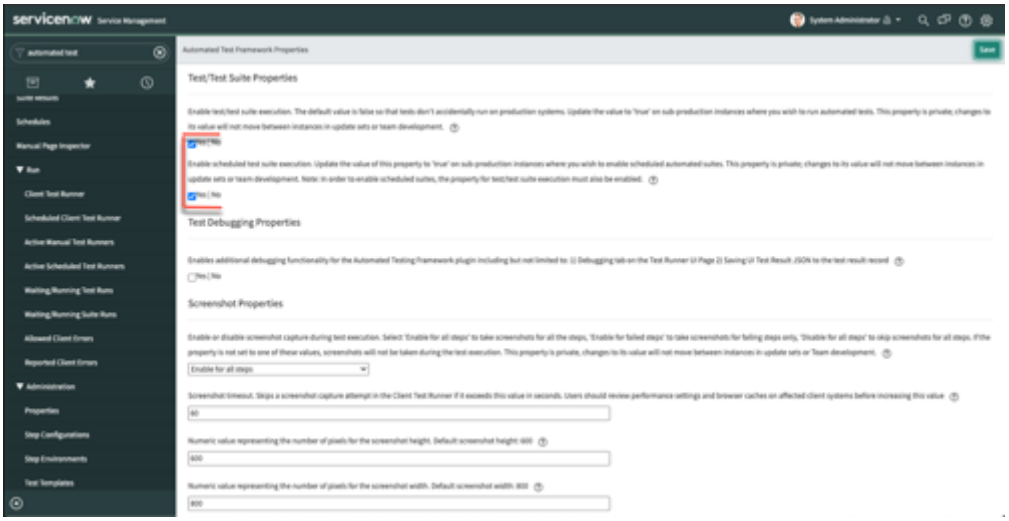
Complete Step 6: Set up instance for Headless Browser in Microsoft Windows

Role required: admin on your ServiceNow instance and local administrator on the host machine.

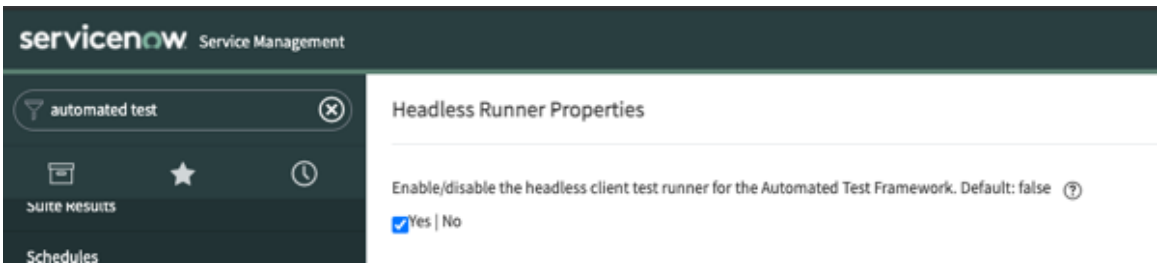
Now that your connection and authentication are set up, you will configure ATF with several properties so that it can start containers successfully on the host machine.

Procedure

1. In your instance, navigate to **ATF** > **Administration** > **Properties**.
2. Enable the top two properties: **Enable test/test suite execution** and **Enable scheduled test suite execution**.



3. Scroll down to the **Headless Runner Properties** section.
4. Enable the top check box.



5. Enter the following values in the Headless Runner Properties form:

Form Label	Property Name	Value to Imput
The user account used to login from the headless Client Test Runner and begin the tests	sn_atf.headless.username	Username of the integration user that you created in Add secrets to Docker for Headless Browser setup in Microsoft Windows
The Docker secret ID that has the password stored of the user account	sn_atf.headless.secret_id	Docker “secret ID” that you obtained earlier, as well as the Docker secret name that you created in Step 2 of the Add secrets to Docker for Headless Browser setup in Microsoft Windows procedure.
The Docker image that is used for headless Client Test Runner	sn_atf.headless.docker_image_name	Image name with tag that you downloaded: `ghcr.io/servicenow/atf-headless-runner:<tagname>`

Note:
If you forgot the secret ID, go to Windows host and run the command docker secret list.

6. At the **The absolute path of the secret file on a docker container** field, enter: `C:\ProgramData\Docker\secrets\<YOUR SECRET NAME>`
7. Leave the rest of the fields as they are.
8. Select **Save**.

Verify Headless Browser procedures for ATF in Microsoft Windows

Verify that your Headless Browser setup procedures have been successful.

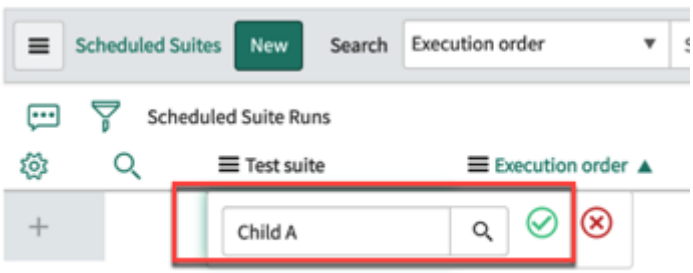
Before you begin

Complete Step 7: Configure Automated Test Framework (ATF) for Headless Browser in Microsoft Windows

Role required: admin

Procedure

1. Go to **ATF Schedules** and next to the Suite Schedules link, select **New**.
2. In the Run box select **On Demand**.
3. Select **Save**.
4. In the Scheduled Suite related list, add a new record for your test suite by double-clicking the **+** sign and choose your browser name of **Chrome** or **Firefox**.
   Use **Child A** as a good example to run.



5. Set the OS name to Windows.
6. Select **Execute Now**
7. Verify that your test passed.
   When your verification is successful, any suites with UI tests run by scheduled suites or via CICD now automatically create headless test runners without the need to manually open the "scheduled client test runner" page.

Headless Browser system properties

Below is a table of the properties you must have as you set up the ServiceNow® Headless Browser for Automated Test Framework.

System properties

Property name	Type	Default value	Purpose
sn_atf.headless.browser_options	string	"--no-sandbox,--disable-gpu"	The CLI options are passed to the browser on startup
sn_atf.headless.default_browser	string	Chrome	Default browser used when starting up headless requests
sn_atf.headless.default_os	string	Linux	Operating system of host machine
sn_atf.headless.docker_image	image_name		Name:tag of the Docker image on the host machine
sn_atf.headless.docker_init_window_seconds	integer	60	If the Docker container fails to start up twice in this amount of time, the service will not attempt to restart
sn_atf.headless.enabled	true/false	false	Determines whether the instance attempts to create headless client test runners for scheduled UI test runs
sn_atf.headless.heartbeat_enabled	true/false	true	When property is true, Docker container sends a REST API request every minute to the instance to check that the sys_atf_agent is still online, and stops the container if the record status is "offline" or the record no longer exists.
sn_atf.headless.heartbeat_uri	string	/api/now/atf_agent/online	The URL of the heartbeat endpoint so container can verify browser is still responsive
sn_atf.headless.images_check_enabled	true/false	false	When this property is true, the instance verifies that the requested Docker image:tag is present on the host before test execution

System properties (continued)

Property name	Type	Default value	Purpose
sn_atf.headless.login_button_id	string	sysverb_login	The HTML ID of the login page Submit/Login button
sn_atf.headless.login_page	string	login.do	URL of login page that the browser navigates to
sn_atf.headless.password_field_id	string	user_password	The HTML ID of the input field for the password on the Login page
sn_atf.headless.request_timeout_sec	integer	200	Number of seconds that HTTP requests being sent to the Docker host have until timeout
sn_atf.headless.retry_count	integer	10	Number of times the instance checks for agent coming online before cancelling the test run
sn_atf.headless.runner_banner_id	string	test_runner_banner	The ID of the element verifying that client test runner page loaded correctly
sn_atf.headless.runner_url	string	atf_test_runner.do?sysparm_nostack=true&sysparm_scheduled_tests_only=true&s	URL of the scheduled client test runner page and its query parameters
sn_atf.headless.secret_gid	string	1000	The GID of the Docker container default user
sn_atf.headless.secret_id	string		The ID of the Docker secret on host machine
sn_atf.headless.secret_name	string		Name of the Docker secret on host machine
sn_atf.headless.secret_path	string	/run/secrets/<secret_name>	Path where Docker secret file exists: (Learn more: https://docs.docker.com/engine/swarm/secrets/#how-docker-manages-secrets)
sn_atf.headless.secret_uid	string	1000	The UID of the Docker container default user

System properties (continued)

Property name	Type	Default value	Purpose
sn_atf.headless.service_cleanup_exclude_list	string		Service ID exceptions that should NOT be deleted during the instance service cleanup job. This job runs every night and deletes any services that are on the host and past their expiration time.
sn_atf.headless.service_stop_deletes	true/false	false	On completing of a test execution, if this property is true the instance will NOT send the service delete requests thus keeping the service and containers around on the host. Useful for debugging.
sn_atf.headless.timeout_mins	integer	1440	Number of minutes before Docker service automatically shuts down
sn_atf.headless.user_field_id	string	user_name	The HTML ID of the input field for the username on the Login page
sn_atf.headless.username	string		Username of user who logs in to the instance
sn_atf.headless.validation_id	string	headless_vp_validation	The ID of the element confirming verification page has loaded
sn_atf.headless.validation_page	string	atf_headless_validation_page	The URL of the validation page on the instance
sn_atf.headless.vp_has_role_id	string	headless_vp_has_role	The ID of the element verifying that the user has correct roles
sn_atf.headless.vp_success_id	string	headless_vp_success	The ID of the element signifying entire page loaded correctly

Related topics

- Headless Browser for Automated Test Framework
- Test your apps with the ATF

Headless Browser troubleshooting

These tips can help you troubleshoot your Linux or Microsoft Windows setup of the ServiceNow® Headless Browser for Automated Test Framework.

There are three basic areas to examine when troubleshooting your Headless Browser setup.

Docker container

Error logs: When the headless test completes, if there are errors they are generated in the Docker container. Whether the operation fails or succeeds, the container's **stdout/stderr** logs are placed in the **sn_atf_docker_service** table.

Headless client test runner did not start in the time allotted message: This message generally means an error occurred in the Docker container while it was initializing or starting up. This may indicate something is incorrect in your Docker container setup. Navigate to the **sn_atf_docker_service** table to read the logs and see the error message.

Instance

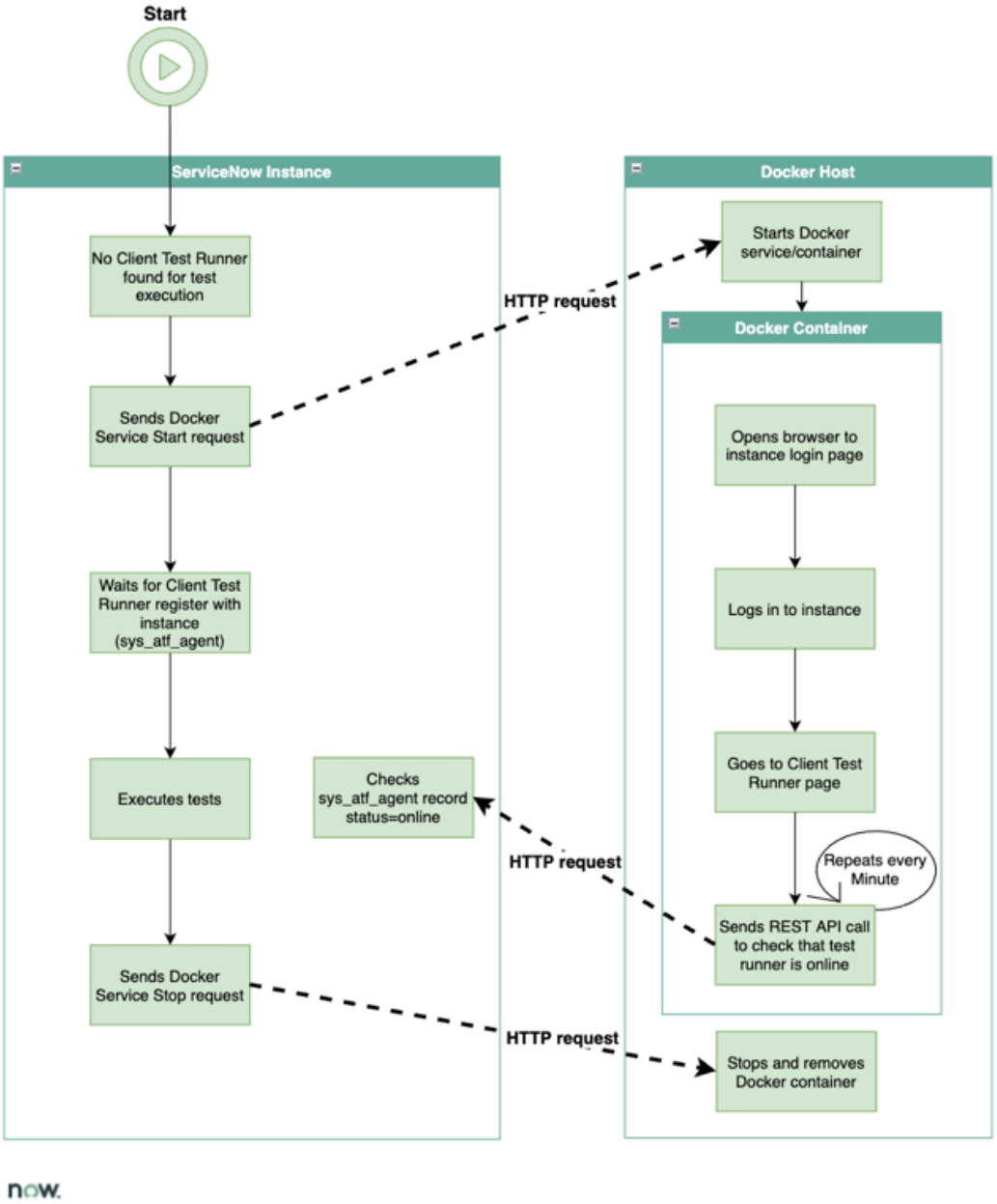
Error caught running Docker flow when starting the ATF tests messages: Follow the URL address to find the flow context, which contains the error logs. For example, the above error message can display when the instance can't access the Docker host.

Network errors

Firewalls: Make sure your firewalls are set up so that the instance can access the host and port and the server can access the instance.

Headless Browser action flow diagram

Example flow diagram



Building and running automated tests with the Automated Test Framework

Basic tasks in the Automated Test Framework.

To build and run any test, you always perform certain operations:

- Create a new automated test
- Add steps to an automated test
- Run an automated test
- View the progress of automated tests
- View test results and automated test results

You perform other operations often, but not necessarily for every test.

- For tests involving form steps, you might need to View results screenshots from an automated test.
- For some tests, you might need to know about Passing data from one automated test step to another.
- Some steps frequently occur in the same sequence in many different tests, so you can Add a predefined list of steps (template) to an automated test.

Create a new automated test

Create a named automated test containing a series of steps to execute.

Before you begin

Role required: atf_test_admin or atf_test_designer

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click **New**.
3. On the Test new record form, enter a **Name** for your test.
   The system identifies this test by this name wherever it displays a list of tests (for example, in the Tests module).
4. **Optional:** Enable parameterized testing to run a test multiple times with different test data for each run.
   For more information, see Parameterized tests.
5. Enter a **Description** for your test.
6. Click **Save**.
   The system creates a new test record and returns to the list of tests.

What to do next

Add steps to an automated test.

Add a predefined list of steps (template) to an automated test

With test templates you can add a predefined list of steps to a test. Any list of steps that follows a set pattern makes a good candidate for a template.

Before you begin

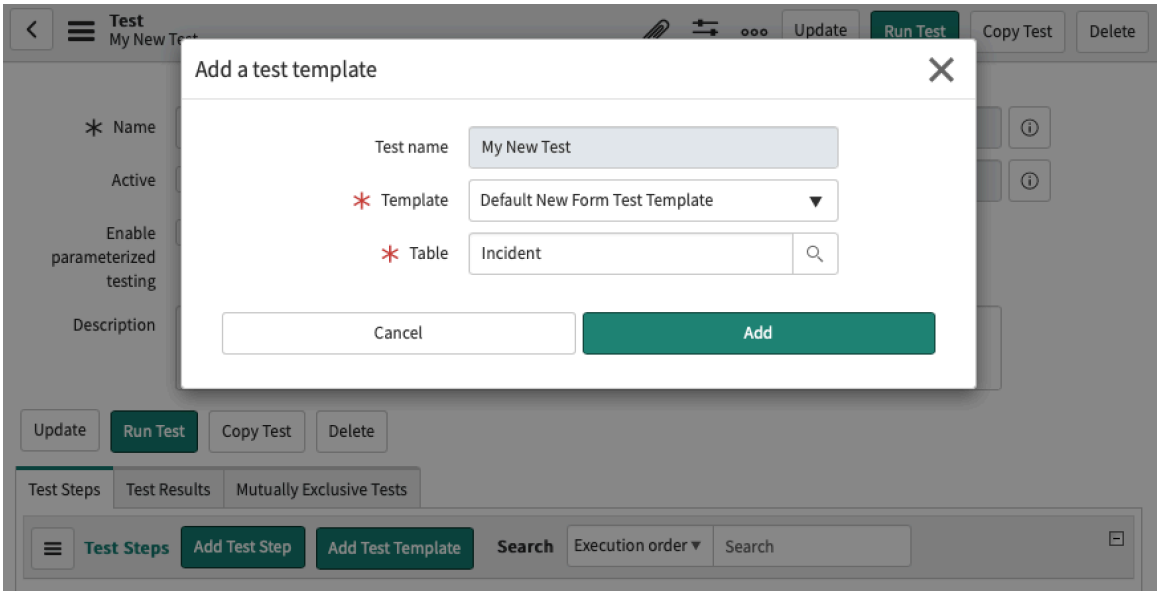
You must have created the test to which you want to add steps.
Role required: atf_test_admin or atf_test_designer

About this task

Many tests follow similar patterns. One common pattern, for example, is to open a form, set some field values, validate some field values, click a UI action, and submit the current form. If a template exists containing these steps, you can add them to a test all at once. The Automated Test Framework comes with default templates. You can also create custom test templates.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click the row for the test to which you want to add steps.
   The system displays the Test form.
3. On the **Test Steps** related list, click **Add Test Template**.
   The system displays the **Add a test template** dialog.



4. From the **Table** field, select the table you want to test with these steps.
5. From the **Template** field, select the template containing the steps you want to add.
6. Click **Add**.
   The system adds the template steps to the test. It also adds to the test description a set of instructions on how to complete the test from this template.
7. Following the instructions in the test description, edit each step added by the template to include the necessary information.

What to do next

Proceed to edit or save the test as you normally would.

Related topics

Create an automated test steps template

Add steps to an automated test

Create a series of steps for an automated test to run in a specified order.

Before you begin

You must create a test before you can add steps to the test.
Role required: atf_test_admin or atf_test_designer

About this task

This procedure lets you add one step at a time. You can also add a batch of steps at once. For more information, see Add a predefined list of steps (template) to an automated test.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click the test that should contain the new test steps.
3. In the Test Steps related list at the bottom of the Test form, click **Add Test Step**.
4. From the left panel of the Add Test Step form, select the test step category, or select **All Steps** to view all available test steps for all test categories.

For example, if selecting a specific test step category, select **Form** for form-related test steps, **Application Navigator** to test application menu or module visibility, or **Server** for tests executed on the server.

Note:
When selecting certain types of server tests, you can specify whether a positive or negative test result constitutes a successful test outcome. For more information, see Assert type in Test your apps with the ATF.

5. Click the type of test step you want to select.
   For example, if you select the **Form** category, select **Set Field Values** to set the field values on a form.

   Start any sequence of steps that work with forms with the **Open a new form** or **Open an existing form** step. Close with the **Submit form** step.
6. If applicable, from the **Insert after** drop-down list, select the step that you want to precede this step.
   If this is the first step in a test, the **Insert after** drop-down list does not appear.
7. Click **Next**.
8. From the **Table** list on the Add Test Step form, select the table that you want to test in this step.
9. **Optional:** In the **Execution Order** field, enter an integer representing the order in which you want the test to execute this step.
   For more information on **Execution Order**, see Edit automated test step order.
10. Fill in the fields that apply to this step.

    For instructions, see Test step categories.

    Some steps return output values that you can pass to the inputs for a subsequent step. For more information, see Pass values from one automated test step to another.
11. Click **Submit**.
    The system creates the step and displays the test record.
12. Repeat Steps 3 through 11 to add additional steps for this test.

What to do next

Run an automated test.

Related topics

Automated Test Framework use case examples

Change automated test step

If necessary, edit a test step after you create it.

Before you begin

Role required: atf_test_admin or atf_test_designer

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click the row containing the test you want to edit.
3. On the **Test Steps** related list, click the row containing the step you want to edit.
   The system displays the **Set Field Values** form.

4. Edit the fields you want to change.
5. Click **Update**.

Edit automated test step order

By default, steps execute in the order in which you created them. You can change this order by editing the **Execution Order** field.

Before you begin
Role required: atf_test_admin or atf_test_designer

About this task
By default, the system assigns the value 1 to **Execution Order** for the first step created for a test. When you add a step, the system assigns it the next-highest available integer value. In other words, if the highest **Execution Order** for any step in the test has the value 7, the system assigns 8 to the new step. By editing these values, you can change the order in which the test executes the steps.

Procedure
1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click the row containing the test you want to edit.
   The system displays the **Test**form.
3. In the **Test Steps** related list, edit the values in the **Execution order** column to determine the new order for the steps.
4. Click **Update**.

Copy automated test

Copy an existing test, which you can then re-name and modify.

Before you begin
Role required: atf_test_admin or atf_test_designer

Procedure
1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click the row containing the test you want to copy top open it.
3. Near the top-right of the Test record form, click **Copy Test**.
   An annotation displays confirming that the system has copied the test. After you dismiss this annotation, the system displays a new test record identical to the copied record, with the exception of the **Name**.
4. In the **Name** field, enter the name you want to assign to this new test.
5. Edit the test steps as desired and proceed as you would for any new test.
6. When you are finished making changes, click **Update**.

> **Note:**
> Scope management in ATF tests helps identify and restrict copying of tests in other scopes. If you want to copy a test, you must be in the same scope as the test. See Application Scope for more information.

Run an automated test

After creating an automated test, run it on a non-production instance.

Before you begin

You must have created the test you want to run.
The test execution property must be enabled. You must have an admin or atf_test_admin role to do so.

Note:
The test execution property is disabled by default to prevent running tests on a production system. Run tests only on development, test, and other sub-production instances.

Role required: atf_test_admin, atf_test_designer, or admin

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. If necessary to view the tests list, click **Tests**.
3. Click the row containing the test that you want to run.
   The system displays the **Test** form.
4. Click **Run Test**.

Note:
If the test execution property is not enabled, the **Run Test** button does not appear. In this case, see the annotation at the top of the form, and click the link to enable running tests.

If the test includes a form step (any step involving a UI), or other kinds of UI test steps, the Pick a browser dialog appears before executing the tests. Use it to choose among any currently running test clients, or start a new runner. For more information, review Browser recommendations for all tests and suites. If the test only includes server test steps, the system executes the tests without displaying the Pick a Browser dialog.

What to do next

Monitor the progress of the test in the Run Test progress dialog. When complete, click **Go to Results** (on the Run Test progress dialog) to display the **Test Results** list, where you can view and analyze test results.

Note:
If your test creates data, the system rolls back that data after all steps in the test complete.

Related topics

Cancelling automated tests and test suites

Implementing breakpoints

Breakpoints allow you to pause your test at any step of a test run in order to troubleshoot and test authoring.

Breakpoints are useful when a test fails. You can add breakpoints to the test steps to debug the test. You can also look at the test steps again to find the reasons of the test failure.

Note:
Breakpoints are user-specific. You can see only your self-created breakpoints.

Rollback of test data

Test runs can also be paused immediately before rollback of test data. Check the **Pause before rollback** option to pause the test just before rollback of test data happens after the completion of the test. See Debug an automated test using breakpoints to learn more about using rollback option in a test run.

Debug an automated test using breakpoints

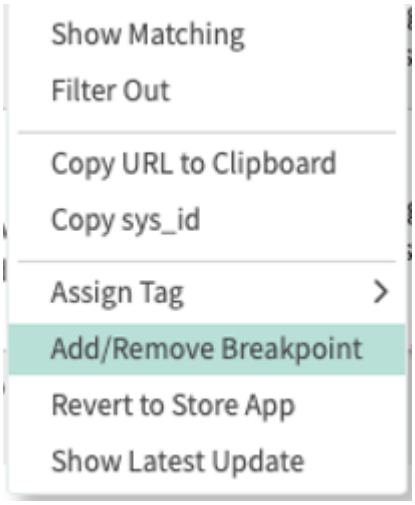
Pause a test to troubleshoot failures or unexpected behavior by adding a breakpoint for a particular test step.

Before you begin

Role required: admin

Procedure

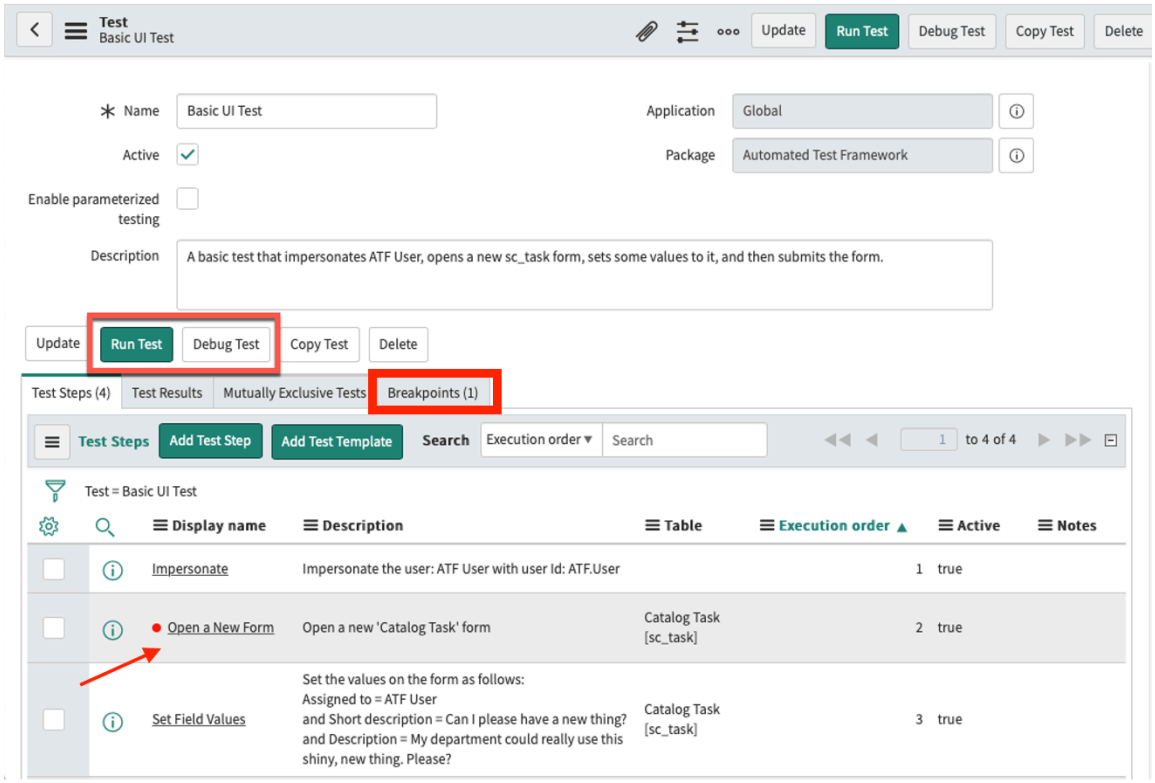
1. Navigate to **All** > **Automated Test Framework (ATF)** > **Tests**.
2. Select the test you want to run.
3. Select Test Steps related list to list out all the test steps for the selected test.
4. Right-click the test step where you want to add or remove a breakpoint.
5. Select Add/Remove Breakpoint from the list.



The test step gets marked for the modified breakpoint.

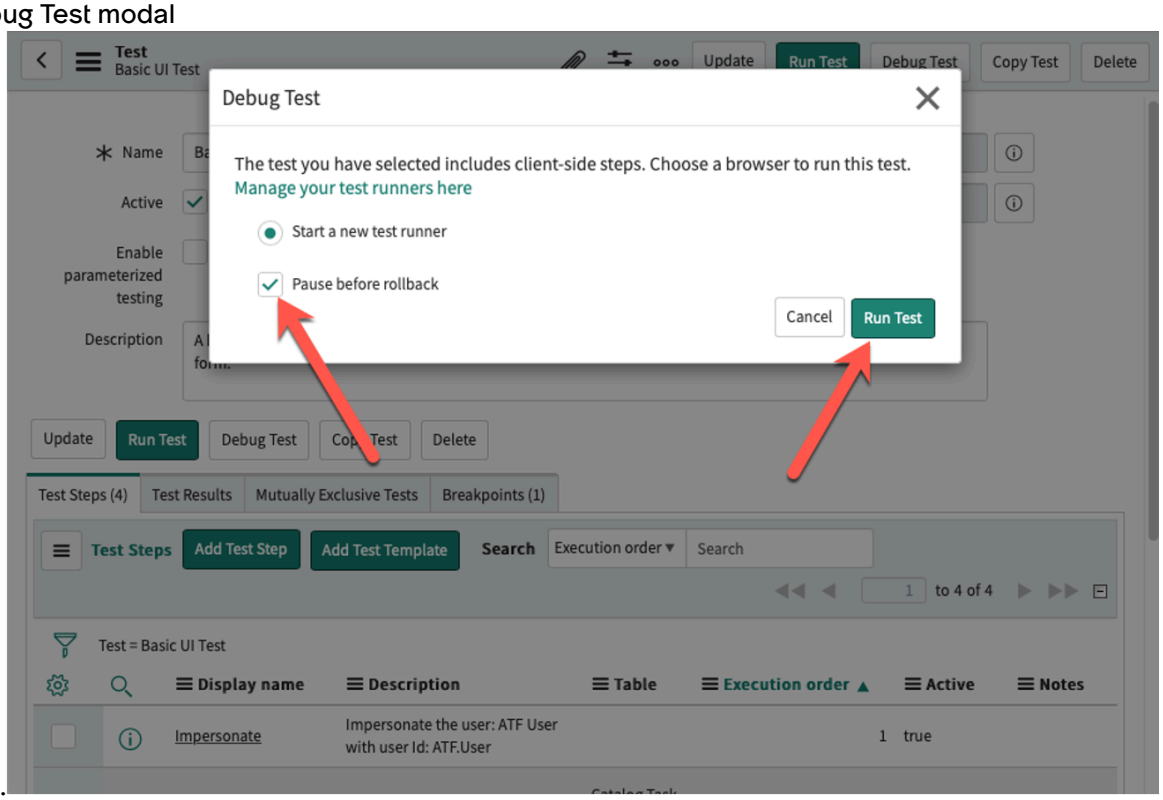
Note:
The breakpoint mark on a step states that if the breakpoint is active and the test is run using **Debug Test**, the test pauses just before the breakpoint test step. The breakpoints are not considered if you click **Run Test** to run the test.

The breakpoints you set on a test are user-specific. Other users can't see and modify your breakpoints on a test, and vice versa.



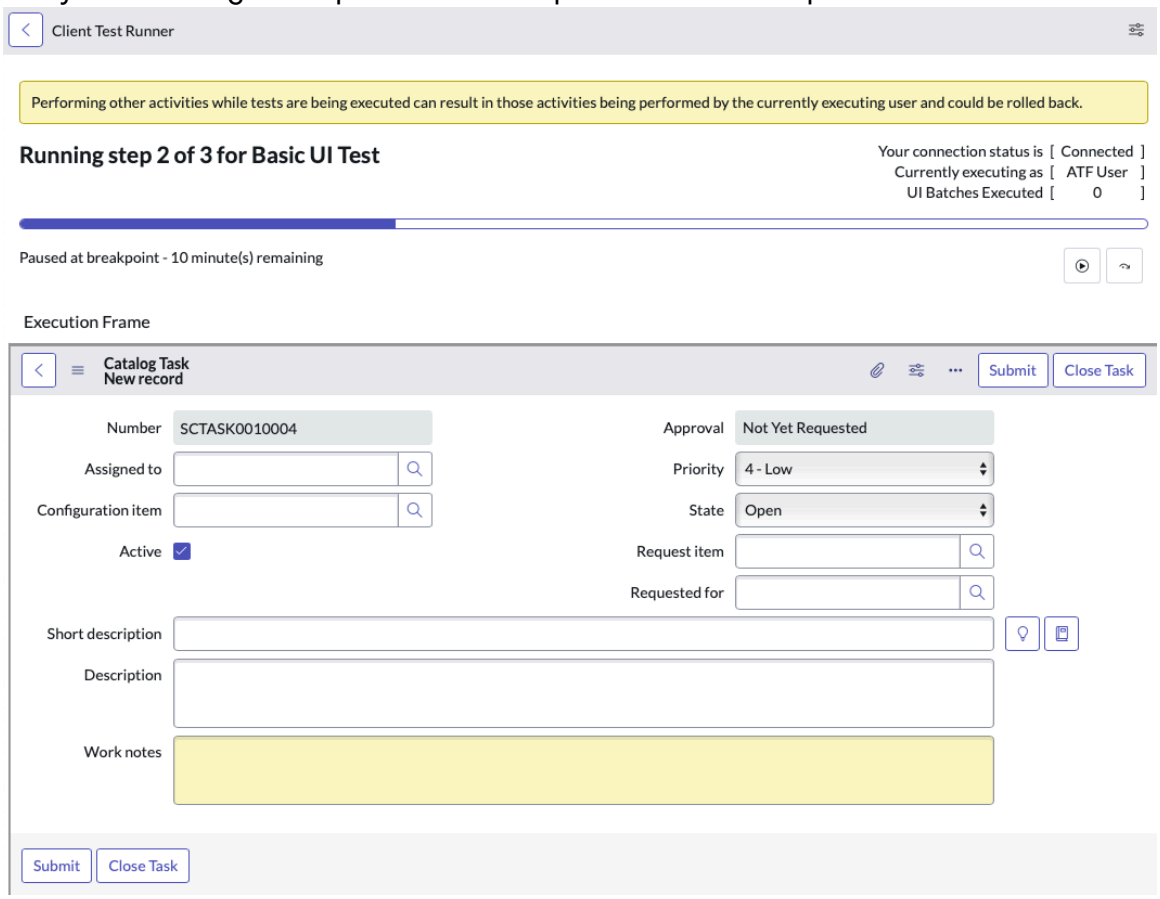
Note:
You can set multiple breakpoints for a test but can have only 1 breakpoint every test step.

6. **Optional:** Click the Breakpoint related list to show all the test steps within a test that have breakpoints.
7. Click **Debug Test** to run the test with breakpoints.
   The Debug Test modal displays.



- Safari – the browser used to run the client-side steps of the selected test
- Pause before rollback – Option to pause the test just before rollback happens after the completion of the test

8. Analyze and debug the steps at which breakpoints have been implemented.



Note:
Every breakpoint causes the running of the test to pause for 10 minutes. You can debug the test step within the allocated 10 minutes and then the test starts running again until it reaches the next breakpoint.

The following options display only when the test is paused for a breakpoint:

- Continue: Test execution continues until the next breakpoint.
- Step over: Test pauses at every step. If you don't want to pause in the current step, select the Step over option to jump to the next step.

View test results and automated test results

View test results from completed test and test suite runs. Carefully consider the results of automated test runs and perform any corrective actions required to resolve any revealed issues.

Before you begin
Role required: none

About this task
Test and test suite results show how long it took the system to execute a test and which steps failed. They can include screenshots of form steps. You can also view reports comparing different runs of the same test suites.

You have multiple options for navigating to the test results, depending on where you are in the user interface. For example, if the Run Test progress dialog is displayed, you can select **Go to Result**. This procedure enables you to view test results from any location in the user interface.

Procedure

1. Navigate to **Automated Test Framework** > **Test Results**.
2. Click a row to access the results for a specific test.
   The system displays the Test results record.
3. View step results for the selected test.
   a. Click the Step Results related list.
   b. Scroll down and click the row for the step result you want to view.
   The system displays the Step results record.
4. View test logs for this test result.
   a. Click the Test Log related list.
   b. Scroll down and click the row for the test log you want to view.
   The system displays the Test logs record.

View automated test results

When viewing test results, step results or test logs, you can allow client errors as ignored or warning entries in the Allowed Client Errors table. This allows test executions to continue past client errors in future test runs.

View results screenshots from an automated test

If the test has a UI component, the system takes screenshots of the UI. View these screenshots to gain further insight into the test results.

Before you begin

You must have run the test whose results screenshots you want to view.

Note:
For best results with screen shots, leave the browser zoom level set to 100%.

Role required: atf_test_admin or atf_test_designer

Procedure

1. Navigate to **Automated Test Framework** > **Test Results**.
2. Click the row containing the test whose results you want to view.
   The system displays the **Test Results**form.
3. View a screenshot taken during the test by finding the screenshot you want in the attachments list.
   Screenshots are named with the word **screenshot** followed by the timestamp (always in UTC time) for when the system recorded the shot. You can match the screenshot to the test step by comparing the step and screenshot timestamps.

View automated test suite results

View results from an automated test suite.

About this task

Note:
You have multiple options for navigating to the test suite results, depending on where you are in the user interface. For example, if the Run Test progress dialog is displayed, you can click **Go to Result** instead. The procedure described here enables you to view test suite results from any location in the user interface.

Procedure

1. Navigate to **Automated Test Framework** > **Suite Results**.
2. Click the row containing the test whose results you want to view.
   The system displays the Test suite results record.

Related topics

Identify and resolve client errors

Allow client errors from test results

Allow client errors from step results

Allow client errors from the test logs

Identify and resolve client errors

Identify client errors and resolve them in client-side scripts.

Before you begin
Role required: admin

About this task
When client errors occur, the Automated Test Framework fails the test on the step that was executing when the error occurred. Even though client-side scripts can fail silently on JavaScript errors while procedures are executing, the error may still impact data, and the procedure being executed. The Automated Test Framework considers these errors as validation failures.

Procedure

1. Navigate to **All** > **Application** > **Automated Test Framework** > **Tests** and run an Automated Test Framework test that interacts with a form.
2. In the test results for this test, check for a step result with the following summary:

   ```
   This step failed because the client error 'DETAILED ERROR
   MESSAGE' was detected on the page being tested. See failing Test
   Logs. To ignore these errors in the next test run, use ‘Add all
   client errors to warning/ignored list’ links.
   ```

   This step result appears only on a step that interacts with the UI.

3. To identify and resolve these script errors, open the developer tools browser console on the Client Test Runner page.

   Note:
   For information about how to open the browser console, see the following article: http://webmasters.stackexchange.com/questions/8525/how-to-open-the-javascript-console-in-different-browsers

4. If you can see the error, try to identify and troubleshoot the problematic client-side script, which may be on any of the following base system tables:

- ServiceNow Client Script
- UI Action
- UI Macro (HTML script)
- UI Page (HTML script)
- UI Policy
- UI Script
- Tables that extend the preceding base system tables

5. Review the script for errors and once you've fixed them, run your test again.

What to do next

Determine the source of the client error by reviewing the script version history. If you customized a base system script, it's possible that the script has new versions that were skipped during upgrade.

Example client errors

There are several types of common client error.

Client JavaScript errors

When a client script causes an error, the browser console displays an error similar to the following example:

```
*********************************************************
A script has encountered an error in render events
TypeError: Cannot read property 'id' of undefined
Script ends. Continuing happily
*********************************************************
[00:00:00.002] onLoad Modify Comments Label
```

In this example, the client script Modify Comments Label caused the error.

Other client script errors

Any other type of script error reports directly to the browser console with any formatting upon occurrence.

```
TypeError: callbacks(id) is undefined
```

Script resource links by Sys ID

In some cases, the console error provides a link to the script resource file using its Sys ID. Following this link may give context to which script had executed it.

```
Uncaught ReferenceError: myobj is not defined
    at incident.do?sys_id=12345678901234567890123456789012
(anonymous)
 @ incident.do?sys_id=12345678901234567890123456789012  <----
 LINK
```

Script access permissions

While identifying problematic scripts, be sure the script has permission to access data. Check:

- Access control rule permissions for tables and fields.
- Application access permissions if the script accesses applications in a private scope.
- Domain separation permissions if domain separation is configured.

Related topics

Allowed client errors

UI test steps

Test user interfaces by mimicking user actions and interacting with the visible components of a page.

Client test runner dependency

UI test steps require an active client test runner to act directly on the visible components of a page. A tester must manually start one or more client test runners for UI testing. Test designers can schedule selecting an open client test runner from a test suite. See Working with client test runners.

Intelligent wait mechanism

UI test steps have an intelligent wait mechanism triggered by UI changes such as clicking a component or setting a value. The wait mechanism requires the UI change to complete before the next UI test step can proceed. Test designers do not need to manually add wait mechanisms between UI test steps.

Custom UI test steps

Test customized user interfaces such as UI pages and UI macros by retrieving their HTML and JavaScript page components and identifying the test actions they support.

Custom UI test steps require the Automated Test Framework to retrieve and identify the testable components from a target web page.

Note:
Now UI is not supported by the Custom UI test steps.

Testable components

Testable page components consist of standard HTML and JavaScript with these characteristics.

Are set or clicked by user interaction

Testable page components allow users to set a value or click them.

Are accessible from the Document Object Model (DOM)

Testable page components are accessible from the DOM and support JavaScript manipulation of the DOM. Custom UI test steps cannot access page components in the shadow DOM.

Are accessible to JavaScript

Testable page components are accessible to JavaScript. Custom UI test steps cannot access page components that interact directly with the operating system such as file fields or display non-HTML content such as Excel or PDF files.

New browser tabs or windows are not supported by Custom UI test steps.

Are not excluded from custom UI testing

Testable page components are not excluded from custom UI testing. Automated Test Framework excludes page components that are already testable by other test step categories and also excludes page components associated with ServiceNow AI Platform features.

Are accessible to the Page Inspector

Testable page components must return results when viewed from the Page Inspector. Test designers can use the Page Inspector to identify the testable components of a page.

Examples of testable page components include these UI elements.

- Buttons
- Links
- Page text
- UI controls
- UI macros
  - ui_date
  - ui_date_time
  - ui_reference
- UI pages
- Wizards

Examples of untestable page components include these UI elements.

Example untestable page components

Reason untestable	Untestable page components
Are not settable or clickable	Hidden controls
	HTML comments
	HTML layout elements such as div, section, and span.
	HTML script elements
Are inaccessible from DOM	Dashboards
	Images
	Lists
	Reports
	Shadow DOM
Are inaccessible to JavaScript	Excel files
	File fields
	PDF files
Are ServiceNow AI Platform features	Flow Designer
	Studio
	Upgrade Monitor

Example untestable page components (continued)

Reason untestable	Untestable page components
Are testable by other test step categories	Form field labels
	Form field values
	Service Catalog
	Workspaces

Settable page components

A settable component is a UI element that has a dynamic value such as a text input field. Settable components support these test actions and test steps.

Test options for settable components

Page Inspector actions	Custom UI test steps
Set Component Value	Set Component Values (Custom UI)
Get Component Value	Assert Text on Page (Custom UI)
	Component Value Validation (Custom UI)
Is Component Disabled	Component State Validation (Custom UI)

Settable components have a data type that determines what values a Custom UI test step can set. For example, a page component intended to display a reference to a particular record can have a reference data type to only display Sys ID values.

Automated Test Framework allows UI developers to specify a data type to use during custom UI testing. UI developers can assign page components a data type to ensure that a test step sets a valid value. These data types are supported.

- Date
- Date Time
- Reference

See Override component data type for more information.

Clickable page components

A clickable component is a UI element that users can interact with by clicking, such as inputs of type check box or radio. Clickable components support these test actions.

Test options for clickable components

Page Inspector actions	Custom UI test steps
Click On Component	Click Component (Custom UI)
Get Component Value	Assert Text on Page (Custom UI)
	Component Value Validation (Custom UI)
Is Component Disabled	Component State Validation (Custom UI)

Clickable components do not have a data type since they do not have dynamic values.

Retrieved page components

Automated Test Framework stores a list of the retrieved page components for each custom UI page you test. Custom UI test steps display the list of retrieved components from the **Component** and **Component values** fields.

By default, the list of page components is static and is only updated when Test designers manually click **Retrieve Components**. Administrators can enable the system property *sn_atf.page_data_capture.enabled* to refresh the list of page components every time a Custom UI test step is run. Enabling this property during test design ensures that your test designers always have access to the most current list of page components. Disabling this property after test design is complete allows your tests to run faster because test steps can use the previously retrieved list of page components.

The ServiceNow AI Platform treats the list of retrieved page components as data and does not include them in update sets or applications files. When transferring tests from one instance to another, test designers must manually retrieve page components again.

Design considerations

Follow these design considerations when testing custom UI pages and page components.

Use the page inspector to identify testable page components

The page inspector determines which page components are available for custom UI testing. Page components that are unavailable to the page inspector are unavailable to custom UI testing.

Navigate to the custom UI you want to test

Use existing test steps to navigate to the target custom UI. For example, to test a Knowledge Base article, use the existing test steps to navigate to a module or to open an existing record. Most custom UI testing requires using existing test step categories as part of the test.

Use the component area to identify page components

The component area describes the HTML layout element containing the component such as a `<div>` or `<section>` element. The area helps test designers distinguish between components by providing the location in the page layout.

Test your custom UI rather than ServiceNow AI Platform UI

The Automated Test Framework prevents custom UI testing of ServiceNow AI Platform features. For example, you cannot test dashboards or graphical designers. Instead, build tests to validate your custom UI pages and elements since you have direct control over these user interfaces.

Use HTML attributes to override page component testing properties

Change the testing properties of a particular page component using HTML attributes that are specific to Automated Test Framework. See Override component test actions.

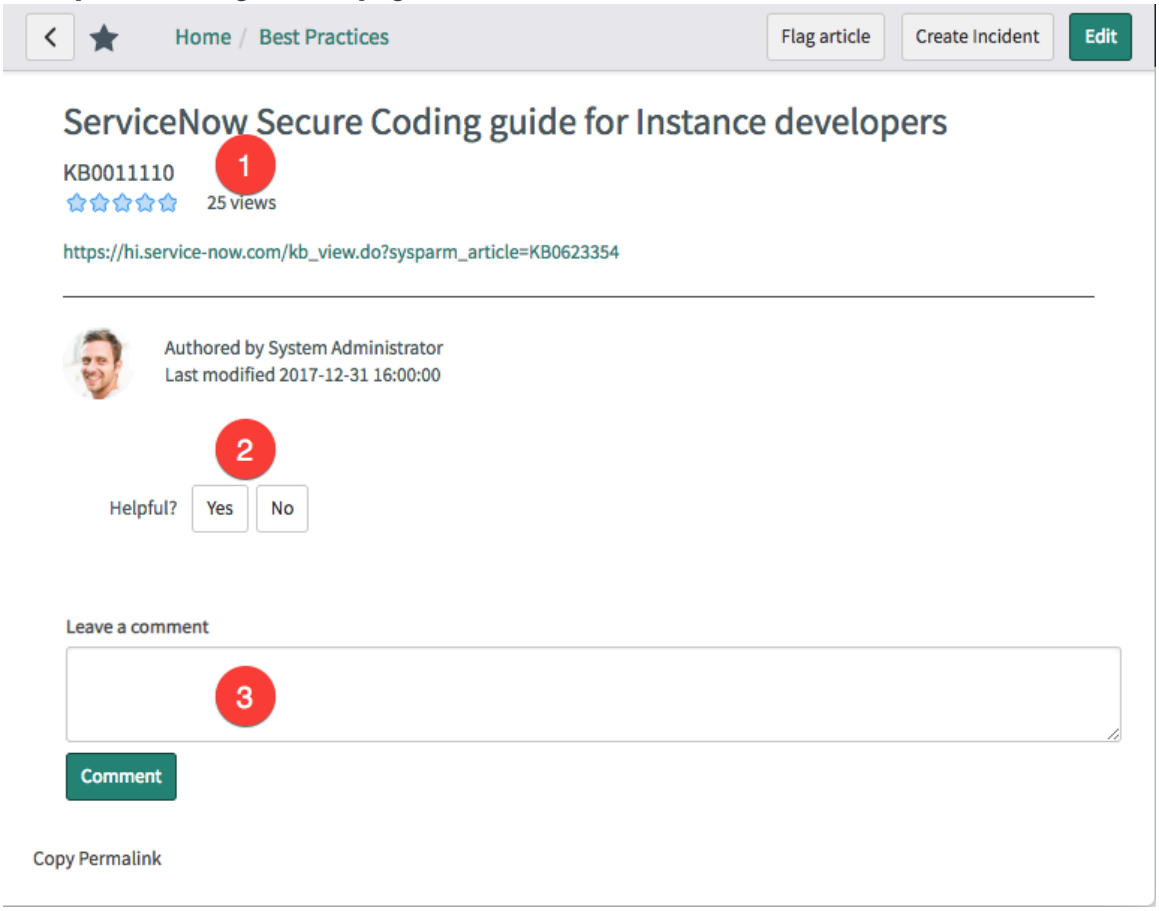
Retrieve page components again when you move tests to another instance

Custom UI test steps do not store UI components as metadata. Testers must manually retrieve page components again when moving tests between instances.

Example custom UI testing

You can use the list of retrieved components to design custom UI test steps. For example, suppose that you want to test reviewing and commenting on a Knowledge Base article. A Knowledge Base article contains several page components that require custom UI steps to test.

Example Knowledge article page



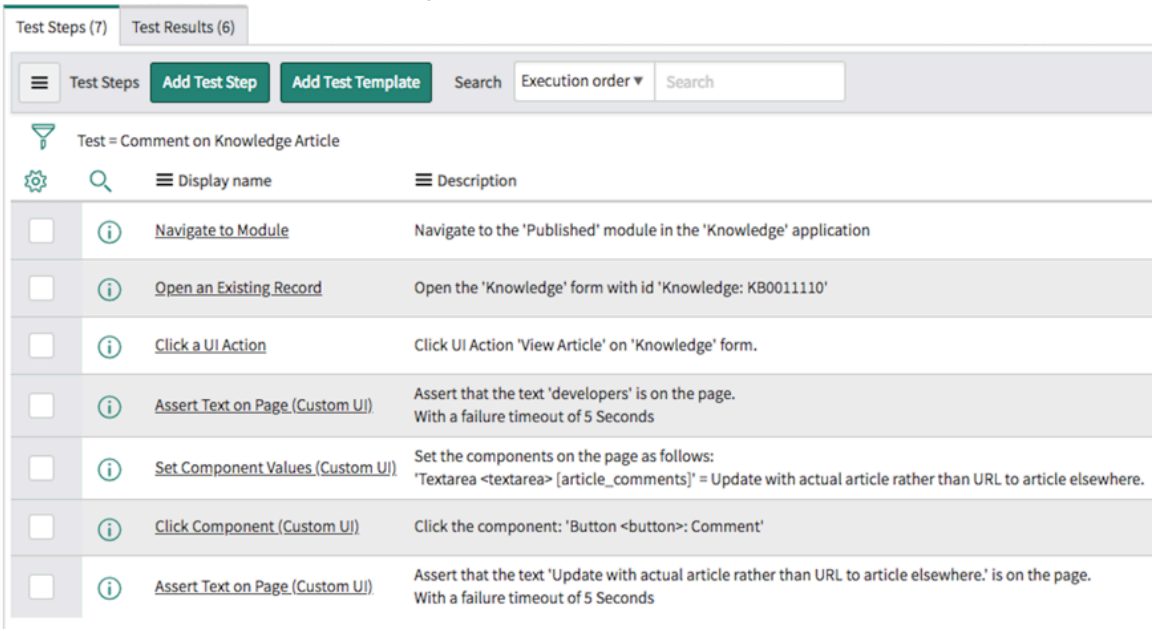
For example, these page components require custom UI test steps.

1. The number of article views.
2. The buttons to mark the article as **Helpful**.
3. The text area to **Leave a comment**.

These steps demonstrate custom UI testing on a Knowledge Base article. The example test consists of these existing and custom UI test steps.

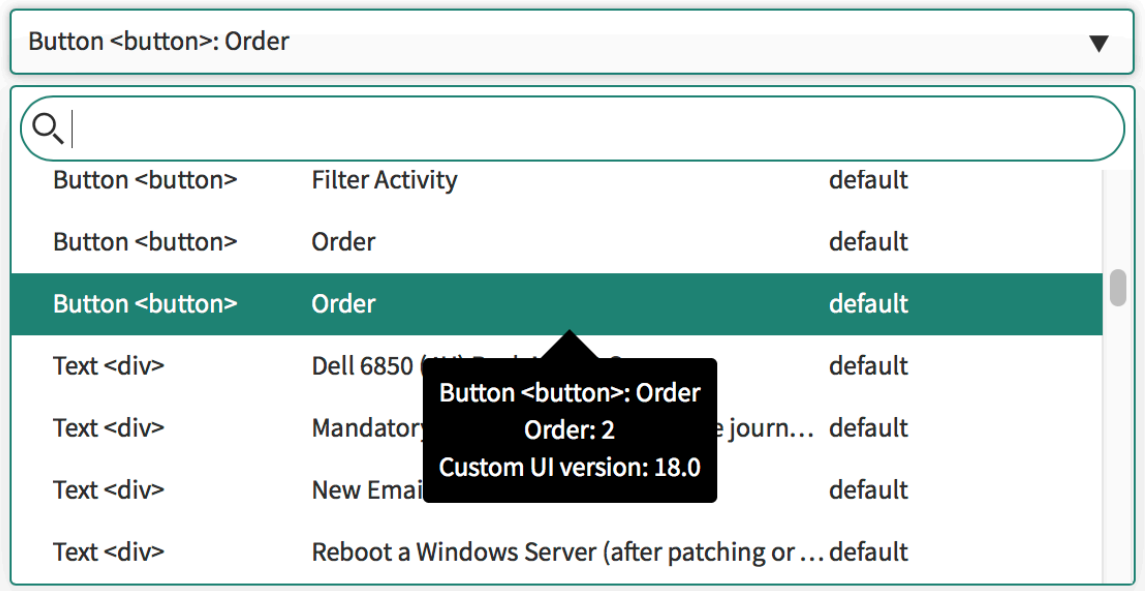
1. **Navigate to Module**. Navigate to the 'Published' module in the 'Knowledge' application.
2. **Open an Existing Record**. Open the 'Knowledge' form with id 'Knowledge: KB0011110'.
3. **Click a UI Action**. Click UI Action 'View Article' on 'Knowledge' form.
4. **Assert Text on Page (Custom UI)**. Assert that the text 'developers' is on the page.
5. **Set Component Values (Custom UI)**. Set the components on the page as follows: 'Textarea <textarea> [article_comments]' = Update with actual article rather than URL to article elsewhere.
6. **Click Component (Custom UI)**. Click the component: 'Button <button>: Comment'.
7. **Assert Text on Page (Custom UI)**. Assert that the text 'Update with actual article rather than URL to article elsewhere.' is on the page.

Example test steps for a knowledge base article test



Custom UI component version and order

When you select a component in any of the custom UI test steps, the Custom UI version might show up. If there are multiple duplicate components, the order of the components show up.



Note:

- The Custom UI version shows up only if there are at least two different components with different Custom UI versions.
- The Order shows up only if there are duplicate components in the component drop-down menu. You can disambiguate them according to the displayed order.
- Both Custom UI version and Order show up if there are multiple duplicate components from different versions.

Identifying components

Implement an alternative way to identify your component by using the sn-atf-id attribute. This is useful if the name or id attribute of your component is dynamic and changes every time a test runs. Add the sn-atf-id attribute with a consistent value to allow ATF identify your component when running a test. It also allows you to identify your component when building a test. For example, in a button component

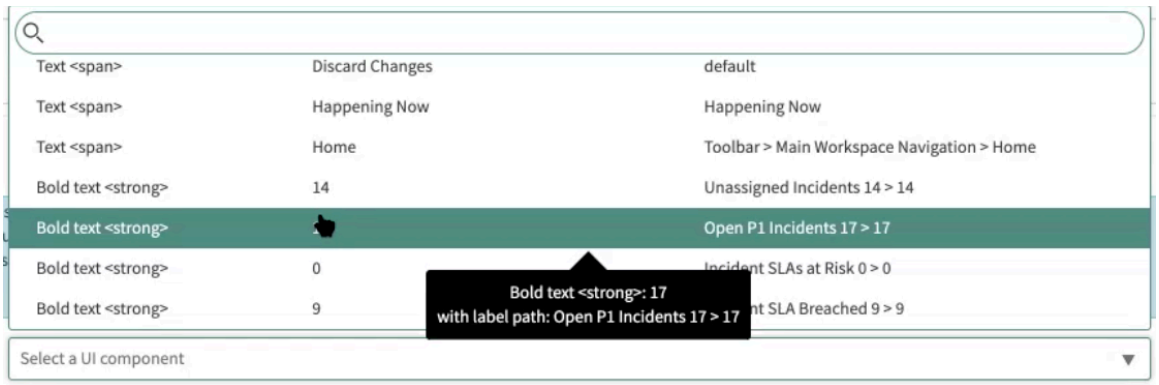
```
<button sn-atf-id="consistentValue">Test</button>
```

Note:
Starting with the Rome release, if you have exactly one component on the page that has sn-atf-id attribute, ATF finds that component irrespective of any other attributes on that component.

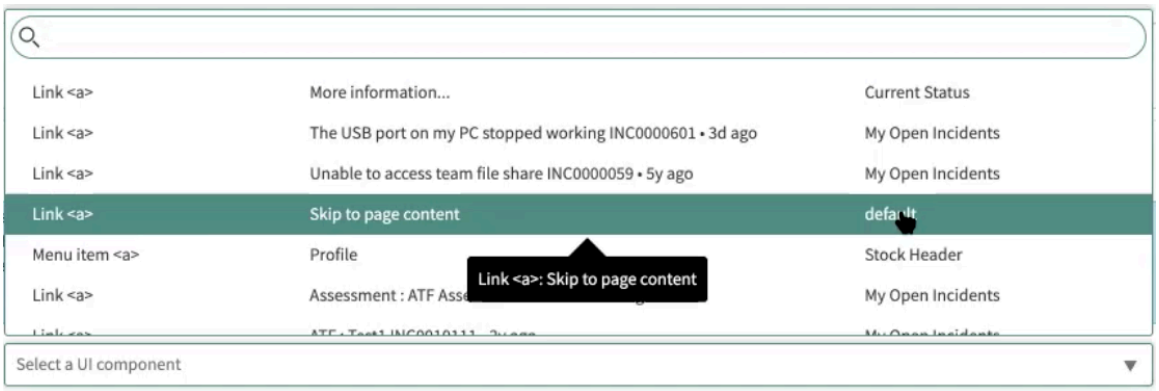
Improve ATF component identification

Identify your component using the label path included in the **Page area** column. Starting with the Rome release, the `sn_atf.element.use_label_path` property has been set to true by default.

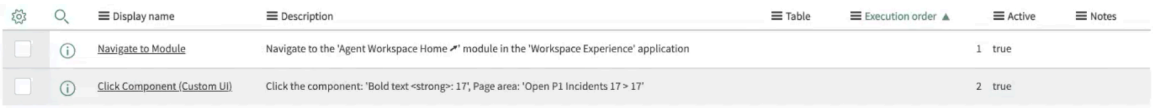
If a component has the `sn-atf-area` attribute, the **Page area** column displays the `sn-atf-area` value. If the `sn-atf-area` attribute is not present, the label path for that component is shown in the Page area column.



Note:
If you have multiple, identically named **Label** values, the label path helps you identify the correct component. If a component doesn't have an `sn-atf-area` attribute or a label path, the value is displayed as default.



When you select the required component from the list and click **Submit**, the description of the test step also gets updated with more details.

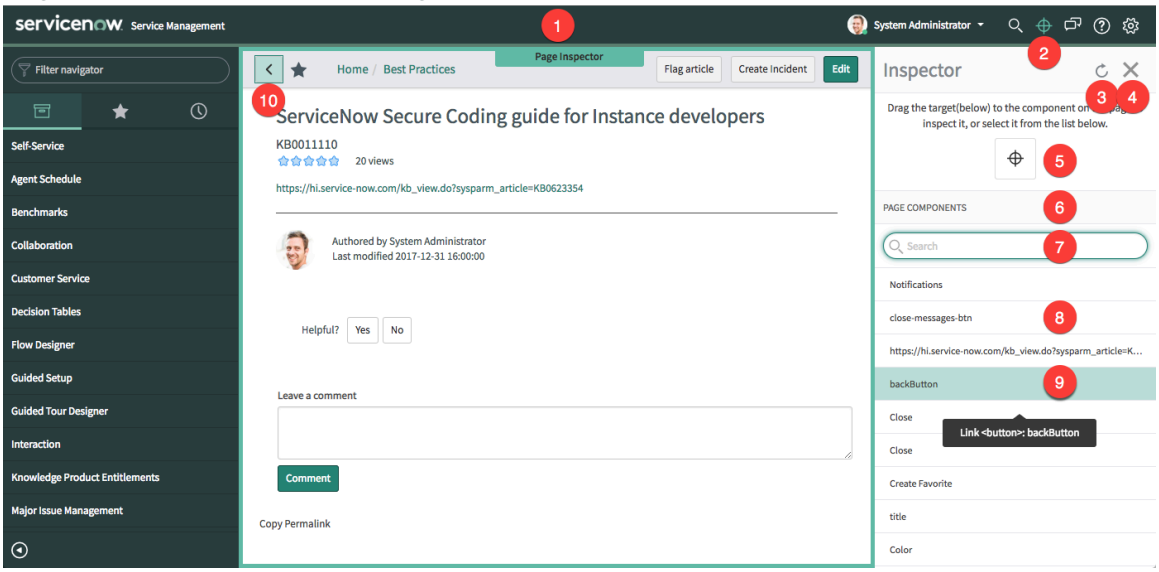


Page Inspector

Identify the HTML and JavaScript page components in your user interfaces that are available for custom UI testing. Enable automated testing by ensuring that your user interfaces only contain testable page components.

The Page Inspector is a developer setting that opens a new pane beside any currently displayed ServiceNow AI Platform page.

Page Inspector view of Knowledge Base article KB0011110



The Page Inspector provides these features.

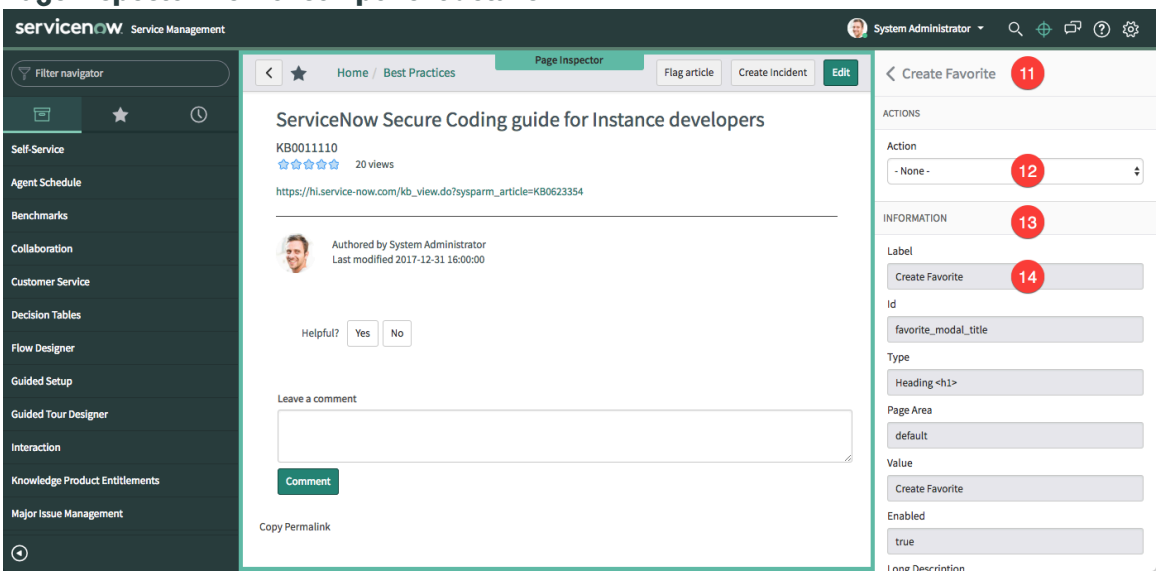
1. A highlighted frame to identify the currently inspected page.
2. A banner icon to display or hide the Inspector pane.
3. A button to refresh the list of page components.
4. A button to hide the Inspector pane.
5. An inspector button to identify specific components on a page.
6. A list of page components available for custom UI testing.
7. A page component search filter.
8. A row to click to see more information about a page component.
9. A highlighted row and tooltip to preview information about the currently inspected component.
10. A highlighted page component to identify the currently inspected component on the page.
11. A component label and back button to return to the list of page components.
12. A list of actions to take on the current page component.
13. A list of attribute information for the current page component.

14. An attribute name and value pair for a page component.

Note:
You can use the Label Path field to show the hierarchical page context around an element while inspecting a component on a page. If there are many "ancestor" label paths of an element, it can display only three. Each label path element is separated by a >. It can be used to differentiate between two similar looking components.

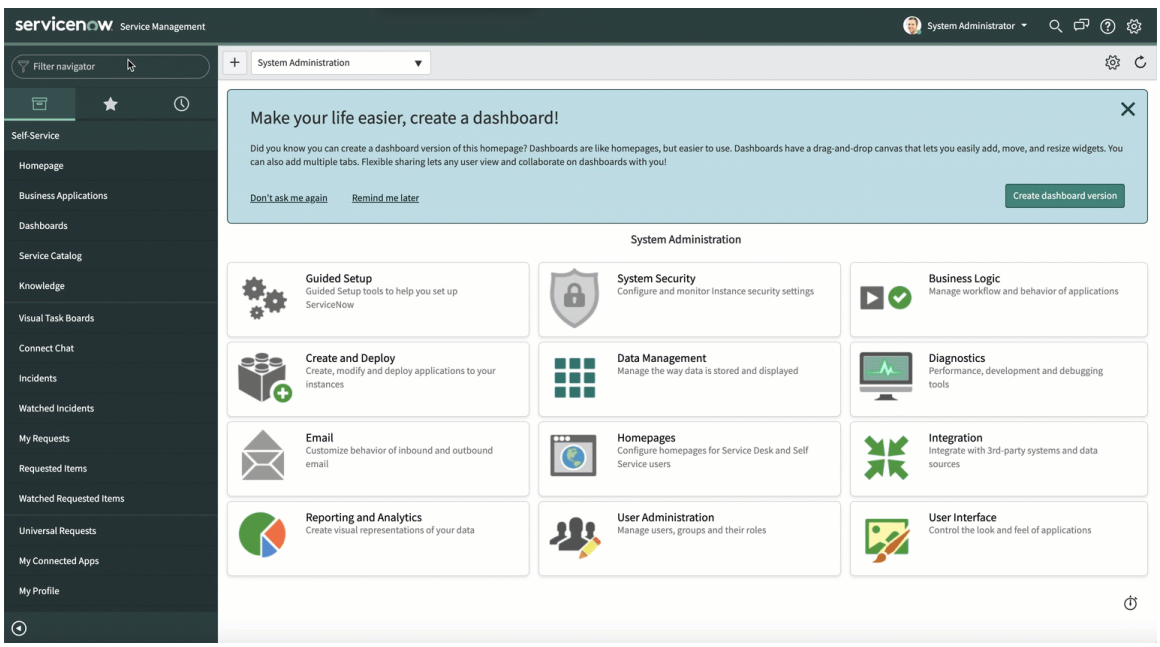
Note:
UI buttons with icons are now visible in the page inspector.

Page Inspector view of component details



Page inspector launcher

The page inspector launcher helps you select and launch a specific page within the page inspector. See Inspect different page types for more information.



Testable page components

The Page Inspector retrieves the list of testable page components when you first load a page. Testable components consist of standard HTML and JavaScript that are accessible to the Automated Test Framework. Test designers can use these components as part of custom UI testing.

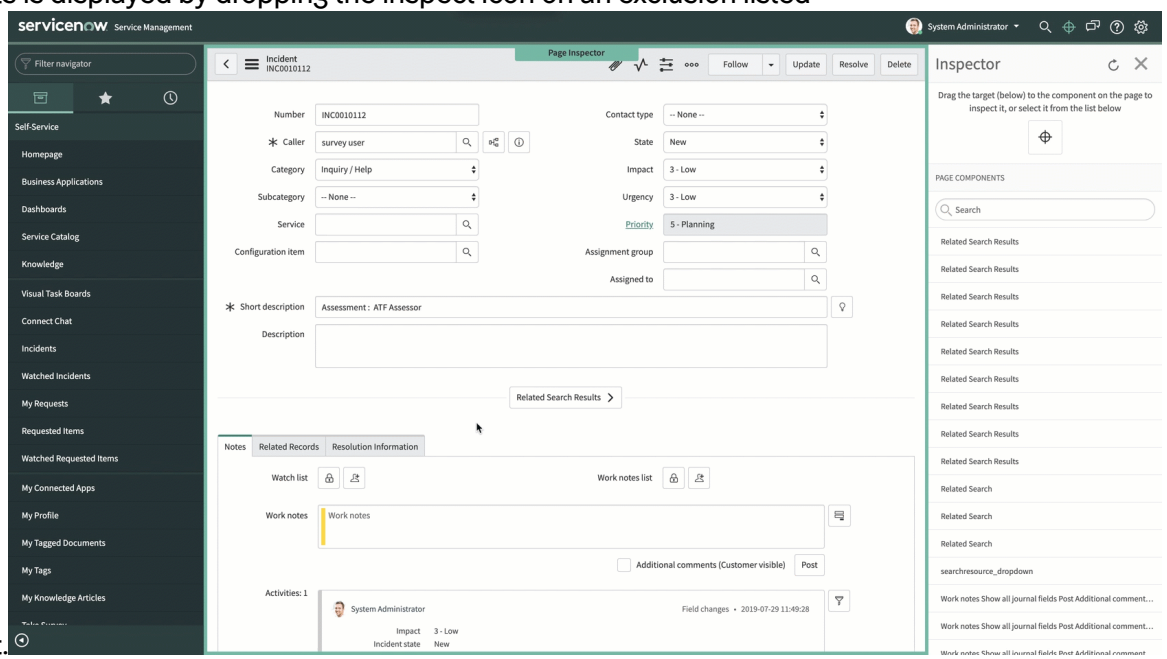
Untestable page components

The Page Inspector identifies these types of untestable page components.

Exclusion listed

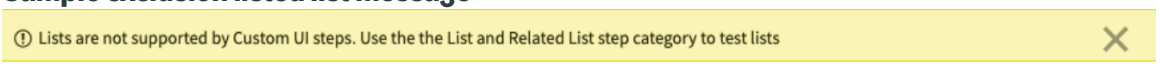
Exclusion listed page components can't be tested using any **Custom UI** test steps. Exclusion listed page components typically include specialized ServiceNow AI Platform user interfaces and components already testable by other means. Test designers can't create custom UI tests for exclusion listed page components. To test these components, they must use another supported test category such as Forms or Service Catalog.

In the Page Inspector, the exclusion listed interfaces are indicated by a grey background color when the inspect icon is dragged over them. A detailed message about the step category that needs to be used to test these components is displayed by dropping the inspect icon on an exclusion listed component.



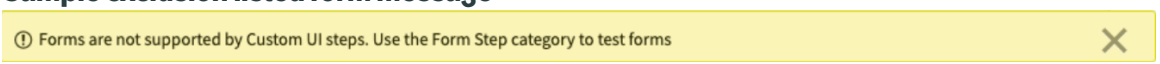
- Lists can't be tested using custom UI test steps. Use List and Related List test step category to test lists.

Sample exclusion listed list message



- Forms can't be tested using custom UI test steps. To test forms, use Form category test steps.

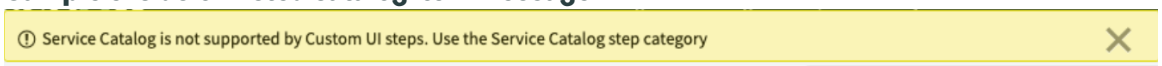
Sample exclusion listed form message



Note:
The UI formatters within forms can be accessed and tested using custom UI test steps. See Create a custom UI test for more information.

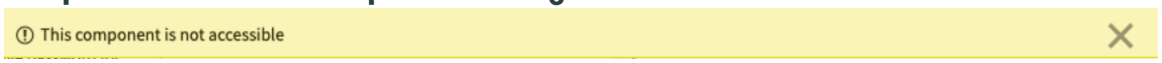
- Service Catalog items can't be tested using custom UI test steps. To test catalog items, use Service Catalog category test steps.

  Sample exclusion listed catalog item message

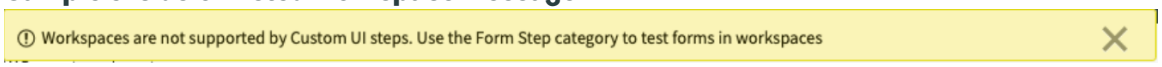
- Certain components are excluded from custom UI pages and are forbidden from being tested.

  Sample exclusion listed component message

- Workspaces can't be tested using custom UI test steps. To test forms in an available workspace, use Form category test steps.

  Sample exclusion listed workspace message

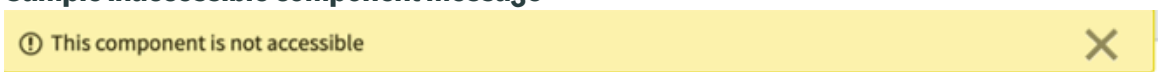
  

Inaccessible

Inaccessible page components are elements that Automated Test Framework either cannot identify or does not have permission to test because of some configuration issue. Inaccessible page components typically include third-party JavaScript libraries or elements with a Shadow DOM. Test designers cannot create custom UI tests for inaccessible page components, but may be able to redesign the page to use components accessible to Automated Test Framework.

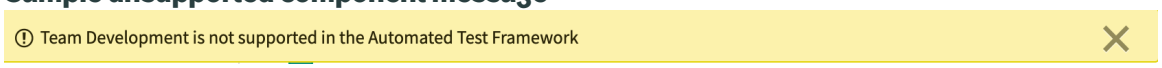
- Certain components are not accessible in the Automated Test Framework

  Sample inaccessible component message

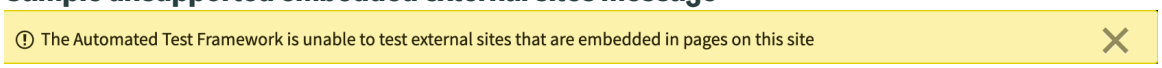
- Team development is not supported in the Automated Test Framework

  Sample unsupported component message

- Testing external sites that are embedded in platform pages is not supported in the Automated Test Framework

  Sample unsupported embedded external sites message

Inspect different page types

Inspect and troubleshoot the functionality of different page types like **UI Pages**, **Service Portal**, **Standard UI**, and **Custom URL** using the Page Inspector.

Before you begin
Role required: admin, atf_test_designer, atf_test_admin

Procedure

1. Navigate to **All** > **Page Inspector** > **Manual Page Inspector**.
2. Select the **Page Type** you want to inspect.

Inputs

Field	Description
UI Pages	Existing UI pages. Select a starting page from the available list.
Standard UI	Standard platform forms, lists, and some UI pages. Select a form or a list or a UI page in the **Starting Page** field. **Note:** The optional **Record** field shows up only if you select a form in the **Starting Page** field.
Service Portal	Any available portal in the instance. Select an available portal and a starting page to be inspected. **Note:** Portal field shows up only if you select Service Portal as the **Page Type**.
Custom	Custom Platform URL to be inspected. **Note:** Don't copy-paste the complete URL onto the **Starting Page URL** field. For example, include only `/home.do` as the input if you want to inspect `https://instance.service-now.com/home.do`

3. Click **Inspect**.

Enable and use the page inspector

Enable a developer setting to inspect UI pages that open within the platform. Use the Manual Page Inspector to inspect pages that open in a new tab, such as Service Portal pages.

Before you begin
Role required: admin

About this task
Identify the HTML and JavaScript page components in your user interfaces that are available for custom UI testing. Enable automated testing by ensuring that your user interfaces only contain testable page components.

Procedure

1. Enable the Page Inspector depending on the type of custom UI page you need to inspect.
   Note that some custom UI pages open in a new tab and can be inspected only manually.

2. Select a component to inspect.
   - Drag the inspect icon ( ) from the Page Inspector pane to a component on the page.
   - Select an available component from the Page Inspector pane.

   The Page Inspector retrieves the list of testable page components when you first load a page. Testable components consist of standard HTML and JavaScript that are accessible to the Automated Test Framework. Test designers can use these components as part of custom UI testing.
3. View component information and perform actions available in the **Action** field.
   Performing available actions helps you manually confirm which test steps are available to a component. For example, when you select **Click On Component**, the Page Inspector selects the component and displays the resulting page.

What to do next

Create a custom UI test

Create a custom UI test

Test components in custom UI pages.

Before you begin

- Use the Page Inspector to identify testable custom UI components. See Enable and use the page inspector.
- Role required: admin

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click **New**.
3. Enter a name and description for your test.
4. Click **Save**.
5. In the Test Steps related list, click **Add Test Step**.
6. Add test steps to navigate to the target custom UI page.

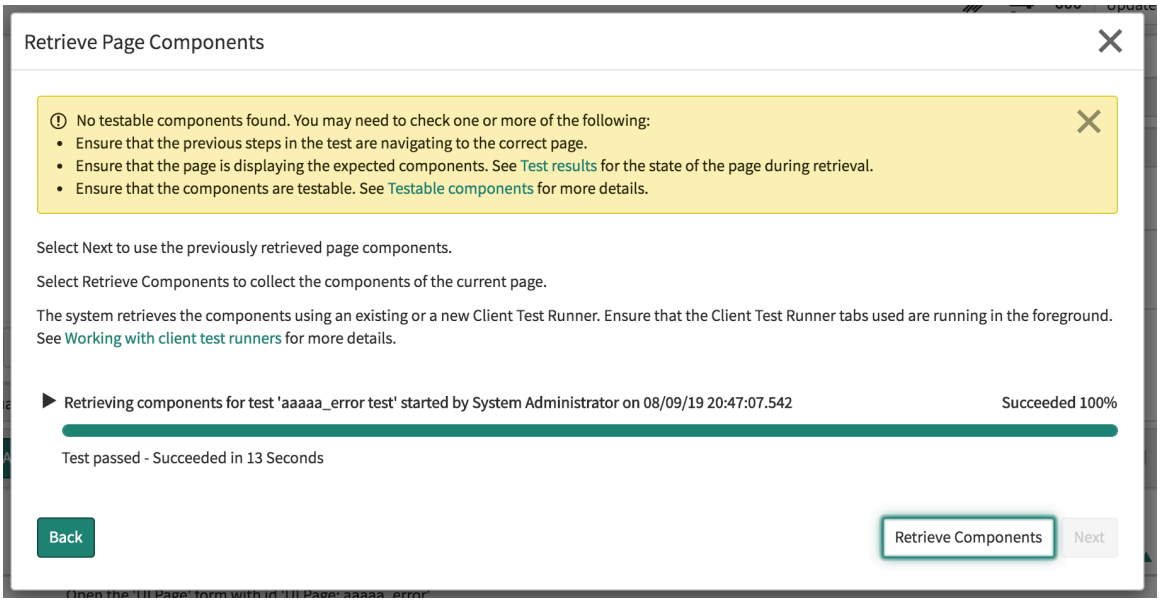
   Example
   Use the **Navigate to Module** step to open a page that has an application navigator module.
   Use the **Open Service Portal Page** step to open a portal page.
   To open a UI page, use these test steps:

   a. **Open an Existing Record**: Open the record for the page.

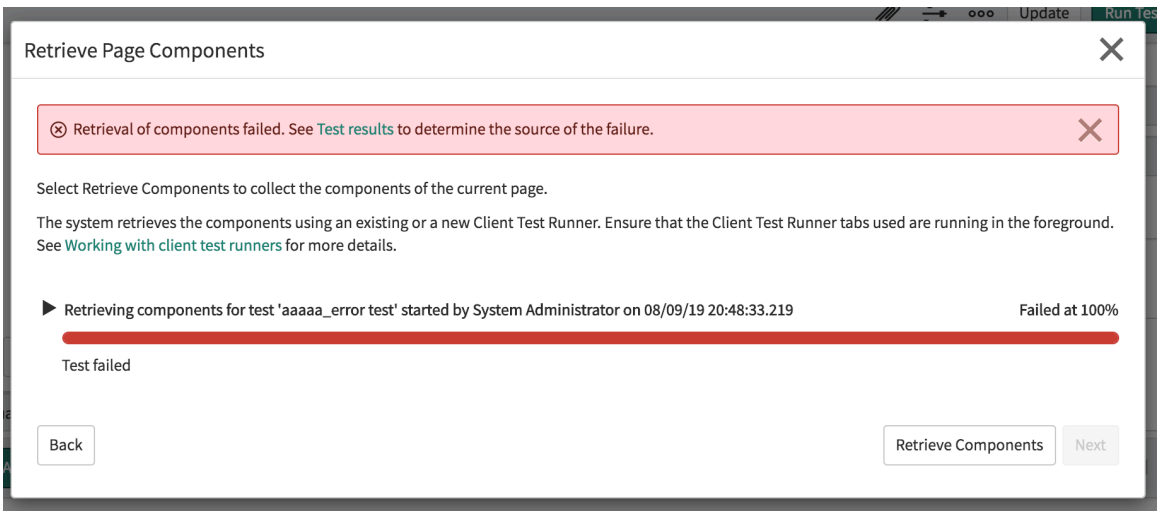
   b. **Click UI Action**: Click the **Try It** UI action to open the page.
7. Add test steps from the Custom UI category to validate the behavior of custom UI components.
   For a list of available Custom UI test steps, see Custom UI category.
8. Retrieve UI components when adding a test step.

Identify the testable page components on a custom UI page by retrieving a list of UI components for test steps. You can collect the components of the current page by selecting **Retrieve Components**.

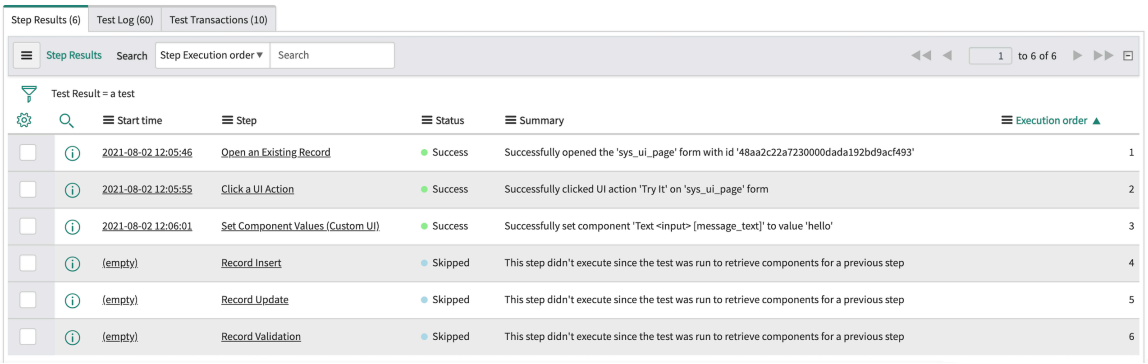
The system retrieves the components using an existing or a new client test runner. Ensure that the client test runner tabs used in the retrieval process are running in the foreground. For tests that don't yield any components after selecting **Retrieve Components**, the following warning message is displayed and the **Next** button is disabled.



If you have already retrieved a list of page components, click **Next** to use the previously retrieved list. You can also click **Retrieve Components** to rerun the current steps and refresh the list of testable page components. If the test fails while retrieving components, the following warning notification is displayed. You can review the test results by clicking the link in the notification.



Note:
If you need to update a step in a test, click **Retrieve Components** for that step and the test runs only up to that particular step. The steps beyond the selected step don't run. The test passes if all the updated steps pass without failure.



Automated Test Framework stores the list of page components and displays the list in the **Component values** or **Component** field on the test step form. The field displays this information about each component:

Field	Description
Type	HTML element description and tag. For example, `Text<input>`.
Label	HTML component label.
Value	Value of the component. For example, if the component is a search input, the value might be **Search**.
Page area	HTML layout element that contains the component such as a `<div>` or `<section>` element.

If you create a parameterized test that includes Custom UI test steps, the system only uses the first data set to retrieve components.

What to do next
Consider adding tests to a test suite. For more information, see Building and running automated test suites.

Override component test actions

Change the testing properties of a particular page component using HTML attributes that are specific to Automated Test Framework.

Before you begin
Role required: admin

About this task
When Automated Test Framework retrieves a component, it determines which interactions it supports, such as whether it is a settable or clickable component. If the component is settable, Automated Test Framework determines the field type that can be set. If Automated Test Framework incorrectly determines your custom component's actions or field types, or your component contains multiple DOM elements that should be treated as one entity, explicitly set them using HTML attributes that are specific to Automated Test Framework.

Using sn-atf-clickable and sn-atf-settable attributes

Use `sn-atf-clickable` and `sn-atf-settable` attributes to specify that an element and its child elements should be treated as a custom clickable or custom settable component.

Before you begin

Role required: admin

Procedure

1. Open the custom UI page you would like to test.
2. Add the `sn-atf-clickable` or `sn-atf-settable` attribute to the element being tested.

Example

```
<div sn-atf-clickable="true" sn-atf-disabled
 id="customClickable">
 <button id="customButton">Click me</button>
</div>
```

```
<div sn-atf-settable="true" id="customSettable"
 sn-atf-component-value="A default value">
 <input id="customInput" value="A default value"></input>
</div>
```

Note:
You can use either `sn-atf-clickable` or `sn-atf-settable` attribute to specify if an element should be treated as a custom clickable or custom settable component. You can't use both attributes on the same element.

3. Use either the `sn-atf-clickable` or `sn-atf-settable` attribute.

   - `sn-atf-clickable`: If you added the `sn-atf-clickable` attribute, ATF clicks the component by sending an `sn-atf-click` event to the DOM element with the `sn-atf-clickable` attribute. You should add an event listener (for example, using `addEventListener`) to this DOM element, and implement your custom click logic for the component. You can interact with a custom clickable component using Click Component test step.

     Clickable component attributes

Attribute name	Description
sn-atf-disabled	The presence of this optional attribute (regardless of its value) tells ATF that this component is disabled **Note:** If this attribute is missing, ATF considers that this component is enabled by default.
sn-atf-component-value	A string or JSON object that tells ATF the current value of this component

   - `sn-atf-settable`: If you added the `sn-atf-settable` attribute, ATF sets the component value by sending an `sn-atf-setvalue` event to the DOM element with the `sn-atf-settable` attribute. You should add an event listener (for example,

using `addEventListener`) to this DOM element, and implement your custom set value logic for the component. The value that needs to be set can be accessed with `event.detail.newvalue`. The `event` argument is passed to your event handler. You can interact with a custom settable component using Set Component Values test step.

Settable component attributes

Attribute name	Description
sn-atf-disabled	The presence of this optional attribute (regardless of its value) tells ATF that this component is disabled. **Note:** If this attribute is missing, ATF considers that this component is enabled by default.
sn-atf-component-value	A string or JSON object that tells ATF the current value of this component.
sn-atf-data-type	Optional type of field to present to user when building a step. It defaults to string. For example, `glide_date_time`, reference, boolean, etc.
sn-atf-data-type-params	JSON object with more data type details.

Example:

```
//A custom clickable component

<div sn-atf-clickable="true" sn-atf-disabled
 id="customClickable">
 <button id="customButton">Click me</button>
</div>
<script>
 var customClickableDiv =
 document.getElementById("customClickable");
 customClickableDiv.addEventListener('sn-atf-click', function()
 {
  document.getElementById('customButton').click();
 });
</script>
```

```
//A custom settable component

<div sn-atf-settable="true" id="customSettable"
 sn-atf-component-value="A default value">
 <input id="customInput" value="A default value"></input>
</div>
<script>
 var customSettableDiv =
 document.getElementById("customSettable");
 customSettableDiv.addEventListener('sn-atf-setvalue',
 function(event) {
  var newValue = event.detail.newValue;
  document.getElementById("customInput").value = newValue;
```

```
 });
</script>
```

Using sn-atf-class attribute

Use the `sn-atf-class` attribute to specify the JavaScript object to use when testing a custom clickable or settable component. Write a custom JavaScript object to specify the test actions available for a custom component.

Before you begin

Role required: admin

About this task

Test designers can manually specify the test actions available for a custom component by writing a custom JavaScript object and assigning the component a `sn-atf-class` attribute. Set the value of the attribute to the name of the JavaScript object containing the component test actions. Testable custom components must be either clickable or settable, and this classification determines the functions and properties your JavaScript object requires. See Custom UI test steps for testable page component requirements.

Procedure

1. Open the custom UI page you would like to test.
2. Add the `sn-atf-class` attribute to the element being tested and set the value to the name of the JavaScript object embedded in the page that handles getValue(), setValue(), click(), or isDisabled() functions.

   Example

   ```
   <div sn-atf-class="MyClickableComponent">
       <label
    for="a_clickable_checkbox">MyClickableComponent</label>
       <input type="checkbox" id="a_clickable_checkbox"
    checked="true"/>
   </div>
   ```

3. Create the JavaScript object specified in the `sn-atf-class` attribute and add the functions and attribute needed to identify your custom page component as either a clickable or settable page component.

   Clickable component functions

Function name	Description
initialize()	Gets the initial values of the component. Enter: `$super(element, area)`
click()	Selects the component. Returns a JSON object with these properties: `success`: true if the component can be clicked. **Note:** Triggers the UI test step intelligent wait mechanism.
getValue()	Gets the value of the element. Returns a JSON object with these properties:

Function name	Description
	○ `success`: true if the value is retrieved. ○ `result`: the value of the component.
`isDisabled()`	Whether the component is disabled. Returns a JSON object with these properties: ○ `success`: true if the component is disabled. ○ `result`: true if the component is disabled.

Settable component functions

Function name	Description
`initialize()`	Gets the initial values of the component. Enter: `$super(element, area)`
`setValue(newValue)`	Sets the value of the component. See the second example below. Returns a JSON object with these properties: `success`: true if the value is set. **Note:** Triggers the UI test step intelligent wait mechanism.
`getValue()`	Gets the value of the element. Returns a JSON object with these properties: ○ `success`: true if the value is retrieved. ○ `result`: the value of the component.
`isDisabled()`	Whether the component is disabled. Returns a JSON object with these properties: ○ `success`: true if the component is disabled. ○ `result`: true if the component is disabled.

Settable component attribute

Attribute name	Description
`isSettable: true`	Identifies the component as a settable page component.

4. When creating your custom component in Jelly, add `<g2:atf_only>` tags around the JavaScript object specified in the `sn-atf-class` attribute.
These tags ensure the system only runs the JavaScript object during automated testing.

Example:

```
//A custom clickable component

<form>
```

```
    <div sn-atf-class="MyClickableComponent">
        <label
 for="a_clickable_checkbox">MyClickableComponent</label>
        <input type="checkbox" id="a_clickable_checkbox"
 checked="true"/>
    </div>
</form>
<script>
var MyClickableComponent = {

    // The constructor must have this signature, but you can
 perform additional setup after the $super(element, area) call
    initialize: function($super, element, area) {
        $super(element, area);
    },

    click: function() {
        document.getElementById('a_clickable_checkbox').click();
        return {success: true};
    },

    // The function returns an object with a result attribute of
 type String
    getValue: function() {
        var isChecked =
 document.getElementById('a_clickable_checkbox').checked ?
 "true" : "false";
        return {success: true, result: isChecked};
    },

    // The function returns an object with a result attribute of
 type Boolean
    isDisabled: function() {
        if
 (document.getElementById('a_clickable_checkbox').disabled)
            return {success: true, result: true};

        return {success: true, result: false};
    },

};
</script>
```

```
//A custom settable component

<form>
    <div sn-atf-class="MySettableComponent">
        <label
 for="a_settable_checkbox">MySettableComponent</label>
        <input type="checkbox" id="a_settable_checkbox"
 checked="true"/>
    </div>
</form>
<script>
var MySettableComponent = {

    // This attribute is required for settable components
```

```
    isSettable: true,

    // The constructor must have this signature, but you can
 perform additional setup after the $super(element, area) call
    initialize: function($super, element, area) {
        $super(element, area);
    },

    // The value parameter is a string
    setValue: function(value) {
        document.getElementById('a_settable_checkbox').checked =
 (value == "true");
        return {success: true};
    },

    // The function returns an object with a result attribute of
 type String
    getValue: function() {
        var isChecked =
 document.getElementById('a_settable_checkbox').checked ?
 "true" : "false";
        return {success: true, result: isChecked};
    },

    // The function returns an object with a result attribute of
 type Boolean
    isDisabled: function() {
        if
 (document.getElementById('a_settable_checkbox').disabled)
            return {success: true, result: true};

        return {success: true, result: false};
    },

};
</script>
```

Reference and record picker

Use custom UI steps to manipulate the values of the `sn-reference-picker` and `sn-record-picker` angular directives. The value on a reference picker returns the `sys_id` of the chosen record. The value on a record picker returns the value field chosen for that record picker. Both elements can be set by selecting a record to set as their value.

Override component data type

Use the `sn-atf-data-type` and `sn-atf-data-type-params` attributes to override the type of field displayed in a Set Component Value test step.

Before you begin

Role required: admin

About this task

Settable components have a data type that determines what values a Custom UI test step can set. For example, a page component intended to display a reference to a particular record can have a reference data type to only display Sys ID values.

Procedure

1. Open the custom UI page you would like to test.
2. Add the `sn-atf-data-type` attribute to the settable component and set the value to the field type you would like displayed in the Set Component Value test step.

   This attribute contains a string of the testable data type. The available values include those listed in the following table.

Attribute value	Description
glide_date	Contains a string specifying a particular day.
glide_date_time	Contains a string specifying a particular day and time of day.
reference	Contains a Sys ID to a related record. This data type requires specifying additional information in the `sn-atf-data-type-params` attribute.

3. Add the `sn-atf-data-type-params` attribute to provide additional information when the value of `sn-atf-data-type` is `reference`.

   This attribute contains a string of JSON formatted key-value pairs. Available key-value pairs include those listed in the following table.

Key	Value
reference	Name of the table that contains the records you want the reference field to display. For example, add `"reference":"incident"` to display records from the Incident table.
reference_qual	Filter to apply to the query. For example, add `"reference_qual":"active=true"` to display only active Incident records. See Reference qualifiers

Example:

```
<input id="someTextField" value="someSysId"
 sn-atf-data-type="reference"
 sn-atf-data-type-params='{"reference":"incident","reference_qua
l":"active=true"}'/>
```

Select2 functionalities in ATF

Use the Select2 component to search and select your option from a drop-down menu easily.

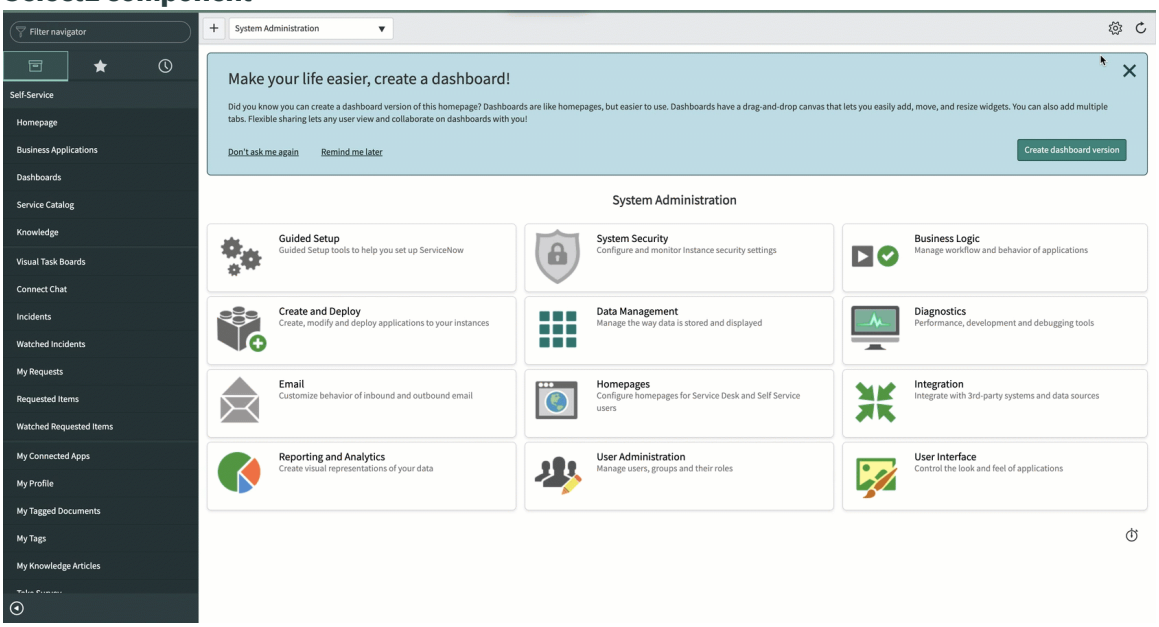
You can set the value of select2 in a test. The test then searches for the value in the search bar of Select2. The first valid result in the component drop-down gets selected.

Drag and drop the page inspector on the required drop-down menu. The right panel for the selected component opens. You can specify the text you want to search in the set component value step within the page inspector.

Select the **Set Component Value** option under **Action** to search for a component. Type in the text in the right panel and click **Submit**. This enables the code to open the select2 to search for that component and pick the first option that shows up.

Note:
If multiple, similarly named component options show up on searching a term, only the first option is selected.

Select2 component



Limitations of the select2 support

- Only certain versions (3.5.1-3.5.4 and 4.0.0-4.0.13) of the select2 library are supported.
- The select2 support depends on the jquery library. So, if you try to set a value of select2 and the name of the jquery library is changed, it causes the test to fail.
- Searching through Select2 won't work when there is no search bar.

Design considerations

Search and select your options efficiently with the following design considerations:

- It is recommended to use uniquely searchable test data
- Wait for the current element to be set before proceeding to set the next element while setting the value of Select2 via the Set Component Values (Custom UI) test step
- Prevent failing of tests by avoiding jquery library name change
- Select2 Adapter and Decorator features are not supported

Browser recommendations for Automated Test Framework

Configure client test runner browsers to run automated tests and avoid performance degradations.

Periodic browser restarts

These browsers have memory-management limitations that make it necessary to occasionally close and restart the browser when running the client test runner.

- Internet Explorer
- Edge
- Older versions of Firefox

How often you should close the browser depends on the memory allocation in the browser application.

Browser CPU throttling

Some browsers throttle CPU usage for windows that are out of focus. Follow these guidelines to avoid CPU throttling issues.

- Run each client test runner in its own browser window.
- Ensure that the client test runner browser window is always partially visible on the screen.
- Ensure that the system screen is not locked or shut off.

Browser zoom level

Client test runners take screen shots as they run tests. For best results with screen shots, leave the browser zoom level set to 100%.

OS X CPU throttling

On OS X with the client test runner on Chrome or Safari: If the screen is locked or the client test runner tab is not shown, when the system attempts to run the test suite, tests run significantly slower and may time out. For best performance, run client test runners for scheduled suites in a virtual machine (VM) environment in which the screen does not become locked or disabled.

Rollback in browser sessions

The session cookies roll back all the changes made during a test. When a test is running, everything performed in that session is recorded for rollback. Don't modify your instance when a test is running in the same browser session. For example, if you modify records while a test is running in the same session, the changes are rolled back after the test completes. If you navigate around in other tabs in the same session, your work may be rolled back and interfere with tests that rely on implicit navigation.

Parallel testing

Follow these guidelines to avoid issues when running multiple tests in parallel.

Run each client test runner in an incognito or private window

Because parallel tests roll back all changes tied to the same browser session, it's possible for legitimate changes made in another browser tab to be rolled back during parallel testing. To prevent unwanted rollback of changes, always run client test runners in their own browser session. Opening client test runners in an incognito or private window ensures that they always have their own browser session.

Close client test runner windows when testing is complete

To prevent unwanted rollback of changes, always close client test runners after testing is complete. Closing the browser window ensures that test rollback doesn't revert any legitimate changes made in another browser tab.

Working with client test runners

If an automated test includes steps that involve a form or any other user-interface (UI) element, it runs those steps in a browser tab or window called a test runner or client test runner.

The Automated Test Framework supports two types of client test runners: Client Test Runners for manually started tests and Scheduled Client Test Runners for tests started by a schedule.

When test execution is enabled, clicking the Client Test Runner module opens the client test runner in the current browser session. If tests are waiting to be run, the Client Test Runner runs a waiting test. If no test is running, the message Waiting for a test to run displays in the client test runner.

While the client test runner is idle, it checks every five seconds for waiting tests to start. This ensures that the system runs any tests it may have been unable to start because no client with the proper configuration was available.

Note:
The client test runner monitors for tests from the current session and runs those tests as the logged-in user (unless it executes an Impersonate User step). If you start a client test runner, log out from the current session, and then log in again, the client test runner executes using the new session.

When the client runner is active, it displays the activity of the currently running test in the **Execution Frame**.

Note:
You are now allowed to execute multiple tests at a given time.

Test execution property

To work with the client test runner module, the test execution property must be enabled.

Note:
By default, the system property that is used to run automated tests is disabled to prevent you from accidentally running these tests on a production system. To avoid data corruption or an outage, run tests only on development, test, and other non-production instances.

If the test execution property is disabled when you select this module, the system displays a message and a link to the automated test framework properties page where you can enable it.

Additional debugging functionality

If you have enabled additional debugging functionality, the client test runner module displays two tabs: **Execution Frame** and **Debug Info**. The **Execution Frame** displays the information normally shown by the client test runner and the **Debug Info** displays additional debugging information.

The system takes screen shots from the tests in the **Execution Frame** tab and records them to the test result record.

Browser recommendations for all tests and suites

- Some browsers have memory-management limitations that make it necessary to occasionally close and restart the browser when running the client test runner. These browsers include Internet Explorer, Edge, and older versions of Firefox. How often you should close the browser depends on the memory allocation in the browser application.
- Some browsers have features that throttle CPU time. To avoid problems, follow these guidelines:
  - Run the client test runner in its own browser window.
  - Keep the client test runner at least partially visible on the screen.
  - Make certain the screen is not locked or shut off.
- The client test runner takes screen shots as the tests run. For best results with screen shots, leave the browser zoom level set to 100%.

Browser recommendations for scheduled suites

The client test runners for scheduled suites have additional browser requirements.

- On OS X with the client test runner on Chrome or Safari: If the screen is locked or the client test runner tab is not shown, when the system attempts to run the test suite, tests run significantly slower and may time out. For best performance, run client test runners for scheduled suites in a virtual machine (VM) environment in which the screen does not become locked or disabled.
- The browser must meet the criteria you specified on the Scheduled suite run record.
- A client test runner meeting the criteria you specified on the Scheduled suite run record must be available to run the test suite at the scheduled time. The system cannot automatically open a client test-runner session.

Javascript window command intercepts

The Client Test Runner captures window object commands including console.log, console.error, alert, confirm, and prompt, with default responses where necessary.

- Any script that calls **window.confirm** receives a boolean response of **true**.
- Any script that calls **window.prompt** receives the string response **test value**.
- Any script call to **alert** is ignored.

Active Test Runners table

When you start a client test runner, the system registers that runner in the Active Test Runners table. You can view this table in the Active Manual Test Runners module and the Active Scheduled Test Runners module. These two modules provide views of the same table, filtered to show only manual or only scheduled test runners.

The Active Scheduled Test Runner module is useful when you create a scheduled suite run. For scheduled suite runs, you can specify the browser to use. To determine the name and version of a browser you want to use, start a scheduled test runner with that browser, then inspect that runner's record in the Active Scheduled Test Runners module.

The data in this table is transient. While the runner is active, it reports in to the system at a specified interval. If the runner does not report in at the expected time, the system marks the runner as inactive. After a period of time the system deletes the runner. You can modify these intervals on the Automated Test Framework properties page.

Related topics

Client test runner

Scheduled client test runner

Allowed client errors

Pick a browser

If the test or test suite you are running contains steps that work with a form (any step involving a UI), or any other UI test step element (such as Automated Service Catalog test steps), work with the Pick a Browser dialog.

Before you begin

You must have created the test you want to run.

The test execution property must be enabled. You must have an admin or atf_test_admin role to do so.

Note:
The test execution property is disabled by default to prevent running tests on a production system. Run tests only on development, test, and other sub-production instances.

Role required: atf_test_admin, atf_test_designer, or admin

About this task

The Pick a Browser dialog appears after you click **Run Test** or **Run Test Suite**. The dialog asks you to choose among any currently-running test clients or start a new test runner.

Procedure

1. Choose the browser in which the test or test suite should run or be rerun again in the case of failed tests.
   All registered client test runners that are currently active appear in the Pick a Browser dialog. (Current session) indicates that the browser is currently running. See Working with client test runners.
2. Click **Manage your test runners here** as needed to view all client test runners registered for the current user.
   See Active manual test runners.
3. If you want to open a client test runner in this browser session, click **Start a new test runner**, which appears when client test runners are only available in other browsers for the current user.
4. Click the appropriate button to run the test.
   - If you are running a single test, click **Run Test**.
   - If you are running a test suite, click **Run Test Suite**.
   - If you are running failed tests again, click **Re-run failed tests**.

Result

The test, test suite or failed tests run in the selected browser or client test runner. The Progress viewer appears for monitoring of the progress of the test run.

Cloud Runner browser

If you are running a test or a test suite, select the Cloud Runner browser option to run your tests in a cloud browser.

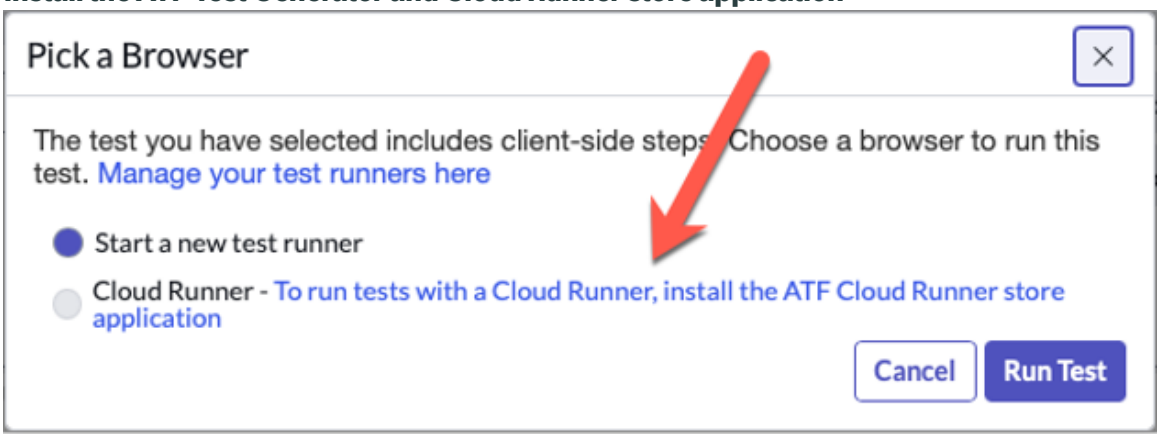
If you want to use the Cloud Runner browser option, you are required to complete the following:

- Install the ATF Test Generator and Cloud Runner store application.
- Configure a user to run and generate a test.

Use cases for Cloud Runner browser

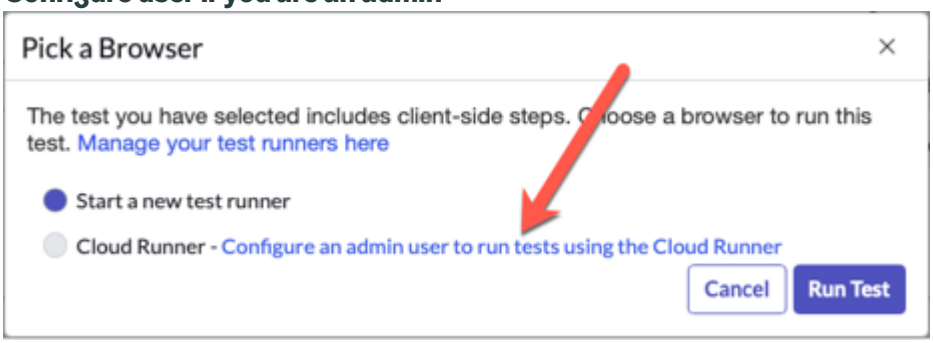
- When you haven't installed the ATF Test Generator and Cloud Runner store application

Install the ATF Test Generator and Cloud Runner store application



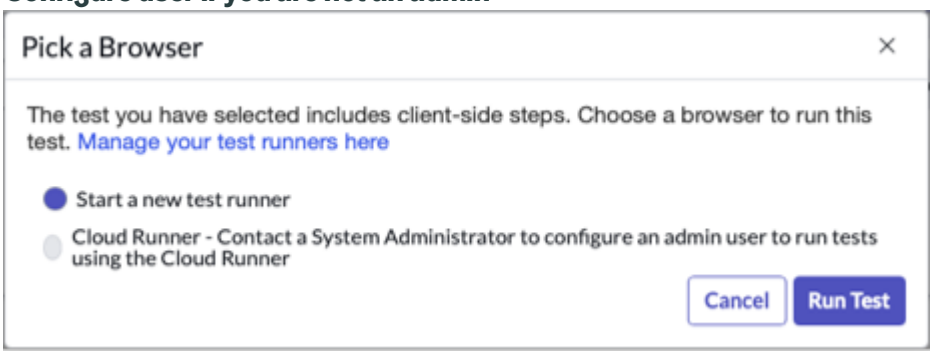
- When you haven't configured the user to run and generate tests
  - If you are an admin

Configure user if you are an admin



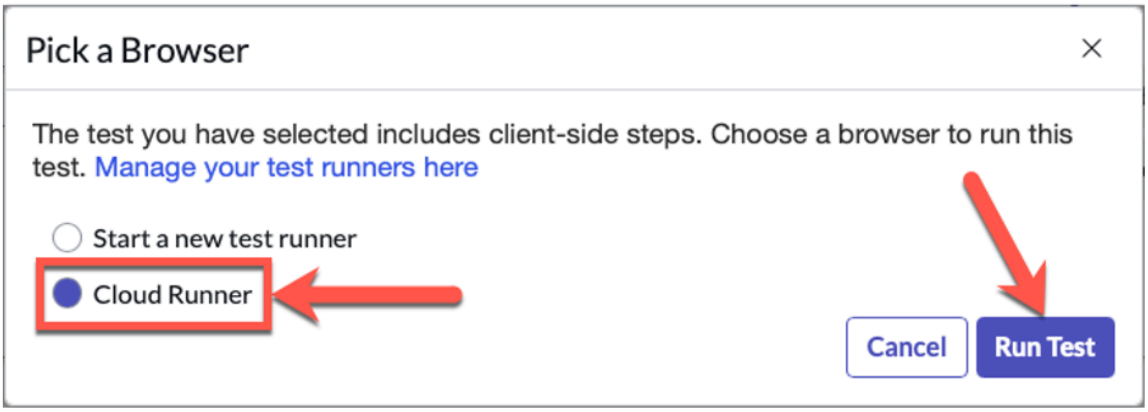
  - If you are not an admin

Configure user if you are not an admin



- When the application is installed and the user is configured

Use the cloud runner option



Server test steps

Test business logic and background processes by performing operations on the server.

Server test steps mimic non-interactive actions such as impersonating users, submitting or saving records, running server-side script, or making REST calls. Since server test steps run directly on the server, they do not require a client test runner. For more information about the server test steps, see Server category.

REST test steps

Test custom inbound web services and backwards compatibility by making REST calls.

REST requests can only be sent to the current instance. You cannot send a request to another instance or third-party at a remote address.

The REST test configuration only supports the XML and JSON response formats. Binary formats are not supported.

You can create tests that include steps from each of the test step configuration categories. The REST test configuration category contains the Send REST Request - Inbound and assert test configurations. Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories. For more information about the REST test steps, see REST category.

Authentication

As part of the Automated Test Framework, there are two situations when you send REST requests:

- When you use the REST API Explorer to create and test a request
- When you run a test that contains a **Send REST Request - Inbound** step

When you use the REST API Explorer to create and test a request, and the request requires authentication, the REST API Explorer uses your credentials. When the ATF runs the test, the REST API Explorer uses the credentials of the user who scheduled the test. This means that a test might fail unintentionally because of the difference in privileges between the user who created the test and the user who runs the test.

To address the issue of user credentials, you can create a basic-authentication profile for a test user and then on the **Send REST Request - Inbound** form, specify that the profile be used when the test is run.

Attachment test steps

Test an attachment-dependent business rule by uploading an attachment either from a form or from a server-side API call. For example, you can have a business rule that doesn't let you close an incident without an attachment such as a screenshot.

Upload from form

As a UI test step, the upload attachment step requires navigation to a form, which you can open using either **Open a New Form** or **Open an Existing Record**. Use **Upload Attachments** to select from the attachments that the test step adds to the form. When you select attachments to add to a form, the system waits to load the attachments before proceeding to the next test step. For more information on UI test dependency and wait mechanism, see UI test steps.

Upload from Server API

As a Server test step, the upload attachment step has no UI dependencies. Use **Upload Attachments** to select from the attachments that the test step adds to the record. When you select attachments to add to a form, the system waits for the attachments to be loaded before proceeding to the next test step. For more information, see Server test steps.

Design considerations

Follow these design considerations for attachment test steps:

- All attachment steps require adding one or more attachments.
- The system rolls back any attachments by the step after the test completes.
- The system cannot roll back any existing attachments after the test completes.
- Avoid testing records with existing attachments to eliminate data dependency.
- If UI testing is involved, add the attachment to a form.
- When no UI is involved, add the attachment to the Server API.

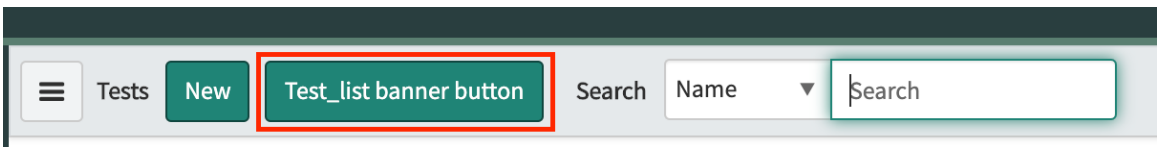
List UI actions test steps

Select a UI action from a list to perform different actions on a list or a related list.

You can create a new UI action of the following types. See Create a UI action for more information.

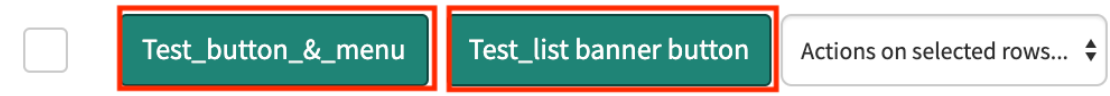
- **List banner button**: Creates a button on the banner of a list.

List banner button



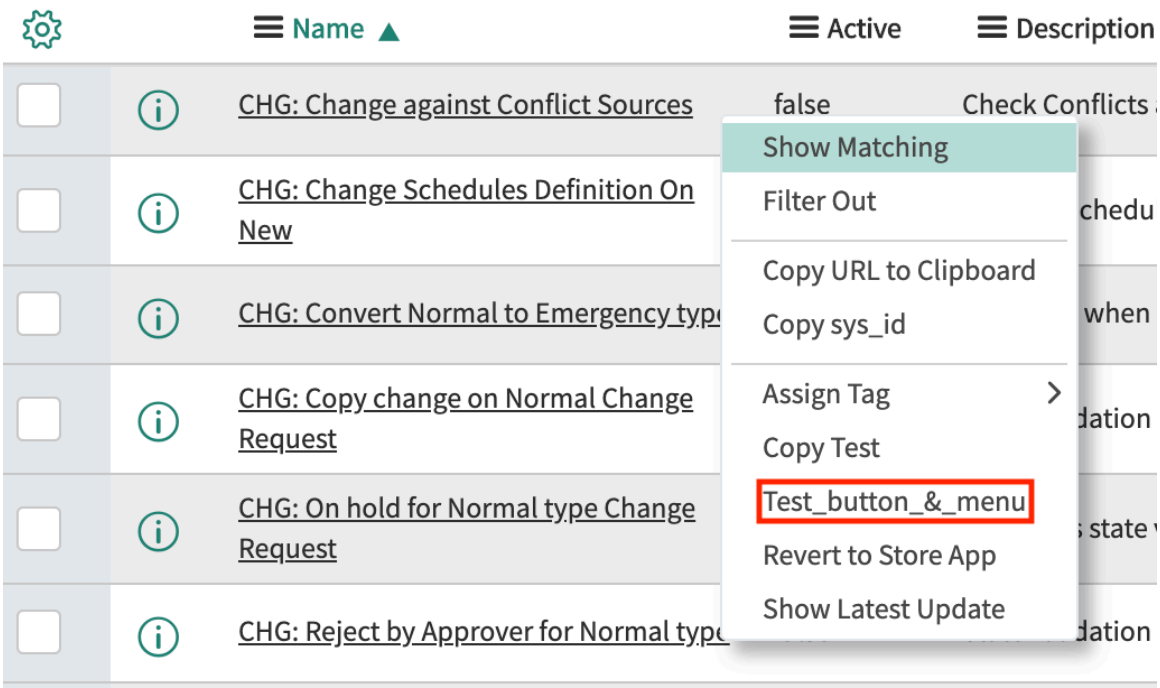
- **List bottom button**: Creates a button at the bottom of the list.

List bottom button



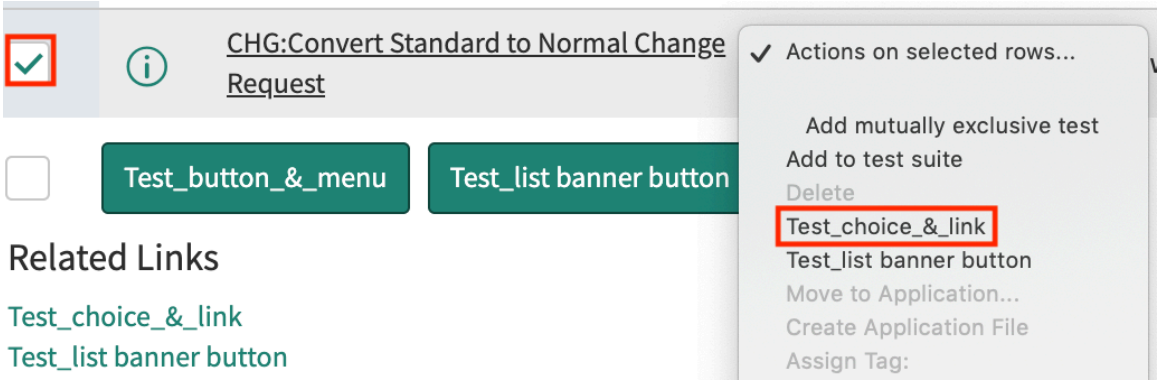
- **List context menu**: Adds an option to the context menu of the list.

List context menu



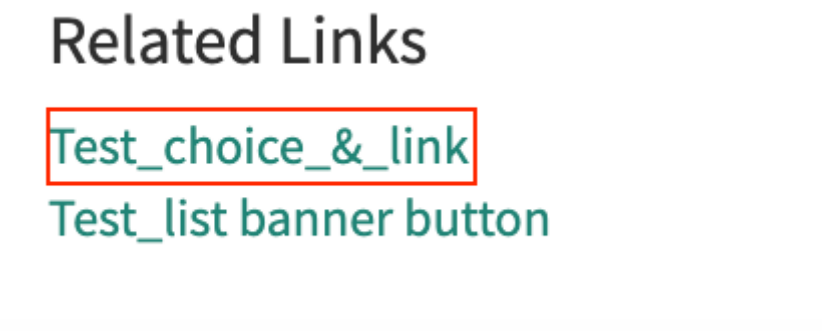
- **List choice**: Adds an option to the list choice at the bottom of the list. You need to select one or more tests to enable the recently added list choice.

List choice



- **List link**: Adds a link to the **Related Links** list.

List link



Design considerations

- To use the **Click a List UI Action** test step, the test first needs to navigate to the list or the form with the related list containing the UI action.
- The **Related list** field appears only when you select Related list as the **List type**.
- **Action type** is mostly auto-filtered depending on the **List action** chosen.
- The **Record** field appears only when you select **Single record** for the UI action to be applied.
- Identify the specific record if you have selected **Single record** to apply the UI action.
- For the **Timeout** field to appear, select **Page reloaded or redirected** as the **Assert** type.

Parameterized tests

Run a test multiple times with different test data for each run. Create parameters to store test data for each test run.

Parameterized testing offers test designers these benefits.

- Eliminates the need to duplicate test steps just to change test data.
- Increases test reuse by separating test actions from test data.
- Produces a separate test result for each data set.

When the test runs, Automated Test Framework replaces the parameters with data set values. For example, you can create a test of the incident form that uses parameter values for the subcategory and priority fields. You can use one data set to test that the Antivirus category produces a high priority incident, and another data set to test that the Email category produces a low priority incident.

Note:
The Run Server Side Script test step is not supported in parameterized tests.

Parameterized test components

Parameterized tests consist of these components.

Parameter

A parameter is a variable that stores a particular type of test data. Each parameter has a unique label and a data type. For example, you can create a parameter to

store the Sys ID of a reference field or the integer value of a choice field. Define parameters during test design.

Parameters can be shared or exclusive. Shared parameters can be used in any parameterized test. Exclusive parameters can only be used with the test for which they were created. Each shared parameter is a column in the Test Run Data Sets [sys_atf_parameter_set] table. Each exclusive parameter is a record in the Parameter Variables [sys_atf_parameter_variable] table.

Data set

A data set, also known as a test run data set, includes runtime data used when the test runs. You can set a value for every parameter available to the current test. Data sets specify the parameter value during test runs. You can manually create data sets for a test, or import data from a file. Each data set is a record in the Test Run Data Sets [sys_atf_parameter_set] table.

Parameterized tests fail if data sets are not defined.

Design Considerations

Follow these design considerations when creating parameterized tests.

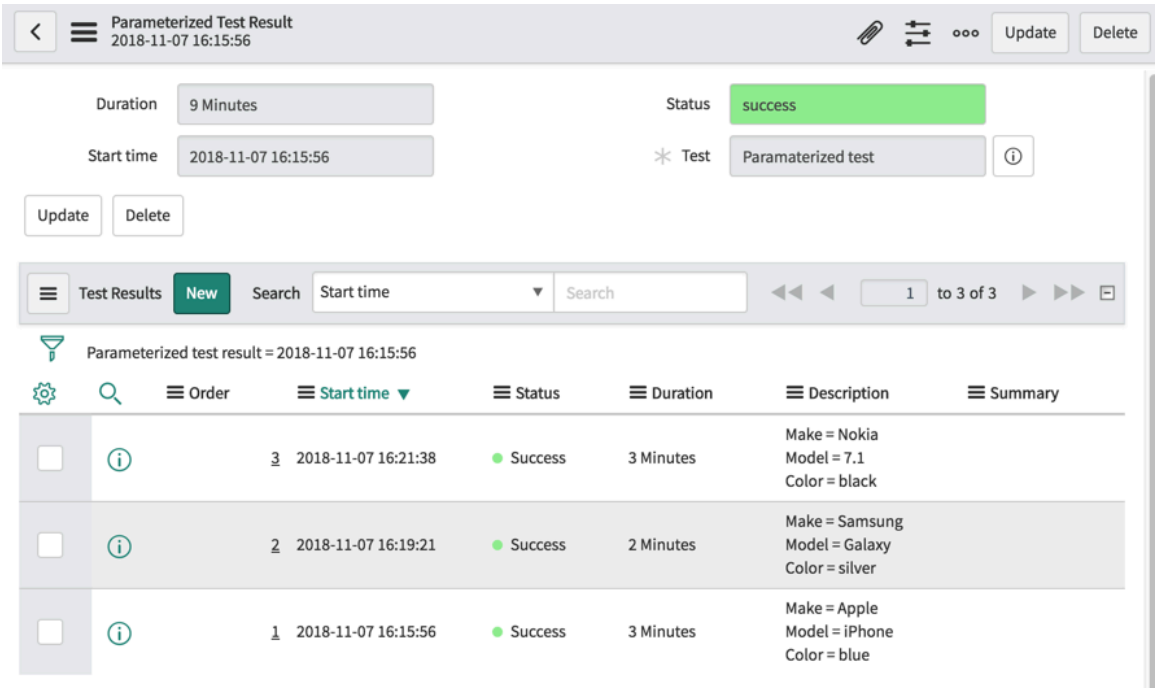
- Parameterized tests support standard Automated Test Framework features, such as reports, test suites, and data rollback. Copying a parameterized test copies all parameters, test run data sets, and test steps.
- If you create a parameterized test that includes Custom UI test steps, the system only uses the first data set to retrieve components.

Parameterized test runs

Automated Test Framework runs each parameterized test once per data set, using the same test steps and execution order. For example, if a parameterized test has five data sets, Automated Test Framework runs the test five times, once for each data set.

Parameterized test results

Parameterized tests display test results by the execution order of the data sets. Open each test result record to view the test details.



The Parameterized Test Result record **Description** field lists the parameters and data sets used in the test run.

Create a parameterized test

Build a test that uses variables to store test data.

Before you begin

Role required: atf_test_admin, atf_test_designer, or admin

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Click **New**.
   A blank Test record opens.
3. On the form, fill in the fields.

Field	Description
Name	Enter a name for the test.
Active	Enable
Enable parameterized testing	Enable
Description	Enter a description for the test.

4. Click **Save.**
   Parameterized testing related lists appear.
5. Create a parameter to hold test run data.
   A parameter is a variable that stores a particular type of test data. Each parameter has a unique label and a data type. For example, you can create a parameter to store the Sys ID of a reference field or the integer value of a choice field.

a. In the Parameter Definitions related list, add a parameter.
   - **Add Exclusive Parameters**: Adds a parameter available to this test only.
   - **Add Shared Parameters**: Adds a parameter available to any parameterized test.

b. Define the name of the variable and the data type.
   When creating parameters for a form, the parameter data type must match the field data type. For example, if you are creating a parameter to test a reference field on a form, you must create a parameter of type Reference. For more information on ServiceNow AI Platform data types, see Field types.

c. Click **Submit**.

What to do next
Add a parameter to a test step

Add a parameter to a test step

Add a variable to a test step to hold a particular type of data when the test runs.

Before you begin

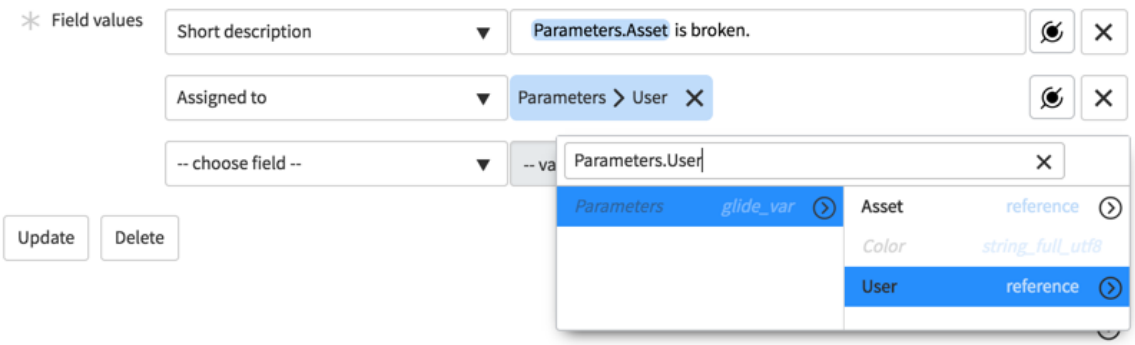
- Create a parameterized test
- Role required: atf_test_admin, atf_test_designer, or admin

About this task

Note:
The Run Server Side Script test step is not supported in parameterized tests.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Open a parameterized test.
3. In the Test Steps related list, create a test step or open an existing step.
4. Click and select a parameter to add it to a field.
   The parameter displays in the field.



Result
When the test runs, Automated Test Framework replaces the parameter with test run data.

What to do next
Add parameterized data sets.

Add parameterized data sets

Add or import test data to specify parameter runtime values.

Before you begin

- Create a parameterized test
- Add a parameter to a test step
- Role required: atf_test_admin, atf_test_designer, or admin

About this task

A data set, also known as a test run data set, includes runtime data used when the test runs. You can set a value for every parameter available to the current test. Data sets specify the parameter value during test runs.

Test designers can add data sets manually, or import data from a file.

Note:
If you create a parameterized test that includes Custom UI test steps, the system only uses the first data set to retrieve components.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Open a parameterized test.
3. In the Test Run Data Sets related list, add or import data sets.

What to do next

Click **Run Test**. Automated Test Framework runs each parameterized test once per data set, using the same test steps and execution order. For example, if a parameterized test has five data sets, Automated Test Framework runs the test five times, once for each data set.

Parameterized tests display test results by the execution order of the data sets. Open each test result record to view the test details.

Allowed client errors

Add known client errors to the allowed client errors list to allow tests and steps to continue running when a specific error occurs. Set the report level to specify what the Automated Test Framework does when the error occurs in future tests.

Scenarios for allowing client errors

Test designers and developers typically allow client-side JavaScript errors to prevent certain types of known failures from impacting test design and results. Scenarios to allow client errors include:

Timing constraints

Temporarily allow a client error until your developers have time to investigate and resolve the issue. For example, when testing an old form containing a longstanding bug.

Minimizing the impact of old libraries

Ignore client errors that cannot be fixed or are unimportant to your operations to eliminate their impact on future test runs. For example, when you find a bug in an old library.

Test design time

Temporarily allow client errors until you finish writing tests and have time to investigate the error. For example, one of your developers modifies a UI policy and the change generates an error.

Possible platform bug

Temporarily allow client errors until a fix is available. Prior to reporting a platform bug to ServiceNow Technical Support, investigate the error, verify it is not a customization error, and identify the type of platform bug involved. For example, a UI policy generates an error during a test. Your investigation verifies that the issue is not a customization error and identifies a platform issue with the UI policy.

Note:
While adding client errors to the allowed client errors list allows the test framework to continue testing, it does not guarantee that your tests pass. Test designers and developers should always investigate client errors to determine if there are issues with your business process. For more details, see Identify and resolve client errors.

Report levels for allowed client errors

The report level indicates whether the test framework reports future occurrences of the error as a warning or ignores them altogether. You can change the report level of an allowed error at any time. For example, if you originally add an error as a warning, you can later change the report level to ignored.

Report level	Description
Warning	Test steps containing the allowed client error report a status of `Success with warning(s)`. The error message appears in the test result output, and is recorded in the test logs with the status `Warning`.
Ignored	Test steps containing the allowed client error report a status of `Success`. The error is recorded in the test logs with an `Ignored` status.

Matching process

The Automated Test Framework identifies allowed client errors using a contains search rather than an exact string match. A match occurs when a client error contains a message from an Allowed Client Error [sys_atf_whitelist] record. For example, if you create an Allowed Client Error record for the error message "`Test message`" with a report level of Ignored, then any client error containing this string is ignored.

Note:
When you create or modify an Allowed Client Error record, the Client Test Runner automatically gets the update.

Identifying and resolving client errors

When client errors occur, the Automated Test Framework fails the test on the step that was executing when the error occurred.

Allow client errors from test results

Allow client errors as you review test results.

Before you begin
Role required: atf_test_admin, atf_test_designer, or admin

About this task
You can allow multiple or individual client errors. For each client error, you must decide how to report future instances of the client error. Report level options include:

- Warning: Test steps containing the allowed client error report a status of `Success with warning(s)`. The error message appears in the test result output, and is recorded in the test logs with the status `Warning`.
- Ignored: Test steps containing the allowed client error report a status of `Success`. The error is recorded in the test logs with an `Ignored` status.

Procedure
1. Navigate to **All** > **Automated Test Framework** > **Test Results**.
2. Select the test result for a specific test.
   The system displays the Test Result record.
3. From the Related links, select one of the following options:
   - **Add all client errors to warning list**: Allow all client errors in this test with a report level of Warning.
   - **Add all client errors to ignored list**: Allow all client errors in this test with a report level of Ignored.

   The Automated Test Framework allows the selected client errors and displays a status message at the top of the form.

Related topics
Allowed client errors

Allow client errors from step results
Allow client errors as you review step results.

Before you begin
Role required: atf_test_admin, atf_test_designer, or admin

About this task
You can allow multiple or individual client errors. For each client error, you must decide how to report future instances of the client error. Report level options include:

- Warning: Test steps containing the allowed client error report a status of `Success with warning(s)`. The error message appears in the test result output, and is recorded in the test logs with the status `Warning`.
- Ignored: Test steps containing the allowed client error report a status of `Success`. The error is recorded in the test logs with an `Ignored` status.

Procedure
1. Navigate to **All** > **Automated Test Framework** > **Test Results**.
2. Select the test result for a specific test.
   The system displays the Test Result record.
3. Select the client errors to be allowed.

Option	Description
Specific step result from Step Results related list	**a.** In the Step Results related list, right-click the step result containing client errors you want to allow. **b.** From the context menu, select one of the following options: ▪ **Add all client errors to warning list**: Allow all client errors in this step with a report level of Warning. ▪ **Add all client errors to ignored list**: Allow all client errors in this step with a report level of Ignored.
Multiple step results from Step Results related list	**a.** In the Step Results related list, select the check box in the first column for each step result containing client errors you want to allow. **b.** From the **Actions on selected rows** list, select one of the following options. ▪ **Add all client errors to warning list**: Allow all client errors in this step with a report level of Warning. ▪ **Add all client errors to ignored list**: Allow all client errors in this step with a report level of Ignored.
Specific step result from Step Result record	**a.** In the Step Results related list, select the Step Result record containing a client error you want to allow. **b.** From the Related links, select one of the following options: ▪ **Add all client errors to warning list**: Allow all client errors in this step with a report level of Warning. ▪ **Add all client errors to ignored list**: Allow all client errors in this step with a report level of Ignored.

The Automated Test Framework allows the selected client errors and displays a status message at the top of the form.

Related topics

Allowed client errors

Allow client errors from the test logs

Allow client errors as you review test logs.

Before you begin

Role required: atf_test_admin, atf_test_designer, or admin

About this task

You can allow multiple or individual client errors. For each client error, you must decide how to report future instances of the client error. Report level options include:

- Warning: Test steps containing the allowed client error report a status of `Success with warning(s)`. The error message appears in the test result output, and is recorded in the test logs with the status `Warning`.
- Ignored: Test steps containing the allowed client error report a status of `Success`. The error is recorded in the test logs with an `Ignored` status.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Test Results**.
2. Select the test result for a specific test.
   The system displays the Test Result record.
3. Select the client errors to be allowed.

Option	Description
Specific test log from the Test Log related list	**a.** In the Test Log related list, right-click the test log client error you want to allow. **b.** From the context menu, select one of the following options: ▪ **Add client error to ignored list**: Allow the client error with a report level of Ignored. ▪ **Add client error to warning list**: Allow the client error with a report level of Warning.
Multiple test logs from the Test Log related list	**a.** In the Test Log related list, select the check box in the first column for each test log client error you want to allow. **b.** From the **Actions on selected rows** list, select one of the following options: ▪ **Add client error to ignored list**: Allow the client error with a report level of Ignored. ▪ **Add client error to warning list**: Allow the client error with a report level of Warning.
Specific test log from Test Result Item record	**a.** In the Test Log related list, select the test log (Test Result Item record) containing a client error you want to allow. **b.** From the Related links, select one of the following options: ▪ **Add all client errors to warning list**: Allow all client errors in this step with a report level of Warning. ▪ **Add all client errors to ignored list**: Allow all client errors in this step with a report level of Ignored.

The Automated Test Framework allows the selected client errors and displays a status message at the top of the form.

Related topics

Allowed client errors

Manually allow client errors

Manually create allowed client error entries as needed in the Allowed Client Errors table.

Before you begin

Role required: atf_admin, atf_test_designer, or admin

About this task

You normally allow client errors directly from a test result, step result, or test log, but you can manually allow them.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Run** > **Allowed Client Errors**.
2. Click **New**.
3. On the form, fill in the fields.

Allowed Client Error form

Field	Description
Report level	Report action to take when the client error is encountered. Options include: ○ **Warning – Step & Test will report Success with Warning(s)**: Test steps containing the allowed client error report a status of `Success with warning(s)`. The error message appears in the test result output, and is recorded in the test logs with the status `Warning`. ○ **Ignored – Step & Test will report Success**: Test steps containing the allowed client error report a status of `Success`. The error is recorded in the test logs with an `Ignored` status.
Active	Check box to enable or disable allowing a client error.
Error message	Message of the client error you want to allow.
Description	Description of the error you want to allow. If this client error was allowed from a test result, step result, or a test log, the test log description is copied into this field.

4. Click **Submit**.

Related topics

Allowed client errors

View the progress of automated tests

When an automated test is running, view its progress in the Run Test progress dialog.

Before you begin

The system must be running a test.
Role required: atf_test_admin or atf_test_designer

About this task

When you execute a test or test suite, the system automatically displays the Run Test progress dialog. If you close this dialog, you can re-display it from either the results list or the results page for the currently-running test or test suite.

Procedure

1. If necessary, navigate to the Test Results list or Test Results page for the currently-running test or test suite.
   - If the system is currently running a test suite, navigate to **Automated Test Framework** > **Suite Results**. If desired, you can click the row for the running test suite to view the Suite Results page for that test suite.
   - If the system is currently running a test that's not part of a test suite, navigate to **Automated Test Framework** > **Test Results**. If desired, you can click the row for the running test to view the Test Results page for that test.

2. Display the Run Test dialog.
   - If you are currently viewing the Test Results page or the Suite Results page, click **Show Progress** under Related Links.
   - If you are currently viewing the Test Results list or Suite Results list, right-click the row for the running test or test suite, then click **Show Progress**.

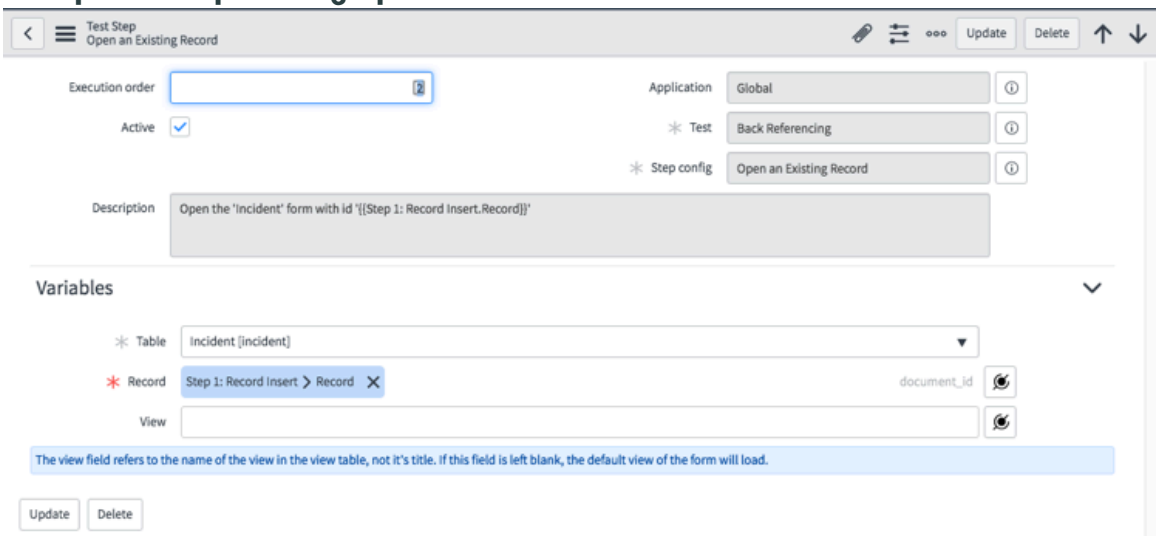
Passing data from one automated test step to another

Some automated test steps create data that you can use as an input to a subsequent step.

You can pass data from one test step to another using input variables and output variables.

The term input variables is another name for the field values associated with a step. These values are input variables because they provide the input the step needs to accomplish its task. For example, the Open Form step has these input variables: **Table**, **Record**, and **View**.

Example test step showing input variables

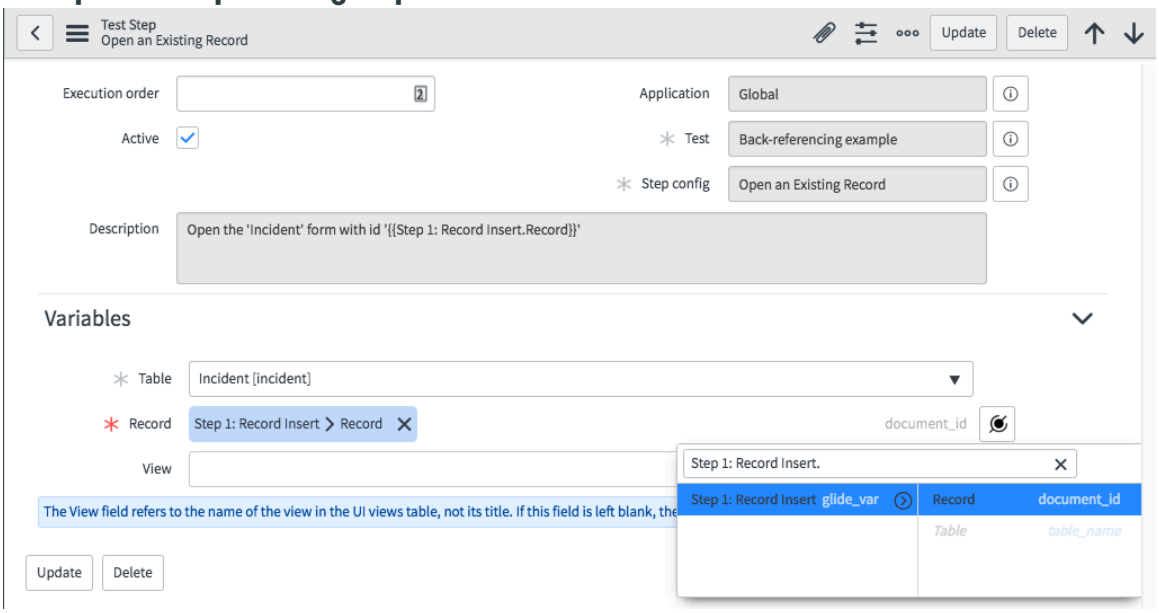


Some step types also have output variables. These are the values that later steps in the same test can use as input. For example, the Record Insert step has an output variable called Record which contains the sys_id of the newly created record.

Note:
Input data can also be passed to an automated test case from an external source such as a .csv file.

The test step form doesn't indicate if a test step has output variables or not. You can easily tell if any output variables are available to provide a value to any given input variable. If you can map the value of one step's output variable to the current step's input variable, the system displays the mapping icon ( ) to the right of that input field. When you click the variable mapping icon, the system displays a tree giving you access to any available output variables from previous steps.

Example test step showing output variables



For step-by-step instructions on how to assign the value of an output variable to another step's input variable, see Pass values from one automated test step to another.

For an example of a test that passes variables using input and output variables, see Automated Test Framework use case: reference a value from a previous step.

Pass values from one automated test step to another

Assign a form field the value of an output variable returned from a previous step.

Before you begin

You must have a previous test step that returns an appropriate output variable.

Role required: admin, atf_test_admin and atf_test_designer

Procedure

1. To the right of the field whose value you want to assign, click the input value icon ( ).
   The input value mapping control lists previous steps that create an output variable. If no previous steps create an output variable, the control displays the message: There are no elements to show .
2. Click the row for the step that contains the output variable you want to use as an input.
3. Click the output variable you want to use.
   If the output variable is an id for a glide record, the control displays a tree picker providing access to fields for this record.
4. Navigate through the tree picker hierarchy until you find and select the value you want.

Related topics

Automated Test Framework use case: reference a value from a previous step

Building and running automated test suites

Run a group of tests in a specific order to test an application or a group of related features.

A test suite can contain both individual tests and other test suites. A test suite that contains another test suite is called a parent, and the test suite contained within the parent is called a

child. While a test suite can have both individual tests and test suites as children, tests cannot have other tests as children. Tests can only contain test steps.

Benefits

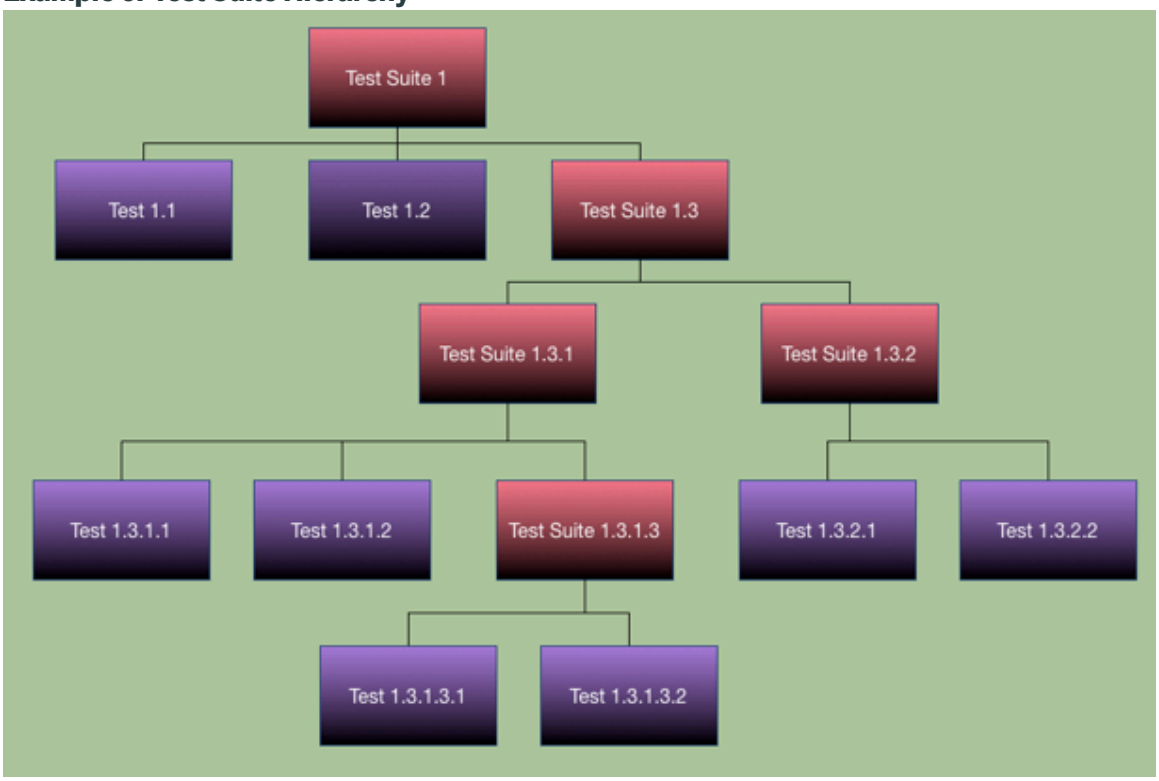
Grouping tests into test suites offers these benefits.

- Allows testers to run every test in a test suite with one action.
- Allows testers to run all child test suites in a parent test suite.
- Allows testers to see test results for every test in a test suite.
- Allows test designers and testers to schedule when to run test suites.
- Allows test designers and testers to schedule starting client test runners to support test runs.

Hierarchies

Automated Test Framework supports building a multi-level hierarchies where a test suite can be both a parent and child. For example, this figure illustrates Test Suite 1 as the parent at the top of the hierarchy. Test Suite 1.3 is a child of Test Suite 1 and also a parent of Test Suite 1.3.1 and Test Suite 1.3.2.

Example of Test Suite Hierarchy



Schedules

To schedule a test suite, you need three components:

- a test suite record
- a schedule record specifying when you want the system to run the test suite
- a scheduled suite run record that associates the test suite to run with the schedule for running it

With this model, you can associate a schedule with many different test suites, and vice versa.

Note:
You can schedule only test suites, not individual tests. Scheduled tests will run only if there is an open Scheduled Client Test Runner page matching the scheduled suite's browser conditions. Scheduled tests cannot run on a machine that is locked, powered down, or does not already have the browser open.

The watchlist on the test suite run record also allows you to specify users to receive an email when the system finishes executing the test suite run.

If the test suite contains one or more form steps (steps involving a user interface), you must ensure that a scheduled client test runner is actively running in a browser when the schedule triggers the suite run.

Note:
See Browser recommendations and requirements for recommendations and requirements for running the client test runner.

For step-by-step instructions on how to schedule a test suite, see Schedule an automated test suite.

Filters

Automate the creation of test suites by using a filter to dynamically add tests to a test suite when they match the filter conditions. Reduce the time that your test designers spend manually creating and maintaining test suites.

Related topics

Suites

Step results record

Create an automated test suite

Group automated tests into a suite you can execute as a batch.

Before you begin

The tests you want to include in the test suite must exist.
Role required: atf_test_admin or atf_test_designer

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Suites**.
2. Click **New**.
   The system displays the Test Suite New Record form.
3. In the **Name** field, enter a name for this suite.
4. Select the tests to be included in this suite.

Option	Description
Filter	Add tests dynamically using a filter: **a.** In the **Filter** field, use the condition builder to create the conditions a test must match for inclusion in the test suite. **b.** Click **Save**.

Option	Description
	All tests that match the filter conditions appear in the **Test Suite Tests** related list. Because the suite is dynamic, any new test that matches the conditions is automatically added to the suite.
Test Suite Tests	Add tests manually: **a.** In the **Test Suite Tests** related list, click **Insert a new row...**. **b.** In the **Test** field, enter the name of the test to add to this test suite.

5. In the **Description** field, enter a description for this test suite.
6. In the **Test Suite Tests** related list, specify options for a test.

Option	Description
Execution order	Enter a value to specify the order in which this test should execute within the test suite.
Abort on Failure	Enter a value that indicates whether you want the test suite to stop or continue if this test fails. ○ By default, the system assigns the value **false** to this field. If this test fails, the system still executes any further tests in the test suite. ○ Set the value to **true** to stop the test suite if this test fails.

7. Repeat steps 4 - 6 for every test you want to include in this test suite.
8. Click **Submit**.

Copy an automated test suite

Reduce time when creating tests by copying an entire test suite. Rename and modify the test suite after copying. The **Copy Test Suite** button on the Test Suite form copies all the nested tests and child test suites within the text suite.

Before you begin

Role required: atf_test_admin or atf_test_designer

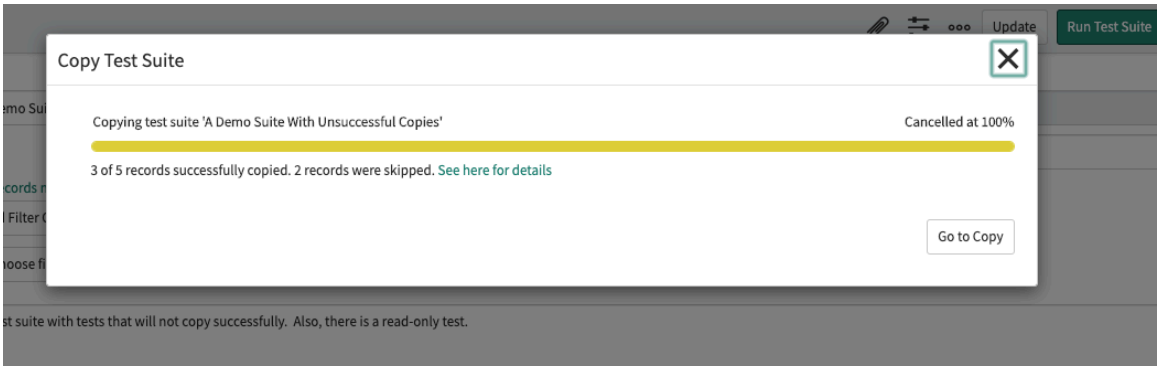
Procedure

1. Navigate to **All** > **Automated Test Framework** > **Suites**.
2. Select the row containing the test suite that you want to copy.
3. Near the top corner of the Test Suite form, select **Copy Test Suite**.
   A progress tracker appears to confirm the status. If you select **Go to Copy**, the system displays a new test suite record identical to the copied record, except for the **Name**.

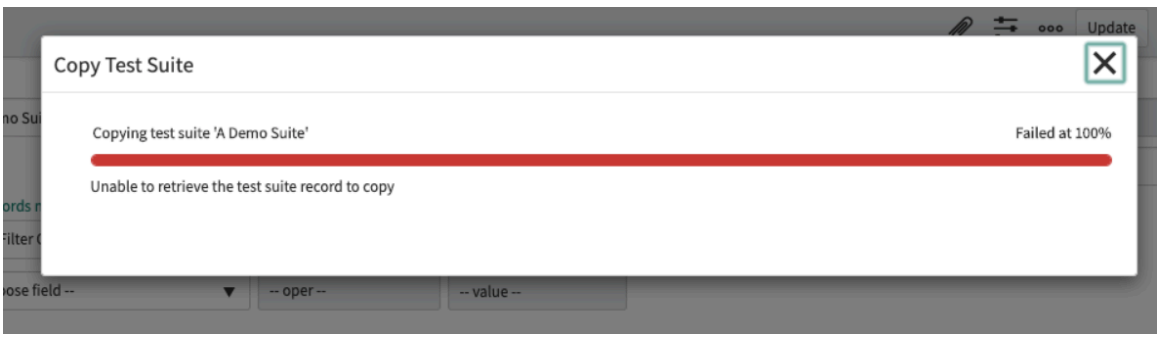
   Note:
   If you apply filter conditions, copied tests are added to the test suite if the conditions are met.

   If any tests or child test suites fail to copy, they are skipped and copying of the remaining tests and child test suites continues. A warning message showing a partial success appears below the progress bar.

Note:
You can copy a test suite only when it's in the same scope as the current scope. The **Copy Test Suite** button appears only when the test suite is in the same scope. The scopes of the tests in a suite are preserved when copying a test suite. See Application Scope for more information.



If the copy operation fails, an error message appears below the progress bar.



4. In the **Name** field, enter the name that you want to assign to this new test suite.
5. Edit the tests and test suites within the copied test suite.
6. When you are finished making changes, select **Update**.

Result
A copy of the test suite along with all the nested tests and child test suites is created.

Add tests to a suite with a filter

Automate the creation of test suites by using a filter to dynamically add tests to a test suite when they match the filter conditions. Reduce the time that your test designers spend manually creating and maintaining test suites.

Before you begin
The tests you want to include in the test suite must exist.
Role required: atf_test_admin or atf_test_designer

Procedure
1. Navigate to **All** > **Automated Test Framework** > **Suites**.
2. Click **New**.
   The system displays the Test Suite New Record form.
3. In the **Name** field, enter a name for this suite.

4. For **Filter**, use the condition builder to specify the conditions a test must meet to be added to the test suite.

   Note:
   If you don't have a filter condition set, it is ignored and you can build the suite manually.

5. Click **Save**.

Result
All tests that match the filter conditions appear in the **Test Suite Tests** related list. Because the suite is dynamic, any new test that matches the conditions is automatically added to the suite.

Add test to an existing automated test suite

Add a test to a test suite that already exists.

Before you begin
The tests you want to include and the test suite must exist.
Role required: atf_test_admin or atf_test_designer

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Suites**.
2. Click the row containing the test suite you want.
   The system displays the Test Suite form.
3. In the **Test Suite Tests** related list, click **Insert a new row...**.
4. In the **Test** field, enter the name of the test to add to this test suite.
5. In the **Order** field for this row, enter a value to determine the order in which this test should execute within the test suite.
   By default, the system assigns a value to this field according to the order in which you add the tests.
6. In the **Abort on Failure** field for this row, enter a value that indicates whether you want the test suite to stop or continue if this individual test fails.
   By default, the system assigns the value **false** to this field. **False** means that if this test fails, the system still executes any further tests in the test suite.
7. Repeat steps 3 - 6 for every test you want to include in this test suite.
8. Click **Submit**.

Add child test suite to parent test suite

Add to a multi-level test suite by including a child test suite within a parent test suite.

Before you begin
The parent test suite must exist.
Role required: atf_test_admin or atf_test_designer

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Suites**.
2. Access a child test suite.
   - If the child test suite exists, open the child test suite form for editing.
   - If the child test suite does not yet exist, create it.

3. In the test suite form in the **Parent suite** field, enter the name of the test suite you want to act as the parent to this child.
4. If desired, add one or more tests to the child test suite.
5. Click **Submit**.

Run an automated test suite

After creating an automated test suite, run it in a non-production instance.

Before you begin

You must have created the test suite you want to run.
The test execution property must be enabled. You must have an admin or atf_test_admin role to do so.

> **Note:**
> The test execution property is disabled by default to prevent running tests on a production system. Run tests only on development, test, and other sub-production instances.

Role required: atf_test_admin, atf_test_designer, or admin

About this task

This procedure outlines how to start a test suite manually. You can also schedule test suites to run at a later time. For more information, see Working with scheduled test suites.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Suites**.
2. If necessary to view the Test Suites list, click **Test Suites**.
3. Click the row containing the test suite you want to run.
   The system displays the Test Suite form.
4. Click **Run Test Suite**.

   > **Note:**
   > If the test execution property is not enabled, the **Run Suite** button does not appear. In this case, see the annotation at the top of the form, and click the link to enable running tests.

   If the tests associated with the test suite include a form step (any step involving a UI), or other kinds of UI test steps, the Pick a Browser dialog appears before executing the tests.
5. Choose among any currently running test clients, or start a new runner.

   For more information, review Browser recommendations for all tests and suites.

   If the tests associated with the test suite only include server test steps, the system executes the tests without displaying the Pick a Browser dialog.

What to do next

Monitor the progress of the tests. When complete, click **Go to results** on the progress dialog window to display the **Test Results** list, where you can view and analyze the results.

Schedule an automated test suite

Schedule one or more test suites to run at a specific date and time.

Before you begin

You must have created the test suites you want to schedule.
Role required: atf_test_admin or atf_test_designer

About this task

To schedule a test suite, you need three components:

- A test suite record
- A schedule record specifying when you want the system to run the test suite
- A scheduled suite run record that associates the test suite to run with the schedule for running it

For more information about the capabilities of and requirements for scheduled test suites, see Working with scheduled test suites.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Schedules**.
   The system displays the list of existing test suite schedules.
2. To create a new schedule, click **New**.
   The system displays the Suite Schedule record form.
3. On the Suite Schedule record form, enter the name of the schedule, the frequency with which to run associated suites, the time at which to run associated suites, and the timezone for this schedule.
4. **Optional:** Specify a condition that must be met for running associated test suites by selecting **Conditional**, then fill in the **Condition** text box with the appropriate script.
5. Add a test suite to run by navigating to the Scheduled Suites related list, then clicking **New**.
   The system displays the Scheduled Suite Run record form.
6. On the Scheduled Suite Run record form enter the appropriate data.

   a. Enter the test suite to run.

   b. If the suite contains UI steps, enter any client constraints you wish to apply (such as browser to use).
   For more information on client constraints, see Scheduled Suite Run.

   c. Add to the record's watchlist users you want the system to inform (by email) when the scheduled suite has finished.

   d. Click **Submit**.

7. **Optional:** Add more test suites to this schedule by repeating steps 5 and 6.
8. Click **Update**.
9. If your suite includes any test steps that work with a form – or any other element on the client side, open a browser window for running the client portion of the scheduled tests.
   a. Review Browser recommendations and requirements for all tests and suites, as well as those that apply only to scheduled suite runs.
   b. In the **Navigator**, right-click **Scheduled Client Test Runner**, and then click the option to open in a separate tab or window, as you prefer.
   c. Leave open the browser window that's running the client test runner and return to the browser window that contains the **Navigator**.

Run a scheduled test suite using a script

Execute a scheduled UI test suite immediately using a script without having to wait for the scheduled time. You can use this method while trying to automate the process of running a test.

Before you begin

You've created and scheduled the test suites that you want to run. See Create an automated test suite and Schedule an automated test suite, for more information.

Role required: atf_test_admin or atf_test_designer

About this task

The following steps might not be in line with your software configurations.

Procedure

1. Spin up a virtual machine (VM) on an operating system with the necessary browsers.
2. Open a browser on the instance and navigate to the Scheduled Client Test Runner.
3. Call the scriptable method `new sn_atf.ScheduledRunsExecutor().setScheduleSysId("SYS_ATF_SCHEDULE_SYS_ID_HERE").start();` to run the scheduled suite immediately.
   To run a test only when the script is called, set the **Run** field to **On Demand** in the Schedule form.

   Note:
   The `start()` method returns `sys_progress_worker.sys_id` of the progress worker.

Re-run failed tests in an automated test suite

Re-run failed tests within a test suite without rerunning the entire suite.

Before you begin

Role required: atf_test_admin, atf_test_designer, or admin

About this task

The **Re-run failed tests** button appears on the Suite Result form and on the Suite Execution Progress Viewer after a suite with failed tests completes. It does not appear if test execution is disabled, the suite is deactivated, the suite passed, or the user does not have one of the required roles.

The **Re-run failed tests** button re-runs all non-passing tests, which includes tests with the following status: canceled, skipped, failure, and error. It does not include the test results with a Success with warning(s) status.

Procedure

1. Navigate to either the suite result form or suite execution progress viewer for the completed suite that had failed tests.
2. Click **Re-run failed tests**.
   If the re-run tests include a form step (any step involving a UI) or other kinds of UI test steps, the Pick a Browser dialog appears before executing the tests.
3. In the dialog box, choose among any currently running test clients or start a new runner.
   If the re-run tests include only server test steps, the system executes the tests without displaying the Pick a Browser dialog. For more information, review Browser recommendations for all tests and suites and Server test steps.

Result
The system re-runs the failed tests.

- The system creates a new suite result hierarchy for the re-run tests. The Progress Workers, Test Result, and Test Suite Result forms show the same suite hierarchy as the previous test suite. They do not include the tests or suites that passed in a previous run.
- If you delete or deactivate a child suite or test that failed, and then re-run it, the system does not execute that suite or test in the re-run.
- If you add a child suite or test to the suite to a failed test, and then re-run it, the system does not execute the added suite or test in the re-run.

What to do next
To view the results from the previous run of a test or suite, click **Previous test result** on the test result form or **Previous suite result** on the testsuite result form. These fields only appear for tests and suites that have been re-run.

Auto-generate ATF tests

Auto-generate ATF tests by selecting the auto-generate option either from the Auto-generate Tests module or Tests/Suites modules.

Before you begin
Role required: admin

Procedure
1. Access the Auto-generate Tests feature using either of the following ways.

2. **Optional:** Install the ATF Test Generator and Cloud Runner store app.

Note:
This step is applicable only if the store app is not installed on your instance. You will be redirected to install the store app.

Parallel testing

Reduce test design time by running multiple tests and test suites in parallel. Design tests to run in parallel by avoiding resource conflicts and data dependencies.

Note:
If two or more users are developing tests simultaneously, parallel testing reduces test design time. After test design completes, it is recommended to organize tests into a single hierarchical suite structure, and run the tests as a single base suite.

Parallel testing limit

Parallel testing enables users to run multiple automated tests simultaneously. This process continues until the number of parallel running tests is as per the following formula.

```
Number of parallel tests = max(1, number of worker threads - 2)
```

Note:
If your instance has 2 or less worker threads, configuration improvements review is recommended.

The actual number of parallel tests that a non-production instance can support depends on the system resources that the instance has when it is provisioned. The parallel testing limit ensures that an instance always has system resources available for other non-testing tasks.

Test waiting queue

When the system reaches the parallel testing limit, it reschedules tests to run later. It automatically places the tests back in sys_trigger until a worker thread is available to pick them up. Each test in the waiting queue has a schedule the next time the test runs.

Design considerations

Run multiple tests and test suites in parallel to reduce test design time. Avoid resource conflicts and data dependencies by designing parallel running tests. Avoid resource conflicts and data dependencies by designing tests that rely only on newly generated or self-created data, or have mutual exclusion rules defined between tests that share resources.

Prevent resource conflicts between parallel tests

Prevent resource conflicts by running tests that create their own data. Tests that run with existing data prevent other tests that need the same data from running in parallel.

Note:
If you have two or more tests with resource conflicts, see Mark tests as mutually exclusive to create a mutual exclusion rule that prevents the tests from running in parallel.

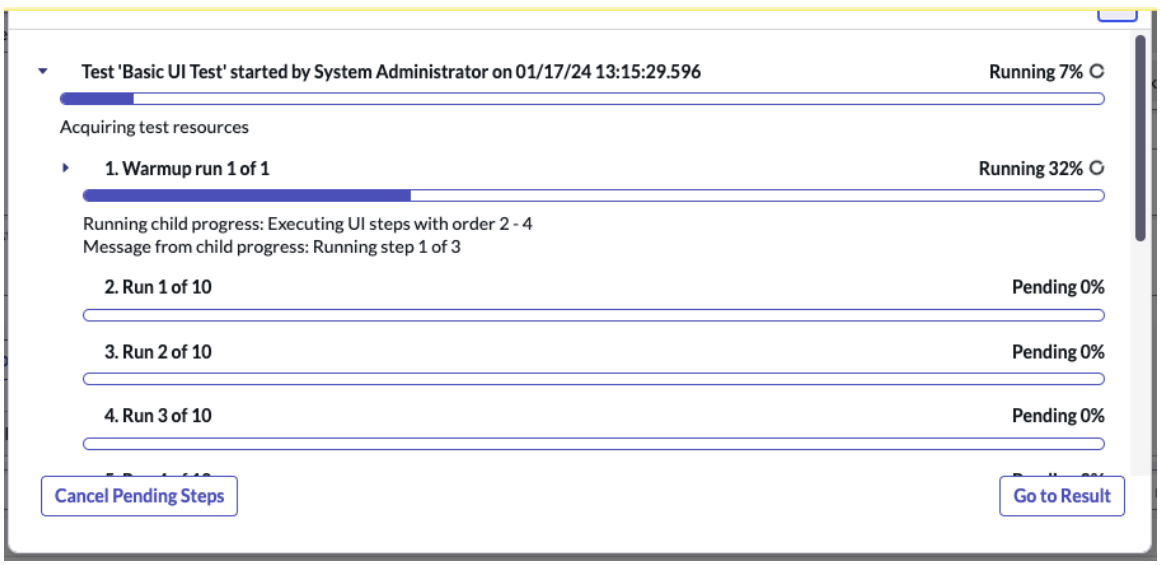
Performance profiling

Performance profiling allows you to do performance testing on your instances.

Apart from being a functional testing medium that ensures nothing breaks when any changes are made to an instance, ATF can also detect performance degradation during upgrades. You can know the cause, investigate, and fix the performance issue.

You can execute performance profiling on any of your ATF tests or suites. For each test or suite, the default test run is 10. In a suite, each test runs 10 times sequentially. The first run is a warmup that helps in warming up the cache values and is not counted to the 10 test runs of a test or suite.

Note:
Performance profiling can't be done on more than 1 test simultaneously. It doesn't support parallel performance assessment.



Note:
When you execute a performance test run, the system pauses and waits for any ongoing jobs to finish before starting the test run. This helps in avoiding any slowdown of your instance.

Execute performance profiling

Execute performance profiling on a test or a suite for performance testing on your instance. You can also detect performance degradation when you upgrade your instance and then investigate and fix the issues.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **Automated Test Framework (ATF)** > **Test**.

   Note:
   You can also select the Suites module if you want to execute performance profiling on a suite.

   A list of tests or suites shows up.
2. Select the test you want to execute performance profiling on.
3. Select **Run Performance Test** on the Test form.
   A **Run Test** modal shows up. The modal shows the warmup run and all the 10 test runs of the selected test or suite. You can track all the test runs as they start executing.

   Note:
   You can select **Run Performance Suite** if you have selected a suite for performance profiling.

4. Scroll down and select the **Performance Test Results** related list.
5. Select the required performance number to view the results of all the test runs within the selected performance test.
   The Performance Run form shows up. You can also directly navigate to **All** > **Automated Test Framework (ATF)** > **Test** > **Performance Profiling** > **Performance Runs** to see the Performance Run form. Add a screenshot

   Note:
   You can view the test name, status of the test runs, and the duration of the test run. If the **Is warmup** value is true, its the first test run that is done to warmup the cache values. The rest of the test runs will have **Is warmup** value set to false. The warmup test is not counted as one of the 10 test runs for each test or suite.

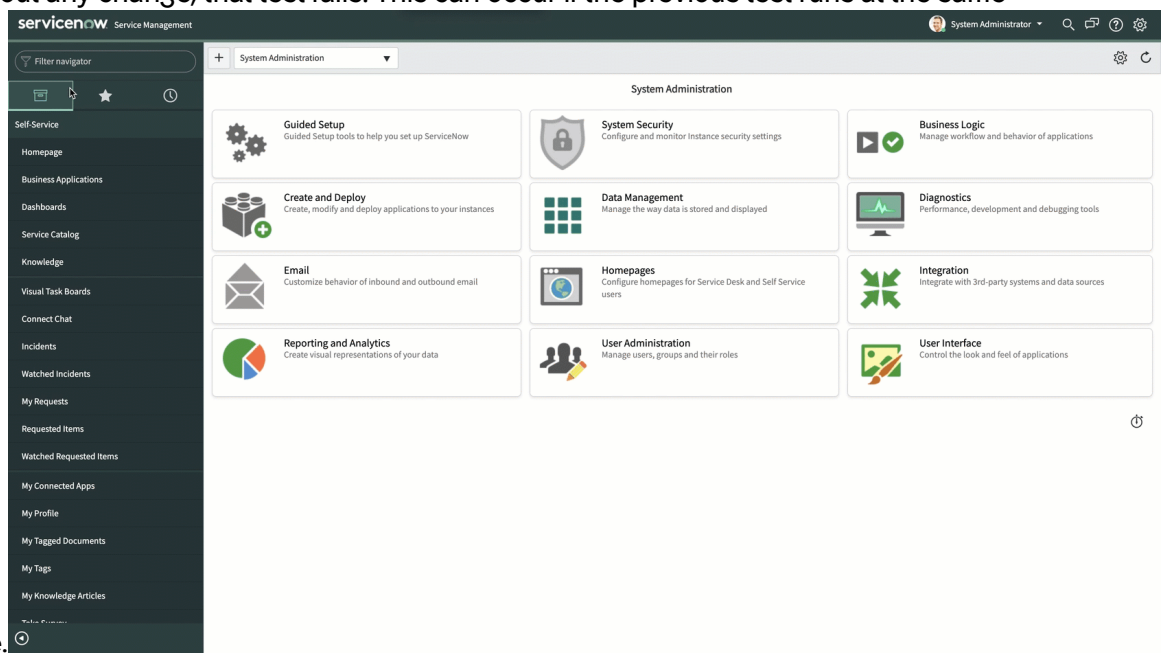
6. Navigate to **All** > **Automated Test Framework (ATF)** > **Test** > **Performance Profiling** > **Performance Comparisons**.
   The Performance Comparisons form shows up. <add a screenshot of the form>
7. Select the first and second run to view a comparison result.
   Add the complete screenshot. You can view information like performance means and the delta mean values.

   Note:
   You can compare only 2 test runs at one point of time. In the next comparison, you can consider one of the previous test run as the baseline to compare with the next test run.

Mutually exclusive tests

Prevent conflicting tests from running in parallel by marking them as mutually exclusive. For example, when the system identifies tests that modify the same record, the system makes these tests mutually exclusive. You can also manually mark tests as mutually exclusive.

The system marks tests as mutually exclusive when there is a potential resource conflict. When the system can't detect resource conflicts automatically, you can create your own mutual exclusion rules that can prevent conflicting tests from running in parallel. For example, if a test changes a sys_properties record, the record shows up under **Records Modified** of that test. If the validation path of another test depends on the same sys_properties record without any change, that test fails. This can occur if the previous test runs at the same time.



You can view mutually exclusive tests on the Mutually Exclusive Tests related list in the test form. This related list shows all tests that don't run in parallel with the current test and the reason. The same test might appear more than once in the list if there are multiple reasons.

Mutually exclusive tests reasons

Tests are marked mutually exclusive for the following reasons.

- Two or more tests modify the same record.
- A test that runs in parallel with itself.
- You can create your own mutual exclusion rules when the system can’t detect resource conflicts automatically.

Mark tests as mutually exclusive

You can mark tests as mutually exclusive using any of the following methods.

Mutually Exclusive Tests tab

Select a test from the Tests list and navigate to **Mutually Exclusive Tests** > **Add Mutual Exclusion** to make the selected test mutually exclusive with another test.

Tests list

Select one or more tests from the Tests list and choose **Add mutually exclusive test** from the Action on selected rows context menu. Enter one test in the Add mutually

exclusive test dialog box to make the selected tests mutually exclusive with the current test.

Parallel Test Runs tab

When two or more tests run in parallel, navigate to **Test Results** > **Parallel Test Runs**. Select one or more tests and choose **Add mutually exclusive test** from the Action on selected rows context menu to mark the selected tests as mutually exclusive.

Note:
The **Parallel Test Runs** tab is visible only if the test runs in parallel with one or more tests.

Automated Test Framework design considerations

Create reliable, scalable, and efficient tests by following these design considerations.

General testing

Avoid modifying ServiceNow system tables or tables extending the Application File [sys_metadata] that can potentially change the behavior of the system. Avoid using or modifying any existing records to prevent unexpected results between tests. The following are some of the common examples of system data changes that can cause unexpected results.

- Impersonate an existing account
- Delete an existing record.
- Run a test that disables a business rule or system property
- Validate with an existing record

Parallel testing

Reduce test design time by running multiple tests and test suites in parallel. Design tests to run in parallel by avoiding resource conflicts and data dependencies.

Prevent resource conflicts between parallel tests

Prevent resource conflicts by running tests that create their own data. Tests that run with existing data prevent other tests that need the same data from running in parallel.

Note:
If you have two or more resource conflicting tests, see Mark tests as mutually exclusive to create a mutual exclusion rule that prevents them from running in parallel.

Parameterized testing

Run a test multiple times with different test data for each run. Create parameters to store test data for each test run. See Parameterized test components for more information.

- Create parameters to store test data for each test run.
- Ensure that the parameterized tests support standard Automated Test Framework (ATF) features, such as reports, test suites, and data rollback. Copying a parameterized test copies all parameters, test run data sets, and test steps.

Note:
If a parameterized test including Custom UI test steps is created, the system only uses the first data set to retrieve components.

Custom UI testing

Test customized user interfaces such as UI pages and UI macros by retrieving their HTML and JavaScript page components and identifying the test actions they support.

Use the page inspector to identify testable page components

The page inspector determines which page components are available for custom UI testing. Page components that are unavailable to the page inspector are unavailable to custom UI testing.

Navigate to the custom UI you want to test

Use existing test steps to navigate to the target custom UI. For example, to test a Knowledge Base article, use the existing test steps to navigate to a module or to open an existing record. Most custom UI testing requires using existing test step categories as part of the test.

Use the component area to identify page components

The component area describes the HTML layout element containing the component such as a `<div>` or `<section>` element. The area helps test designers distinguish between components by providing the location in the page layout.

Test your custom UI rather than ServiceNow AI Platform UI

The Automated Test Framework prevents custom UI testing of ServiceNow AI Platform features. For example, you cannot test dashboards or graphical designers. Instead, build tests to validate your custom UI pages and elements because you have direct control over these user interfaces.

Use HTML attributes to override page component testing properties

Change the testing properties of a particular page component using HTML attributes that are specific to Automated Test Framework. See Override component test actions.

Retrieve page components again when you move tests to another instance

Custom UI test steps don't store UI components as metadata. Testers must manually retrieve page components again when moving tests between instances.

Clone tests from production system

Move your tests to the production system to clone the most updated instances for testing. Speed up the testing time by directly copying or cloning a test from the production system to a subproduction instance.

Note:
By default, the system property that is used to run automated tests is disabled to prevent you from accidentally running these tests on a production system. To avoid data corruption or an outage, run tests only on development, test, and other non-production instances.

Warning messages for all testing

Warning messages	Design considerations
`Impersonating an existing user may cause unexpected behavior for this test. Avoid potential issues by adding a 'Create a User' step instead. See the`	Create a new user to ensure proper roles and groups and avoid using existing records. See General testing for more information.

Warning messages	Design considerations
`documentation for Test Design Considerations.`	
`Using a table that extends Application File [sys_metadata] may cause unexpected behavior for other tests running in parallel. See the documentation for Test Design Considerations.`	Avoid running a test with a table that extends the Application File because it might affect other tests. See Parallel testing for more information.
`Using a system table may cause unexpected behavior for other tests running in parallel. See the documentation for Test Design Considerations.`	Avoid using a system table because it might affect other tests running in parallel. See Parallel testing for more information.
`Using an existing record may cause unexpected behavior for this test. See the documentation for Test Design Considerations.`	Avoid using existing records because these records might not have the state and values as expected by the test. Use records created during the test to ensure proper state and values. See General testing for more information.
`Modifying an existing record may cause unexpected behavior for other tests running in parallel. See the documentation for Test Design Considerations.`	Avoid using existing records because it might affect other tests. Use records created during the test. See General testing for more information.
`Using assert type '--None--' may cause unexpected behavior for server UI actions. Avoid potential issues by setting the assert type and using a timeout. See the documentation for Test Design Considerations.`	Server UI actions cause the current form to submit and the page to reload. Select an assert type other than **None** to avoid any unexpected behavior for server UI actions. Set a timeout to ensure that your test waits for the form to be submitted or not submitted before moving on to the next step. When testing server UI actions, the **None** assert type configures automatically to **Form submitted to server**.

Domain separation testing

When testing domain separation, you must set the domain first. This should be part of the first impersonation step of each of the ATF test steps when they are dependent on a domain being set. To learn more about domain separation recommended practices, see Domain separation recommended practices for service providers.

Cancelling automated tests and test suites

You can cancel automated tests and automated test suites that are running or are queued to run.

How you cancel an automated test or automated test suite depends on whether the test or test suite is currently running or is queued to run.

Cancel queued automated test suite

You can cancel an automated test suite that is queued but has not yet run.

Before you begin
Role required: atf_test_admin or atf_test_designer

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Run** > **Waiting/Running Test Runs**.
2. Select the action check boxes for the test suites to cancel.
3. From the action choice list, select **Delete**.

Cancel running and pending tests in an automated test suite

Cancel running and pending tests in a running suite test.

Before you begin
Role required: atf_test_admin or atf_test_designer

Procedure

1. If necessary, display the Run Test progress dialog.
2. Click **Cancel Pending Steps**.
   The system displays a dialog asking you to confirm that you want to cancel this test execution.
3. In the Confirmation dialog, click **Yes**.

Result
The system cancels any running and pending tests in this test suite and rolls back any data that tests in the test suite may have changed.

Cancel running automated test

You can cancel a running test from the Run Test progress dialog.

Before you begin
The system must be running a test.
Role required: atf_test_admin or atf_test_designer

Procedure

1. If necessary, display the Run Test progress dialog.
2. Click **Cancel Pending Steps**.
   The system displays a dialog asking you to confirm that you want to cancel this test execution.
3. In the Confirmation dialog, click **Yes**.

Result
The system cancels the running test and rolls back any data changes the test made.

Cancel waiting automated test

You can cancel a waiting automated test from the Waiting/Running Test Runs module.

Before you begin
Role required: atf_test_admin or atf_test_designer

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Run** > **Waiting/Running Test Runs**.
2. Select the action check boxes for the tests to cancel.
3. From the action choice list, select **Delete**.

Administering the Automated Test Framework

Enable or disable the Automated Test Framework, modify retention policies, move tests between instances, control user access to the Automated Test Framework, and create custom test step configurations and step environments.

For details about individual properties that control how the Automated Test Framework works, see Properties.

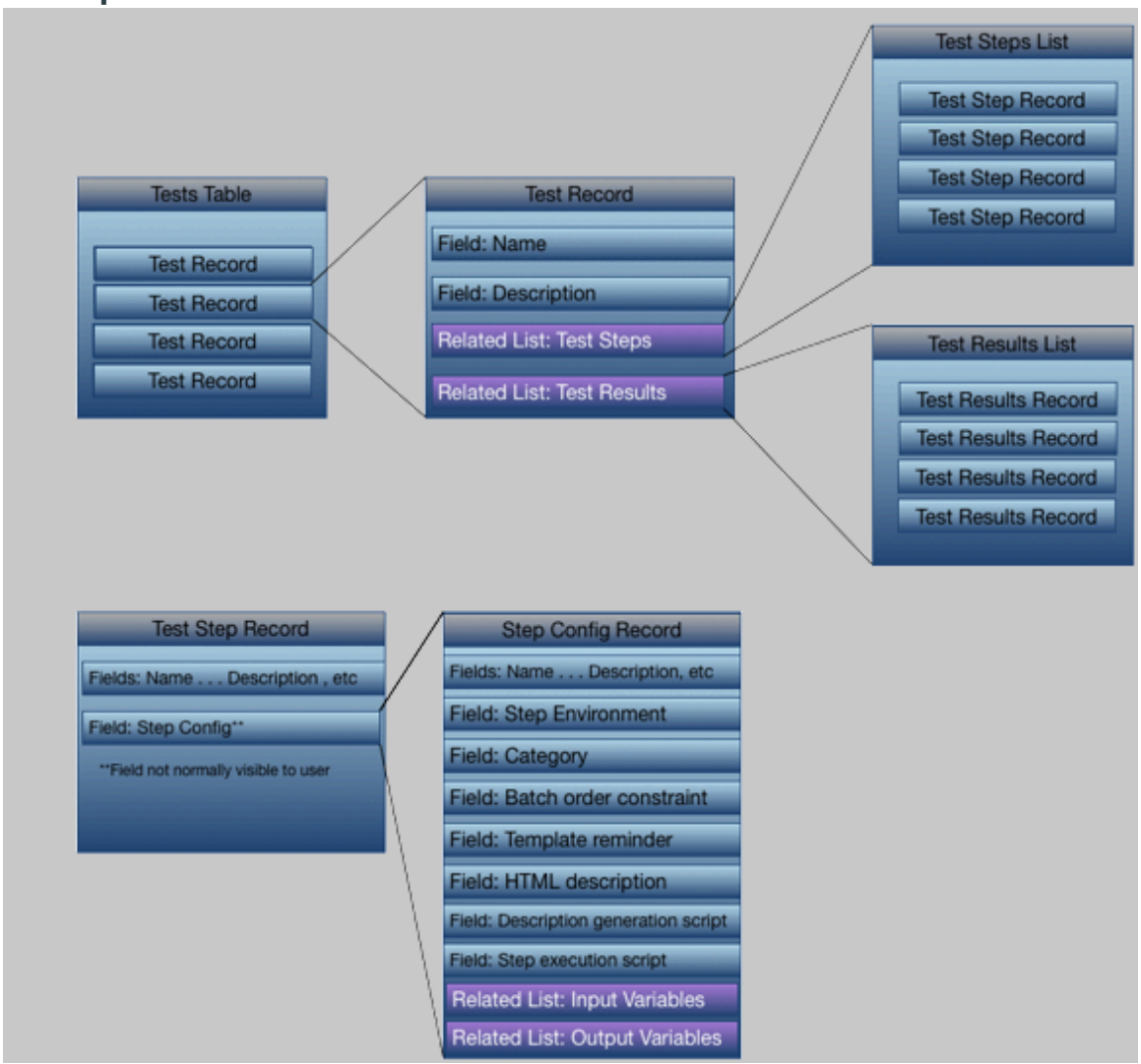
Creating custom test step configurations

Step configuration records (or step configs) define how each step type behaves. You can create new step configurations that define custom steps that run on the server.

When you add a step to an automated test, that step has a defined type such as Set Field Values, or Record Insert. Each step configuration has a number of characteristics that affect how steps of that type behave, including the inputs required, the actions performed, and so on.

These characteristics are defined in the step's step configuration record.

Test step architecture



TheStep execution scriptfield determines the actions the system executes when a step with this config runs.

Related topics

- Step configurations
- Step execution scripts

Create custom step configuration

Create a custom step configuration that can form the basis of new steps that run on the server.

Before you begin

Role required: atf_test_admin

> **Note:**
> Creating configurations are only supported in the server steps, the UI steps (running in the browser) are not supported. Also, only scripted steps are supported, custom Java-based steps are not supported.

About this task

The Automated Test Framework includes specific types of steps such as Open Form, Set Value, Assert Value and so forth. With the **Step Configurations** module, you can create steps that

perform actions you specify. You can only create configurations for steps that run on the Server. You cannot create configurations for steps that run on the browser.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Administration** > **Step Configurations**.
2. Click **New**.
   The system displays the **Test Step Config** form.
3. In the **Name** field, enter a name for your step type.
4. Leave **Active** checked.
5. Leave **Step environment** set to **Server-Independent**.
   You can define only step configurations that run on the server and not step configurations that run in the browser.
6. **Optional:** In the **Category** field, select the category to which you want to assign this step. Categories are used for filtering the step list in the Add Step dialog. For more information, see Category field example.
7. **Optional:** In the **Batch Order Constraint** field, choose one of the following values.
   - **None**: A step based on this configuration can appear at any point in a test.
   - **Start Batch Execution**: If this test includes a batch with this step, this step must be the first in the batch.
   - **Run in the middle of an execution**: If a test includes a batch with this step, this step must appear after the first and before the last step of the batch.
   - **Stop Execution**: If a test includes a batch with this step, this step must be the last step in the batch.
8. In the **Order** field, enter an integer specifying where steps with this configuration appear in the step list on the Add Test Step dialog.
   For more information, see the example using the **Order** field in Order field example.
9. In the **Template reminder** field, enter the instructions you want to appear when this step is included in a test as part of a template.
   For more information, see the example of using the Template reminder field.
10. In the **HTML description** field, enter the text you want to appear when the cursor highlights this step on the Create New Step dialog.

    For more information, see the example using the HTML description field in .

    Note:
    The next two steps involve writing scripts, which you should wait to complete until you have added input and output variables.

11. In the **Description generation script** field, add code to the provided template to generate the description assigned to a Test Step record when a step of this type is included in a test.
    For more information about writing this script, see Step description generation script. To see an example of where the system displays this description, see Description generation script example.
12. In the **Step Execution Script** field, add code to the provided template to define the script that executes when a step of this type runs.
    The script template provides instructions and examples for working with step inputs, outputs, and step results. For more details on the step execution script, see Step execution scripts.
13. Click **Submit**.
    The system creates a new test step configuration and returns to the list of test configurations.

14. **Optional:** Add input variables to this step config.
    a. Access the step config record.
    b. Scroll to the Input Variables tab, then click **New**.
    c. Fill in the required fields for the new variable.
    d. Click **Submit**.
    e. Repeat these steps until you have added all the input variables needed.
    f. **Optional:** Control the order in which input variables appear on the New Step form by editing the values in the **Order** column for the Input Variables related list.
15. **Optional:** Add output variables to this step config.
    a. Scroll to the Output Variables tab, then click **New**.
    b. Fill in the required fields for the new variable.
    c. Click **Submit**
    d. Repeat these steps until you have added all the output variables needed.

Related topics

Step execution scripts

Step configurations

Automated Test Framework Step Config record

Add a predefined list of steps (template) to an automated test

Create a custom step configuration category

Create a custom step config category.

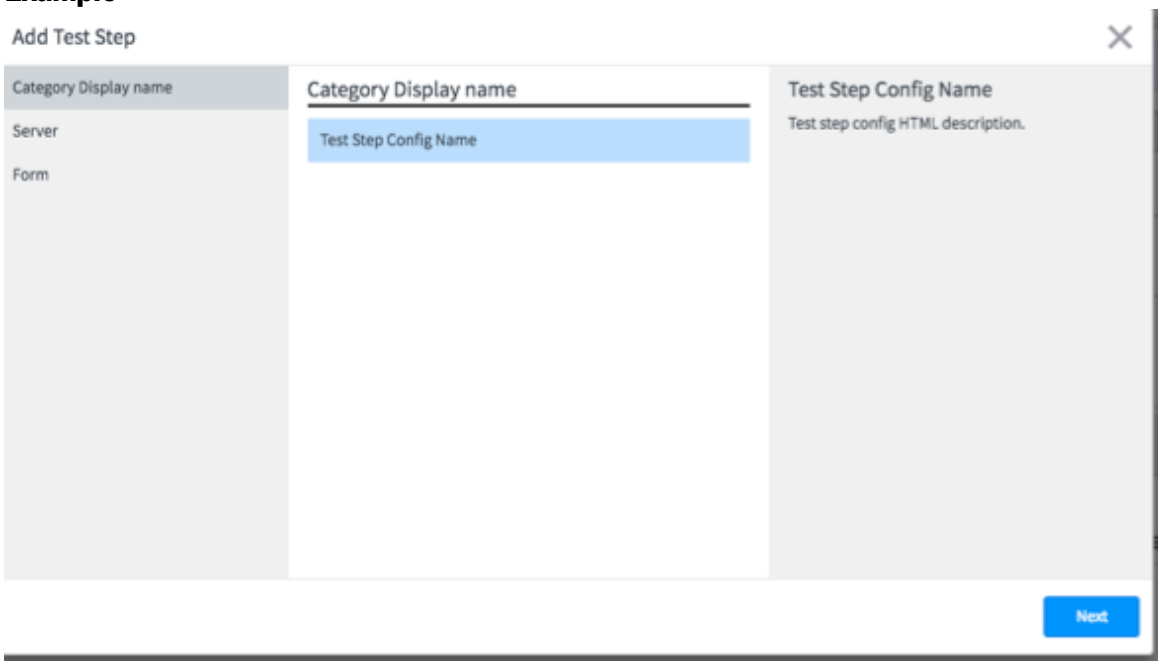
Before you begin
Role required: atf_test_admin

About this task
Categories are used for filtering the step list in the Add Step dialog. For more information, see Category field example.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Administration** > **Step Configuration Categories**.
2. Click **New**.
   The system shows the **Test Step Config Category** form.
3. In the **Name** field, enter a name for your step category.
4. In the **Step Environment** field, enter the step environment in which steps under this category execute:
   - Server - Independent, if you want this category to contain steps that execute on the Server.
   - UI, if you want this category to contain steps that execute on browser.
   - Server - REST, if you want this category to contain steps that send Inbound REST messages to the instance.
5. In the **Display name** field, enter the category name you want to appear in the middle column of the Add Test Step dialog when this category is selected.

Example



6. Click **Submit**.
   The system creates a test step category and returns to the list of test step environments.

Working with test step templates

Test step templates contain a list of steps to be added all at once to an automated test.

Related topics

Add a predefined list of steps (template) to an automated test

Create an automated test steps template

Reduce testing time by creating a template containing a list of steps to add all at once to an automated test.

Before you begin

Role required: atf_test_admin and atf_test_designer

About this task

Many tests follow similar patterns. One common pattern, for example, is to open a form, set some field values, validate some field values, click a UI action, open a record producer, open a catalog item, and submit the current form. If a template exists containing these steps, you can add them to a test all at once. The Automated Test Framework comes with default templates in the base system. With this procedure, you create your own templates.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Administration** > **Test Templates**.
   The system displays the Test Templates list.
2. Click **New**.
   The system displays the Test Template form.
3. In the **Name** field, enter a name for your template.
4. In the **Test Template** field, click the lock icon ( ).
   The **Test Template** field unlocks and expands to allow editing.

5. In the **Test Template** field, enter the name of the first test step to add to this template.
   The system adds the test step to the Test template list.
6. Continue adding test steps – in the order that you want them to appear – until you've added all the steps that you want to include in the list.

   Note:
   The test template supports tables, catalog items, and the record producer.

7. In the **Description** field, enter a description for this template.
8. Click **Submit**.

Related topics

Add a predefined list of steps (template) to an automated test

Edit automated test steps template

Edit an existing test template.

Before you begin
Role required: atf_test_admin and atf_test_designer

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Administration** > **Test Templates**.
   The system displays the **Test Templates** list.
2. Click the row for the test template you want to edit.
   The system displays the Test Template form.
3. In the **Test Template** field, click the lock icon ( ).
   The Test Template field unlocks and expands to allow editing.
4. **Note:**
   You can delete steps anywhere in the list, but you can add steps only to the end of the list. You can re-order steps in a test after you add them using the template, but you cannot re-order steps in the template itself.

   Add or delete steps to the template.
   - To delete a step, select that step, then click the **X** icon.
   - To add a step, in the **Test Template** field, enter the name of the step to add.
5. Continue adding and deleting test steps until it contains the steps you want.
6. Click **Submit**.

Related topics

Add a predefined list of steps (template) to an automated test

Enable or disable executing Automated Test Framework tests

Allow or prevent tests and test suites from executing on this instance.

Before you begin
Role required: atf_test_admin

About this task
By default, the system property that is used to run automated tests is disabled to prevent you from accidentally running these tests on a production system. To avoid data corruption or an outage, run tests only on development, test, and other non-production instances.

Procedure
1. Navigate to **All** > **Automated Test Framework** > **Administration** > **Properties**.
2. Set the test execution property.
   - To enable test and test suite execution, select **Enable test/test suite execution**.
   - To disable test and test suite execution, clear **Enable test/test suite execution**.
3. Select **Save**.

Modify data retention policy for ATF test results

Modify the Auto Flush data retention policy, which designates how long the system retains data, and referencing data, for test and test suite results. You can change the frequency of flushing for the sys_atf_test_result or sys_atf_test_suite_result base tables. This setting controls how far back in time test result data is available.

Before you begin
Role required: atf_test_admin

About this task
The system regularly flushes data in the sys_atf_test_result and sys_atf_test_suite_result base tables, (and optionally, referencing data). By default, the system deletes test and test suite results data 30 days after creation. This task enables you to modify the Auto Flush retention policy for data stored in a specific base table (sys_atf_test_result or sys_atf_test_suite_result).

Procedure
1. Navigate to **All** > **Automated Test Framework** > **Administration** > **Table Cleanup**.
   The system displays a list of the retention policies (Auto Flushes) it maintains for automated testing results tables.
2. Select the retention policy (sys_atf_test_result or sys_atf_test_suite_result) to modify.
   The system displays the record for this retention policy.

   Note:
   The **Tablename** field displays the name of the table to which the selected Auto Flush retention policy applies. Selecting another tablename in this field compromises the integrity of the Auto Flush record. Leave the tablename on existing ATF policies at the base system (default) value so it does not adversely affect ATF data retention behavior.

3. Specify how the system should determine the length of time for retention of data, and referencing data.

   a. In the **Matchfield** field, enter the field you want the system to use to monitor duration. For example, to specify that you want to delete data x amount of time after the system created it, leave **Matchfield** set to its default value of **sys_created_on**.

   b. In the **Age in seconds** field, enter the amount of time (in seconds) the system must wait before deleting the associated data and referencing data.

4. If you want to apply the policy to the specified data (for example, sys_atf_test_result), and to any data that references it, select **Cascade delete** (default value).
   Affected referencing data is stored the following tables: sys_atf_test_result_item, sys_atf_test_result_step, and sys_attachment (when table_name = sys_atf_test_result). If you want the policy to simply flush data in the selected table (for example, sys_atf_test_result), and skip flushing of the referencing data, then clear **Cascade delete**.
5. In the **Conditions** field, specify the filter conditions to use for selection of data (and optionally, referencing data) for this Auto Flush retention policy.
   The default is **Retain indefinitely is false**, because the Test results record also contains a **Retain indefinitely** check box that allows opting out of the auto flushes for specific test results.
6. Click **Update**.

Manage status and retention policies for automated test client runners

Modify how often active client test runners report in to the system and how long the system retains records for inactive client test runners.

Before you begin
Role required: atf_test_admin

About this task
When you start a client test runner, the system registers that runner as active, meaning that it is either running a test or is available to run a test. While the runner is active, it reports in to the system at a specified interval. If the runner does not report in at the expected time, the system marks the runner as inactive. After a period of time the system deletes the runner. This task enables you to modify the Automated Test Framework properties that control these intervals.

Procedure
1. Navigate to **All** > **Automated Test Framework** > **Administration** > **Properties**.
2. Navigate to the Test Runner Properties section.
3. To set the interval at which active client test runners report in to the system, enter the reporting interval in seconds in the **Test runner heartbeat interval** field.
4. To set the period of time a test runner can remain inactive before the system deletes it, enter the period of time in seconds in the **Test runner timeout** field.
5. Click **Save**.

Related topics
- Working with client test runners
- Active manual test runners
- Active scheduled test runners

Moving automated tests from one instance to another

Move automated tests from one instance to another using the normal process for update sets.

Role required: atf_test_admin

You can move automated tests, automated test suites, and related data using update sets. For more information, see System update sets.

Compare results and execution times for different automated test and suite results

You can compare execution times for different runs of an automated test or automated test suite. You can also compare results over time for a single automated test suite.

Before you begin

Role required: atf_test_admin or atf_test_designer

Compare execution times for different runs of an automated test

Compare how long it took the system to execute each test step across different runs of the same test.

Procedure

1. Navigate to **Automated Test Framework** > **Tests**.
2. Select the row for the test whose results you want to compare.
   The system displays the Test form.
3. Navigate to the **Test Results** related list.
4. Check the rows for the test results you want to compare.
5. From the **Actions on selected rows** control, click **Compare test step results**.
   The system displays the Compare test result execution times bar graph.

Compare execution times for different runs of the same automated test suite

Compare how long it took the system to execute each test across different runs of the same test suite.

Procedure

1. Navigate to **Automated Test Framework** > **Suites**.
2. Select the row for the test suite whose results you want to compare.
   The system displays the Test Suite form.
3. Navigate to the **Test Suite Results** related list.
4. Check the rows for the test suite results you want to compare.
5. From the **Actions on selected rows** control, click **Compare test results**.
   The system displays the Compare test result execution times bar graph.

Compare results for automated test suite runs (aging report)

Compare how many tests passed versus failed across different runs of the same test suite.

Procedure

1. Navigate to **Automated Test Framework** > **Suites**.
2. Select the row for the test suite whose results you want to compare.
   The system displays the Test Suite form.
3. Under Related Links, click **Display aging report**.
   The system displays the Test suite aging report.

Administering REST test step configurations

Set request and response payload sizes, filter request and response headers, and create basic auth profiles.

Create a basic auth profile using the Automated Test Framework

Create basic auth profiles to specify basic authentication credentials for Send Request - Inbound test steps.

Before you begin
Role required: web_service_admin

About this task
The user name and password must be valid credentials on the instance where the tests using the profile are run.

Procedure
1. Navigate to **All** > **Automated Test Framework** > **Tests**.
2. Select a test that uses a Send Request - Inbound step.
3. Select a Send Request - Inbound step.
4. In the **Basic authentication** field, select the hour glass to look up the available profiles.
5. On the **Basic Auth Configurations** form, select **New**.
6. In the **Name** field, enter a name for the profile.
7. In the **Username** field, enter a user name.
8. In the **Password** field, enter a password.
9. Click **Submit**.

Filter REST request and response headers
You can add a list of REST request and response headers that are not to be saved in step-result records. You can filter headers that might contain authentication credentials or other sensitive information. The phrase "Header redacted for security" is saved instead.

To specify headers to be filtered, create a system property `glide.atf.rest.log.header_blacklist` with a comma-separated list of header names to be filtered. For information on adding properties, see Add a system property.

Automated Test Framework REST properties
These properties are installed with ATF REST.

Note:
To open the System Property [sys_properties] table, enter `sys_properties.list` in the navigation filter.

Property	Usage
glide.atf.rest.log.header_blacklist	A list of headers whose content is not to be added to the log, or shown on a form. The phrase 'Header redacted for security' is saved instead. • **Type**: String • **Default value**: empty • **Other possible values**: A comma-separated list of header names. • **Location**: System Property [sys_properties] table
glide.atf.rest.request_payload_max_size	The maximum size of the request payload.

Property	Usage
	• **Type**: String • **Default value**: 100 Kb • **Maximum value**: 1024 Kb • **Location**: System Property [sys_properties] table
glide.atf.rest.response_payload_max_size	The maximum size of the response payload. • **Type**: String • **Default value**: 100 Kb • **Maximum value**: 5120 Kb • **Maximum value**: System Property [sys_properties] table

Optimizing automatic test performance

You can troubleshoot automatic test performance by inspecting system transaction log records and potentially shorten execution time by adjusting how often automatic tests capture screenshots.

Managing automatic test screenshot settings

Capturing many screenshots can impair test performance. You can control which types of screenshots the system captures to minimize this effect.

By default, the system captures a screenshot every time it executes a form test step. This information can be useful for understanding test results, but may slow down how fast the system executes the test.

You can change automatic test framework settings so that the system captures all screenshots (as it does by default), no screenshots, or just screenshots for failed test steps.

You can change these settings to affect all tests run on this instance, or to affect just the current client test runner session. To affect all tests run on this instance, set the automatic test framework property from the automatic test framework properties page. To affect just the current client test runner session, set the screenshot mode from client test runner browser window.

Set the system property to control when the Automated Test Framework captures screenshots

To control how often this instance captures screenshots for form test steps, set the screenshot capture mode on the automatic test framework properties page.

Before you begin

Role required: atf_test_admin or admin

About this task

Setting the screenshot mode from the automatic test framework properties page affects any new client test runners started on this instance. This setting does not affect currently-running test-runners.

Procedure

1. Navigate to **All** > **Automated Test Framework** > **Administration** > **Properties**.
2. Set the **Enable or disable screenshot capture** property from the drop-down menu.
   - To capture screenshots for all steps, select **Enable for all steps**.
   - To capture screenshots only for failed steps, select **Enable for all failed steps**.
   - To capture no screenshots, select **Disable for all steps**.
3. Set the **Screenshot timeout** time interval, in seconds.
   The Client Test Run does not take a screenshot capture if the length of the attempt exceeds this value. Users should review performance settings and browser caches on affected client systems before increasing this value.
4. Click **Save**.

Set the client test-runner property to control when the Automated Test Framework captures screenshots

To control how often the current client test runner captures screenshots for form test-steps, set the screenshot capture mode on the client test runner browser window.

Before you begin
Role required: atf_test_admin or admin

About this task
Setting the screenshot mode from the client test-runner browser window affects only this client test-runner and persists until this test-runner session is closed. This setting does not affect any other running test-runners or any future test runners.

Procedure

1. From the client test-runner browser window, click the form preferences icon ().
2. Click the **Screenshot mode** option.
   - To capture screenshots for all steps, select **Enable for all steps**.
   - To capture screenshots only for failed steps, select **Enable for all failed steps**.
   - To capture no screenshots, select **Disable for all steps**.
3. Click **Save**.

View transaction data for automated test results

To help troubleshoot performance issues with automatic tests, you can inspect related records from the transactions log entry [syslog_transaction] table.

Before you begin
Role required: test_admin or admin

About this task
Note:
The system may not be able to log some transactions with short durations.

Procedure

Navigate to related transaction record in the system transaction log.

- From the Step results record, view transactions in the Step transactions related list.
- From the Test Results record, view transactions in the Test Transactions related list.

Working with scheduled test suites

You can schedule a test suite to run at a specified date and time.

To schedule a test suite, you need three components:

- a test suite record
- a schedule record specifying when you want the system to run the test suite
- a scheduled suite run record that associates the test suite to run with the schedule for running it

With this model, you can associate a schedule with many different test suites and vice-versa.

Note:
You can schedule only test suites, not individual tests. Scheduled tests will only run if there is an open Scheduled Client Test Runner page matching the scheduled suite's browser conditions. Scheduled tests cannot run on a machine that is locked, powered down, or does not already have the browser open.

The watchlist on the test suite run record also allows you to specify users to receive an email when the system finishes executing the test suite run.

If the test suite contains one or more form steps (steps involving a user interface), you must ensure that a scheduled client test runner is actively running in a browser when the schedule triggers the suite run.

Note:
See Browser recommendations and requirements for recommendations and requirements for running the client test runner.

For step-by-step instructions on how to schedule a test suite, see Schedule an automated test suite.

Designate users to receive email when system finishes running a scheduled test suite

You can designate users to be notified when a scheduled test suite finishes executing.

Before you begin

Role required: atf_test_admin

Procedure

1. Navigate to the scheduled test suite run record.
2. Add to the record's **watchlist** all users you want the system to notify when this scheduled test suite run completes.

Automated Test Framework scheduled test suite completed email

When the system completes executing a scheduled test suite, it sends an email to users on the Scheduled Suite Run record watchlist. This email contains information and links to further information about the Scheduled Suite Run and its results.

Note:
Email functionality must be enabled in order to send an email.

Suite Stats

The Suite Stats section of the email reports the number of suites and individual tests broken down by result status: Failed (F), Error (E), Skipped (S), Canceled (C), or Passed (P). For a description of what each status means, see Test suite results record.

Test Suite Results

The Test Suite Results section of the email reports test suite results over time.

Note:
By default, this report includes only suites with failed tests, but you can change this setting with the **Email properties** field on the Properties page. If set to default, only the failed tests within a test suite are reported.

Preview email report

Note:
ATF reports only recent tests results on the email report. All the previous test results still exist if it's within the set retention time limit. See Table cleanup for more information on retention policy for ATF test results.

Test Results Key

Letter	Color	Description
F	red	Test failed
P	green	Test passed

Test Results Key (continued)

Letter	Color	Description
S	white	Test skipped
C	white	Test canceled
E	orange	Test has error

Each entry acts as a link to the result record for that run. If you point the mouse to any of these entries, the system displays the parent suite number, the parent suite end time, and the test result output.

Properties affecting email content

On the Automated Test Framework Properties form, you can set options affecting the format and content of the email.

Automated Test Framework use case examples

Use cases can help you construct tests for common scenarios.

Automated Test Framework use case: test basic form operations

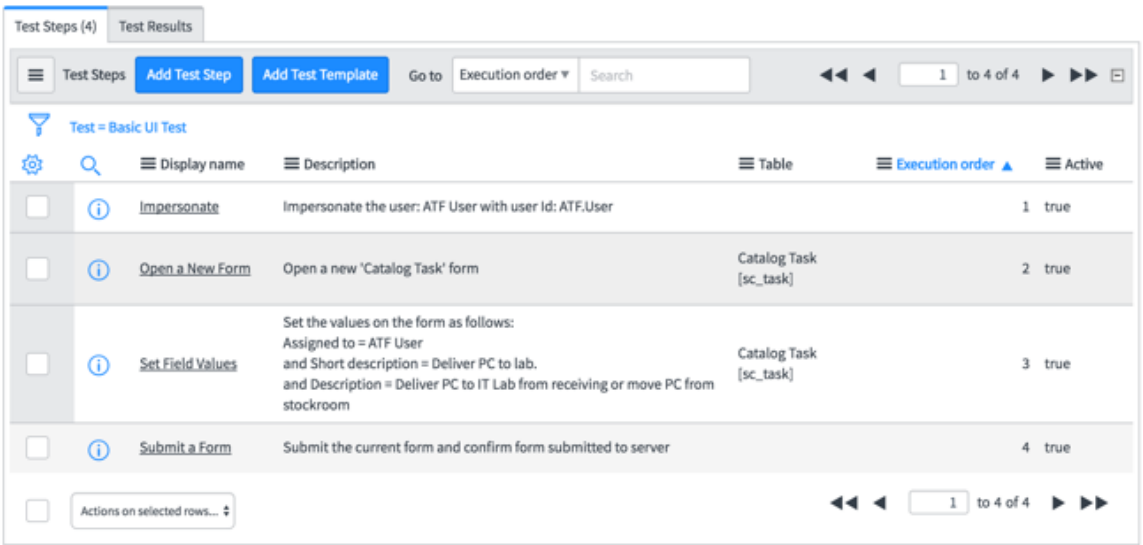
This use case illustrates testing basic form operations with the Automated Test Framework.

Before you begin

Role required: atf_test_admin

About this task

Steps in test

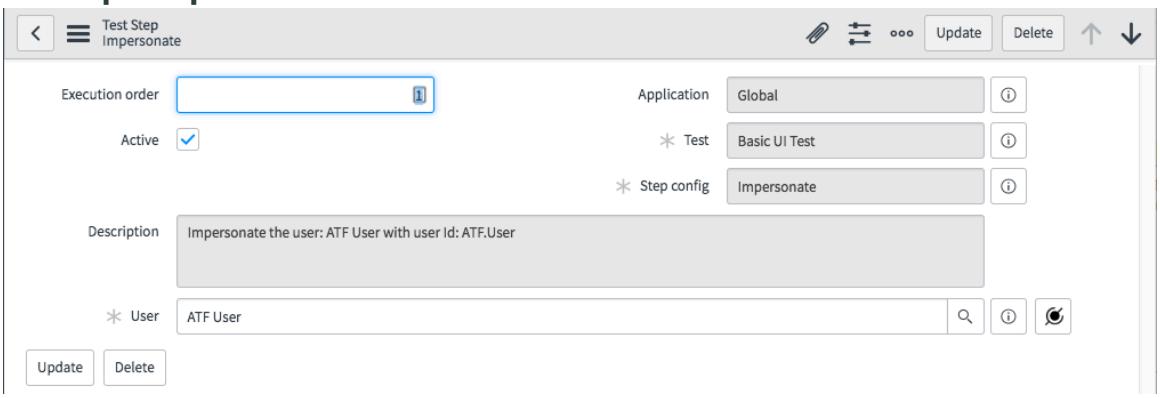


Procedure

1. Impersonate a user with the permissions needed to perform these steps, in this example ATF.User.

   Example

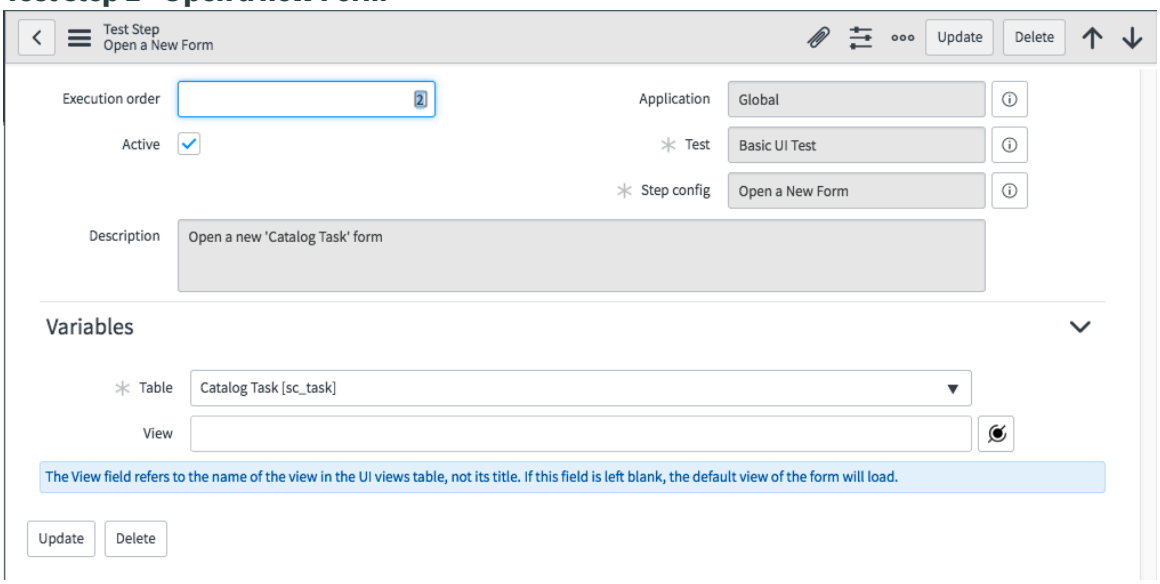
   Test step 1 - Impersonate

2. Open a form, in this example a Catalog Task form.

   Example

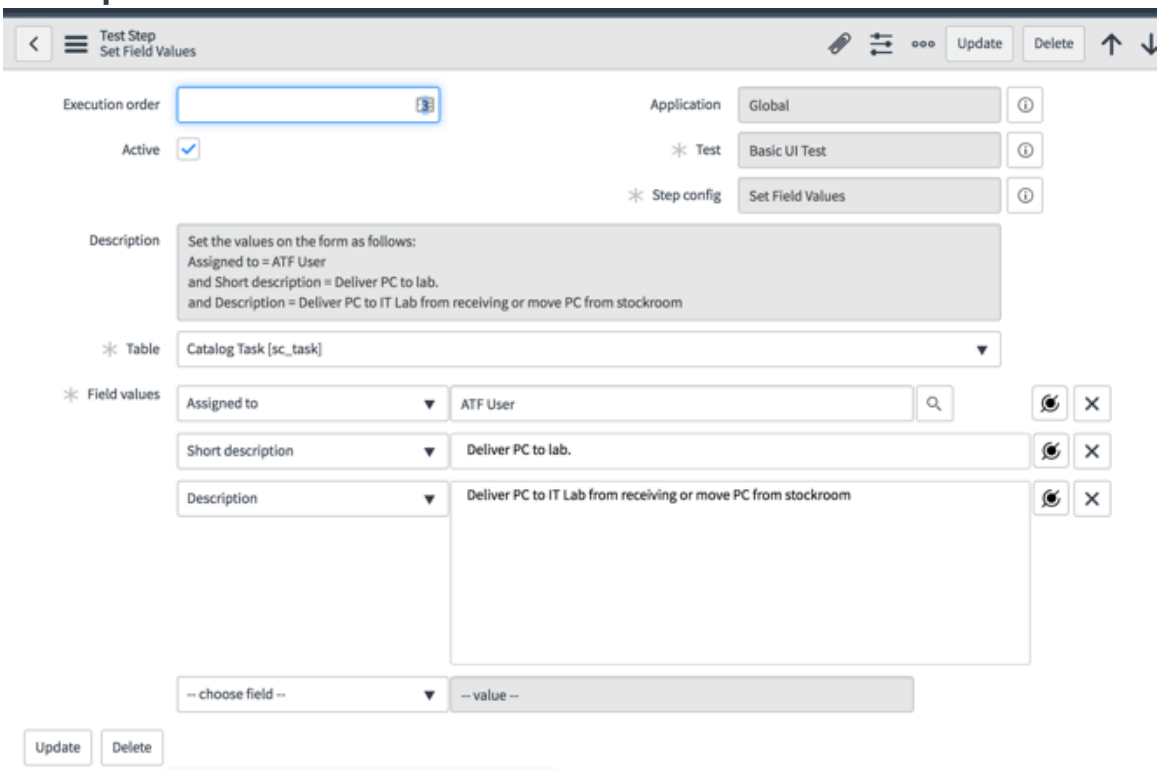
   Test step 2 - Open a new Form

3. On the open form, set field values, including for any mandatory fields.
   This example sets field values for **Assigned to**, **Short description**, and **Description**.

Example

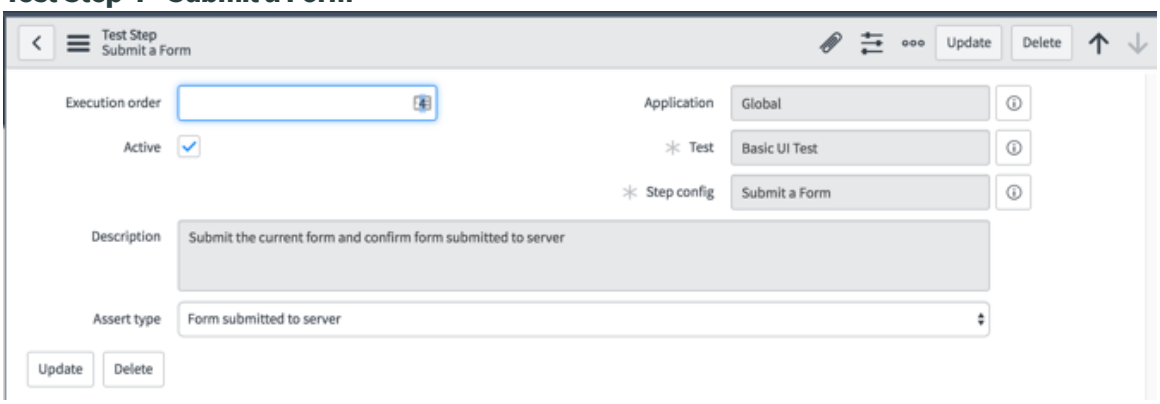
Test step 3 - Set Field Values



4. Submit the open form.

Example

Test Step 4 - Submit a Form



Automated Test Framework use case: reference a value from a previous step

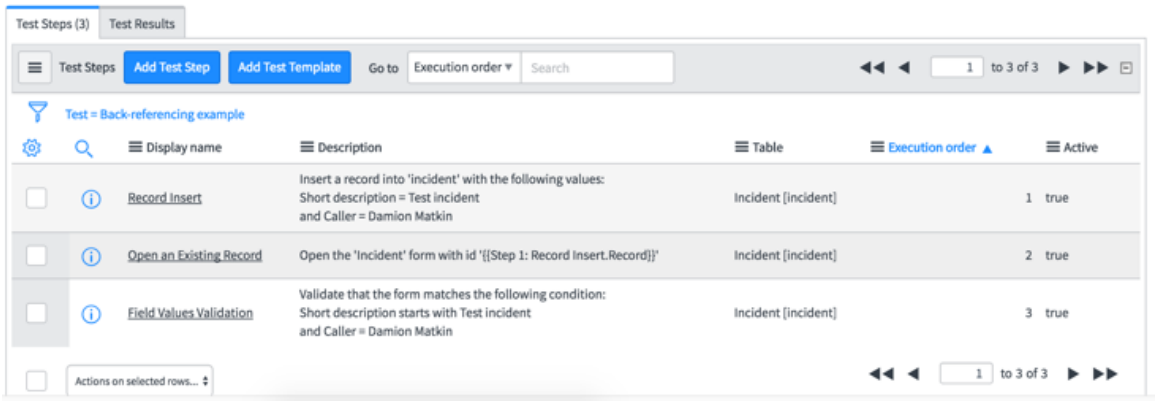
This use case illustrates assigning a form field the value of an output variable from a previous step.

Before you begin

Role required: atf_test_admin

About this task

Automated Test Framework: In this example, the second step references an output value from the first step. Pass values from one step to another example.

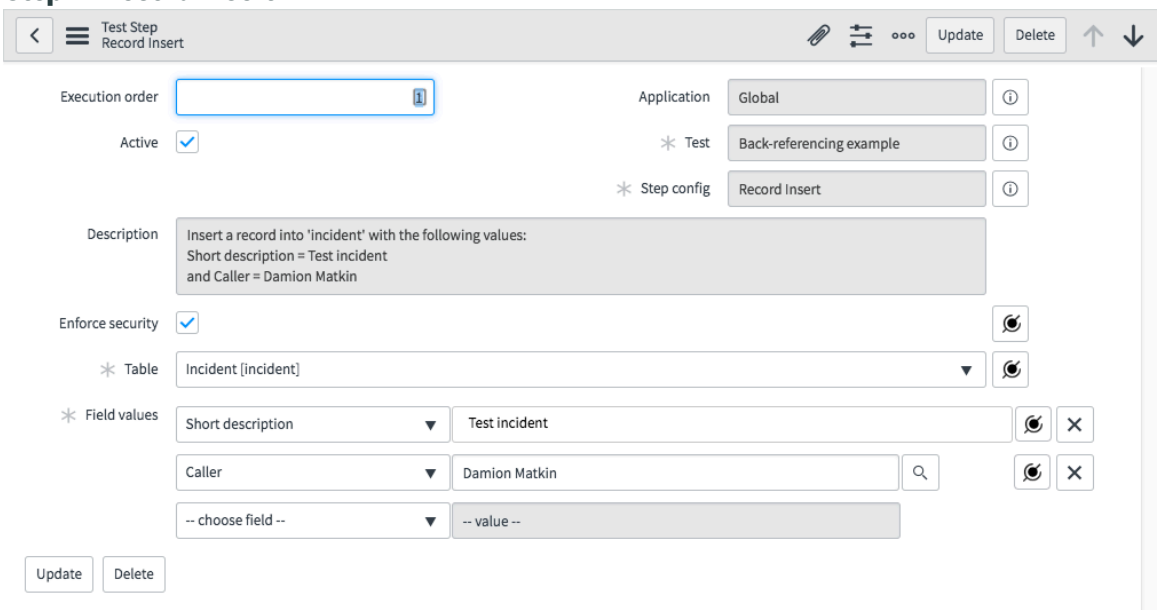


Procedure

1. Insert a record into the incident table.
   This example inserts a record into the Incident table.

   ### Example

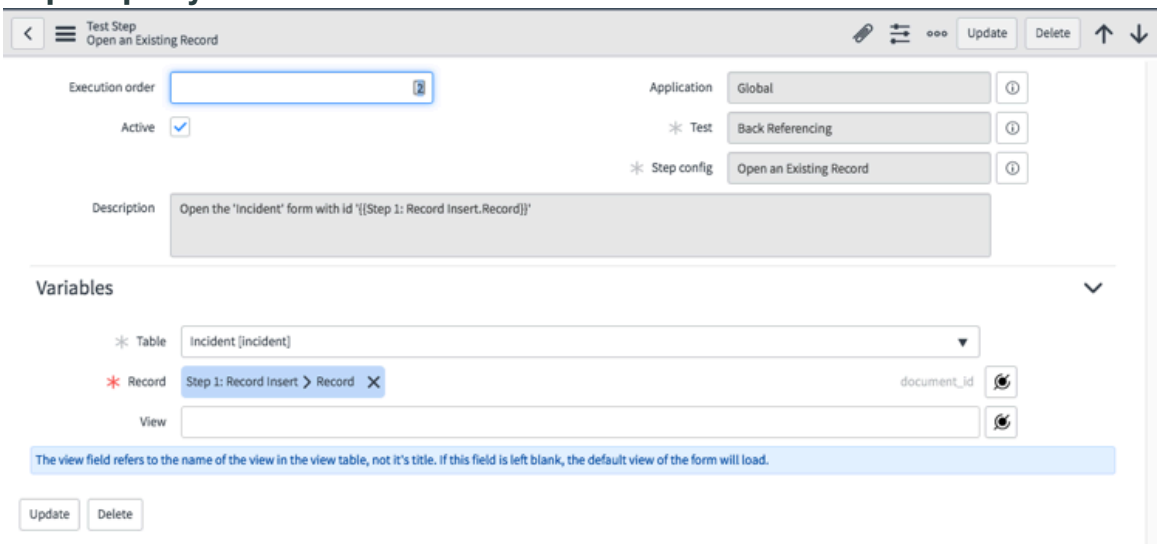
   Step 1 - Record Insert

2. Open the record just inserted by assigning to the *Record* field the output variable from Step 1.

Example

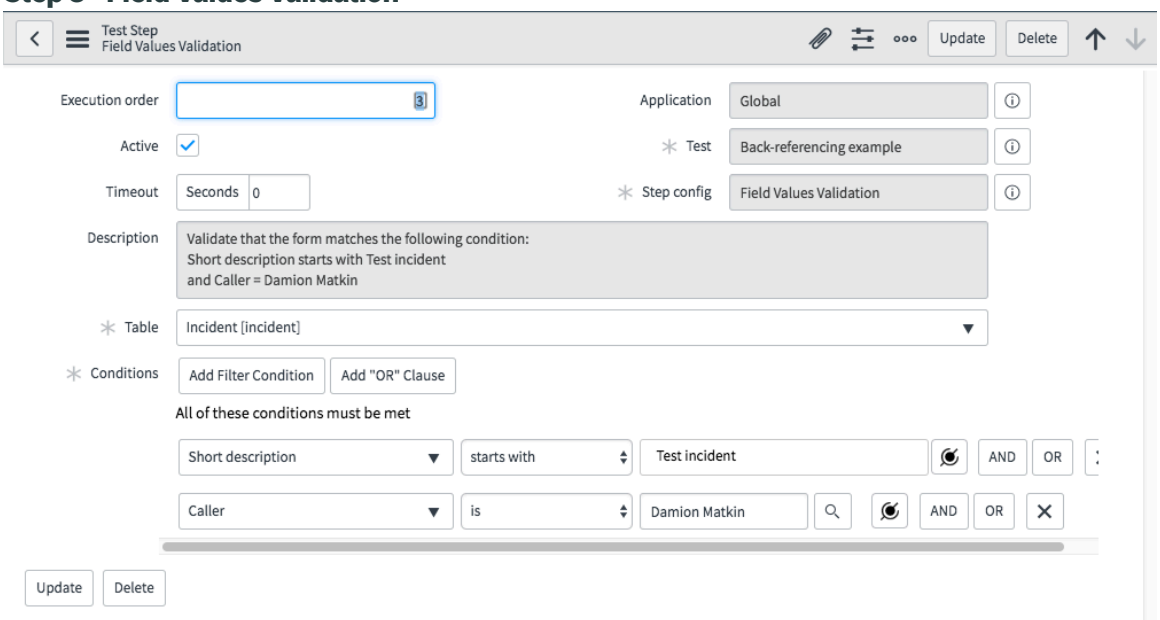
Step 2 - Specify the record



3. Validate that fields on the open record have the values you expect.

Example

Step 3 - Field Values Validation



Related topics

Pass values from one automated test step to another

Automated Test Framework use case: test a business rule

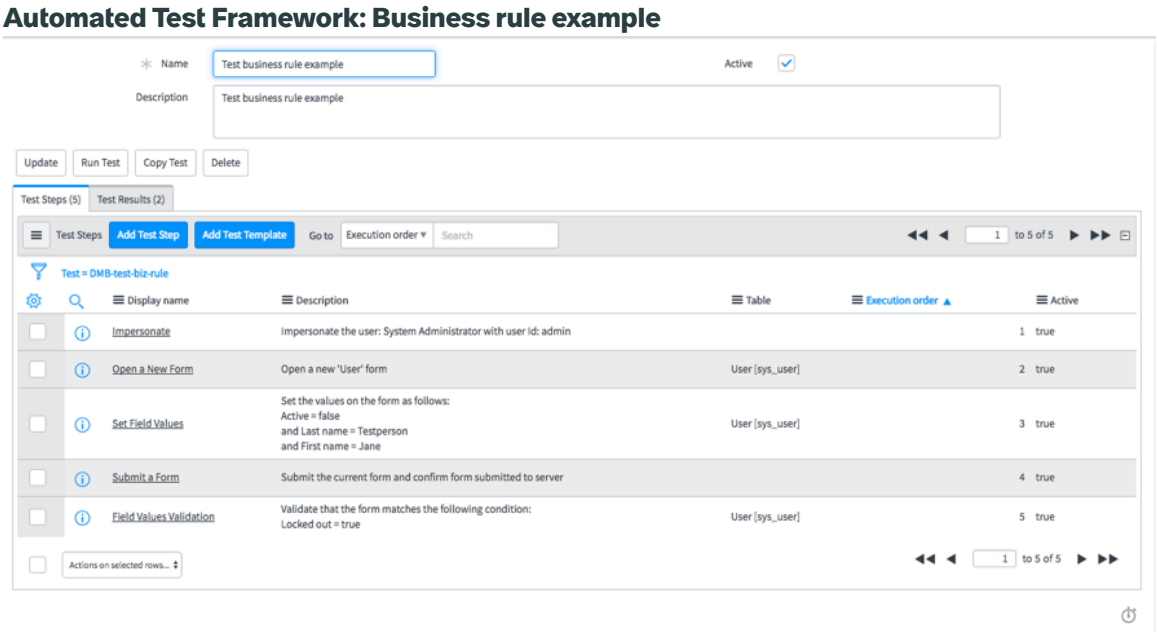
This use case illustrates testing a business rule with the Automated Test Framework.

Before you begin

Role required: atf_test_admin

About this task

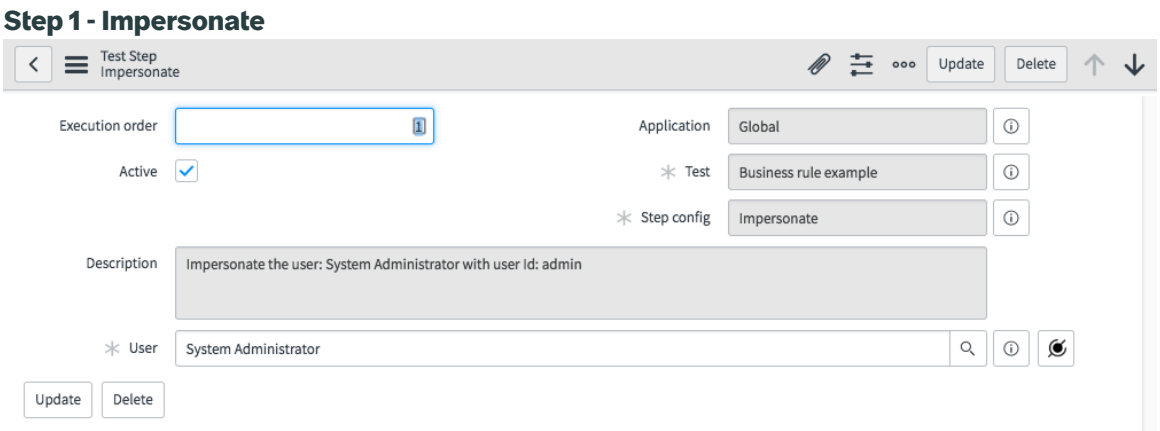
This example tests a business rule that sets the value of **Locked out** to **true** when **active** is set to **false**.



Procedure

1. Impersonate a user with the necessary permissions.
   In this example, the step impersonates the admin user.

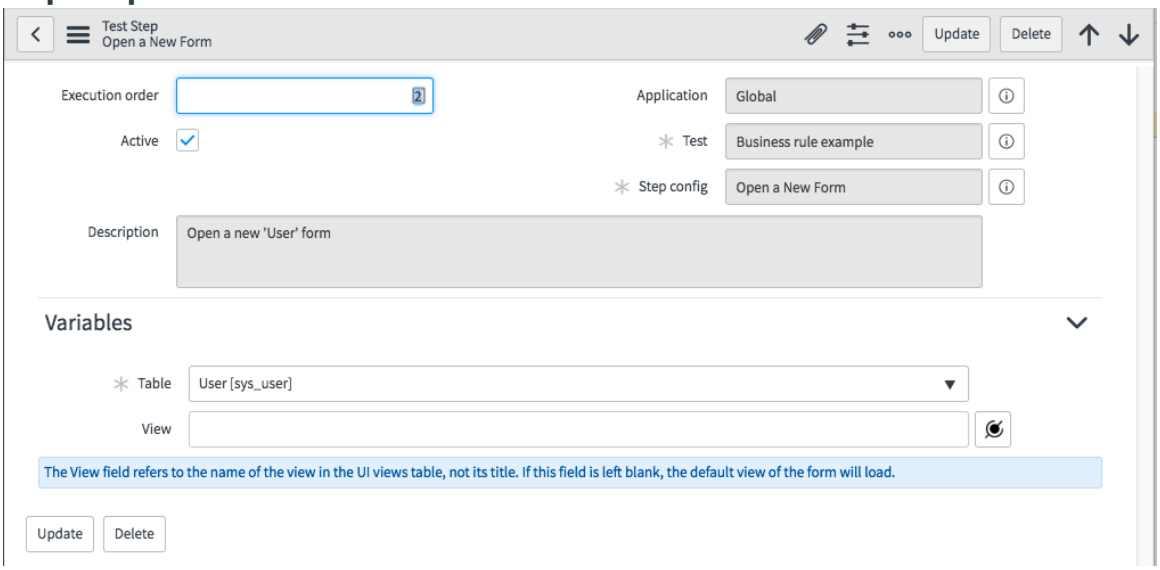
   Example



2. Open a form for the table to which this business rule applies.
   This example opens a new User form.

Example

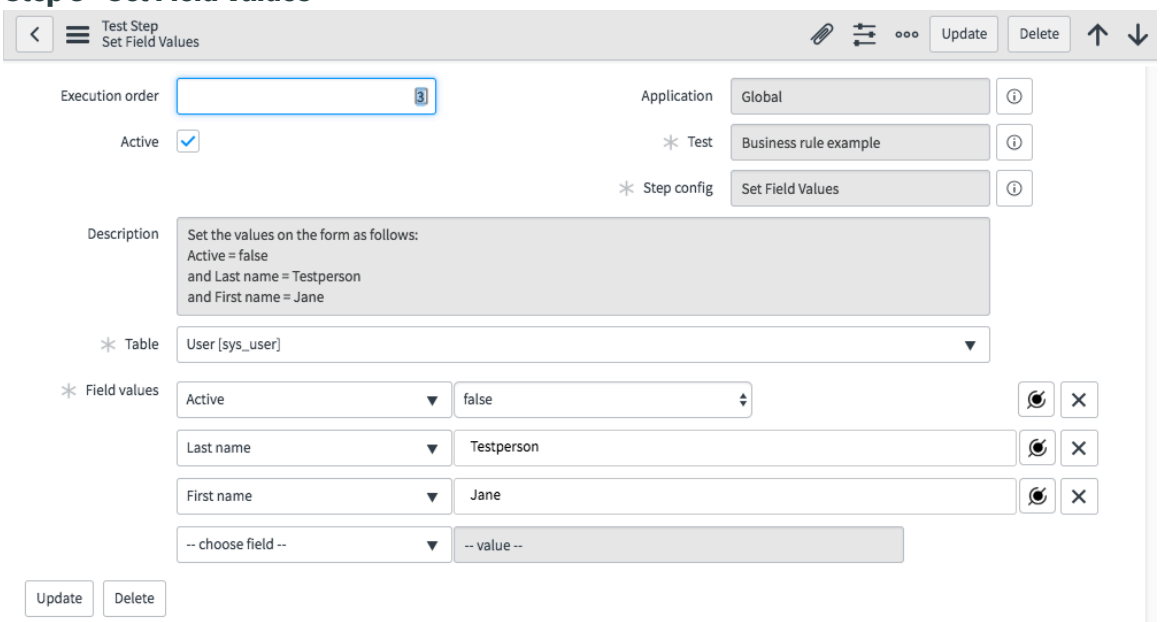
Step 2 - Open a New Form



3. Set values on the form that meet the requirements for submitting the form and for triggering the business rule.
   This example sets values for the **Active**, **Last name**, and **First name** fields.

Example

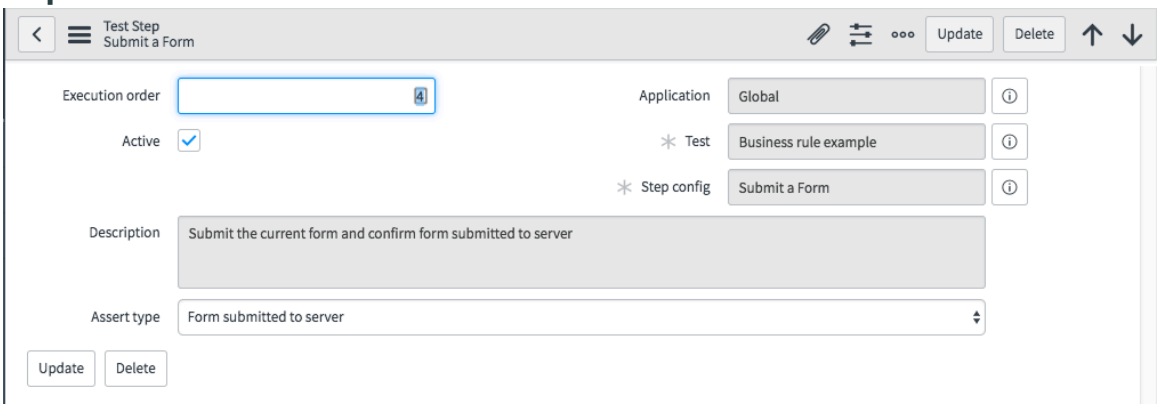
Step 3 - Set Field Values



4. Submit the form.

Example

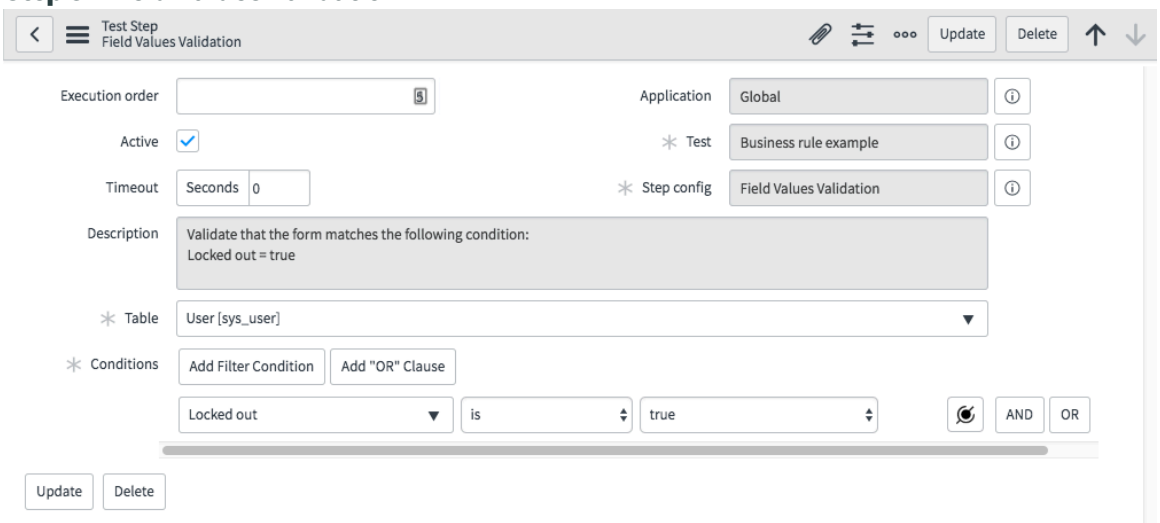
Step 4 - Submit Form



5. Validate that the business rule ran.
   In this example the business rule tested sets **Locked out** to **true** if **Active** is set to **false**.

Example

Step 5 - Field Values Validation



Automated Test Framework use case: test a data policy

This use case illustrates testing a data policy with the Automated Test Framework.

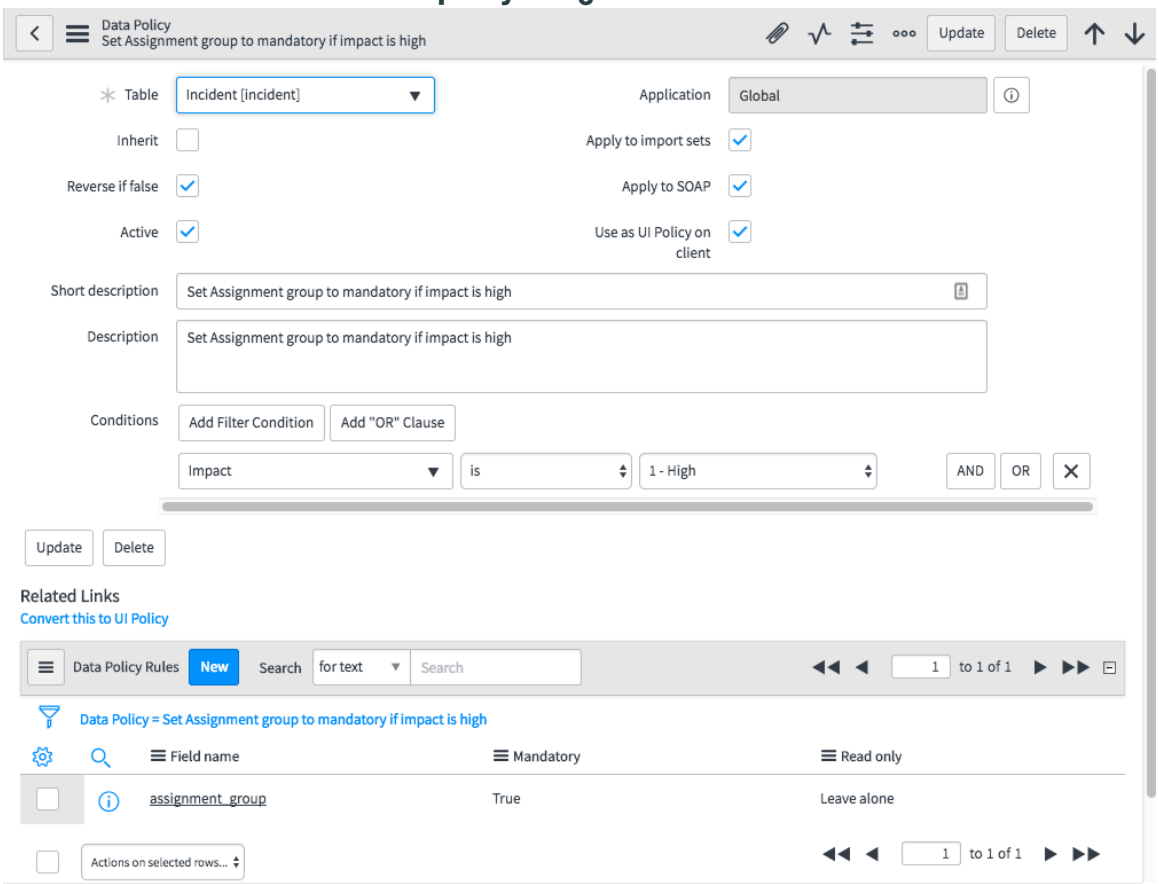
Before you begin

Role required: atf_test_admin

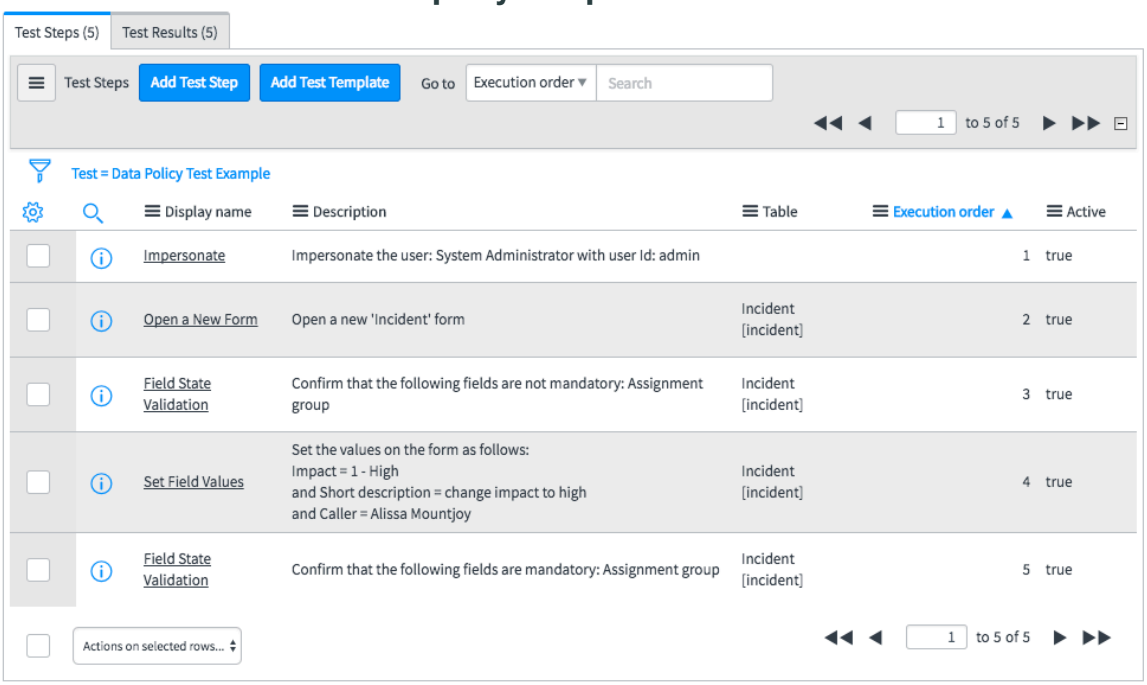
About this task

This example tests the data policy that sets the field **Assignment Group** to **mandatory** if **impact** is **high**.

Automated Test Framework: Data policy being tested



Automated Test Framework: Data policy example

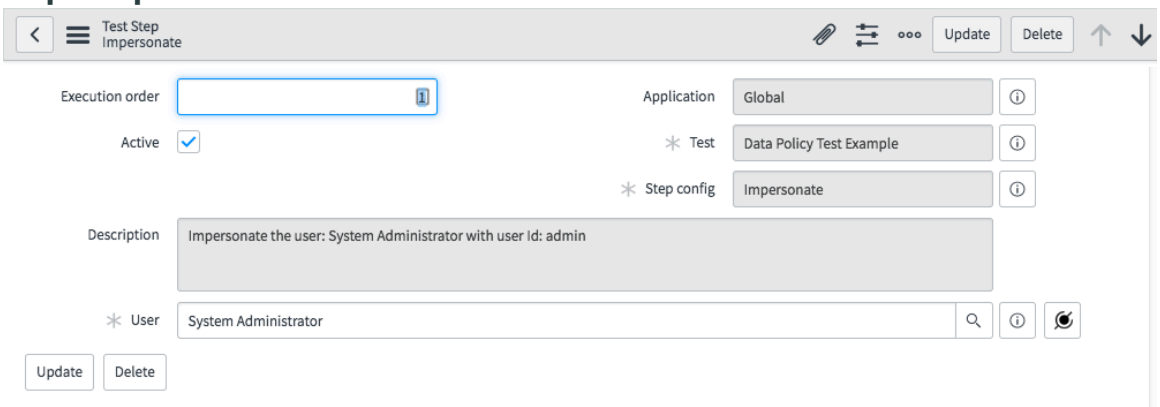


Procedure

1. Impersonate a user with the necessary permissions.
   In this example, the step impersonates an admin user.

Example

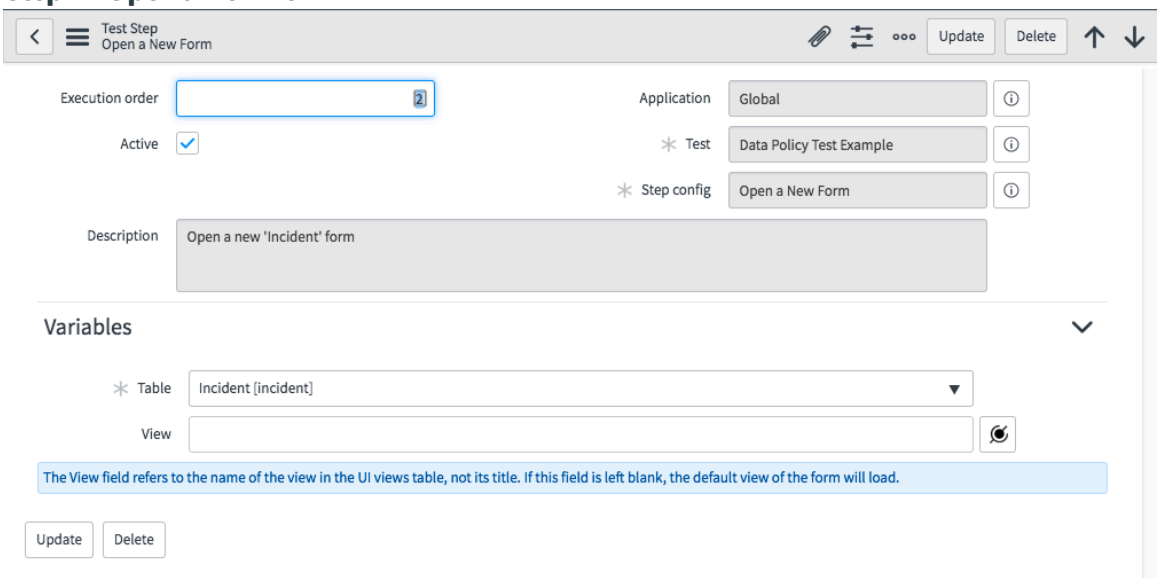
Step 1 - Impersonate



2. Open a form for the table to which this data policy applies.
   This example opens a new incident form.

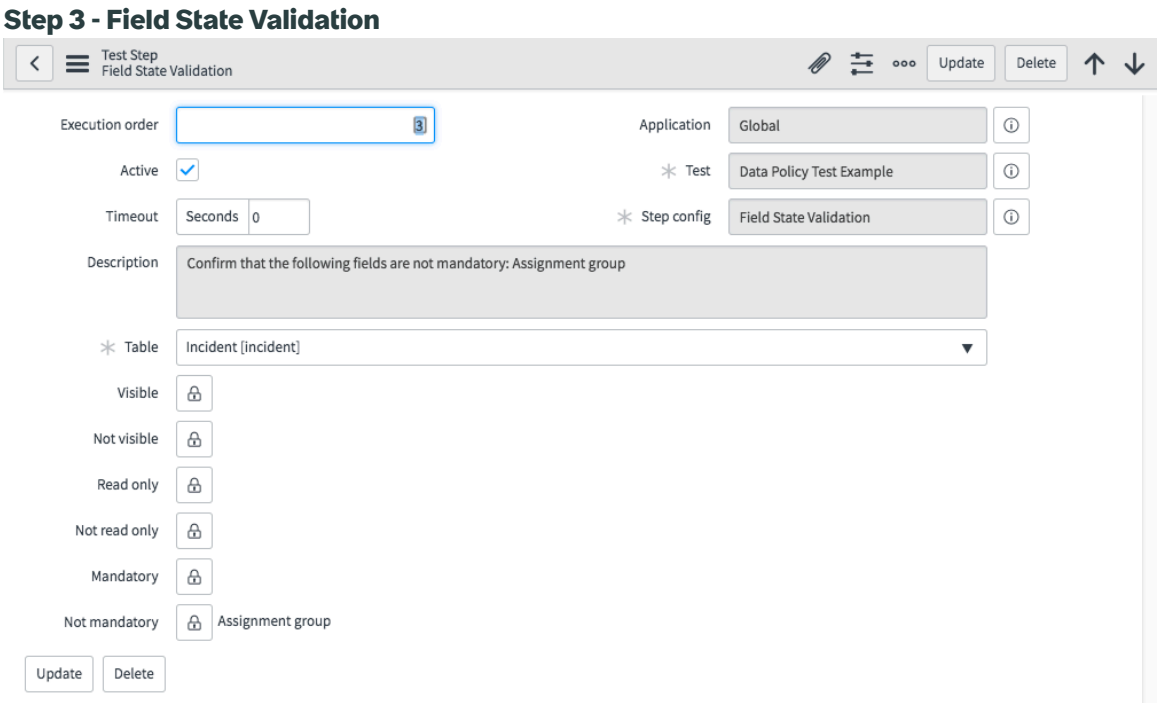
Example

Step 2 - Open a New Form



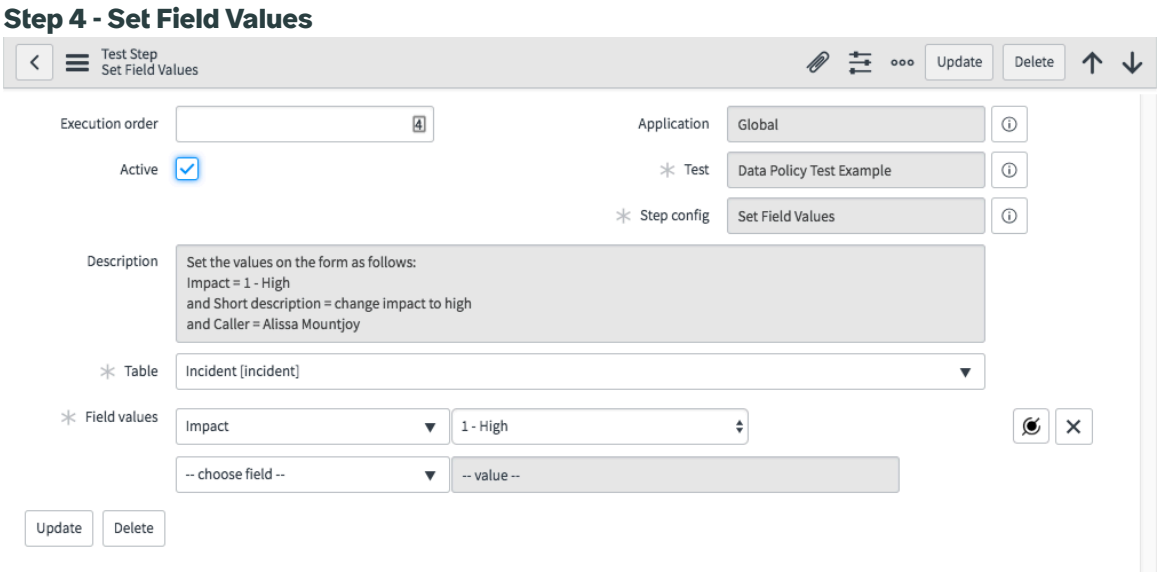
3. Check that the data policy has not yet been triggered.
   In this example, the step checks to confirm that **Assignment group** is not **mandatory**.

Example



4. If applicable, set the conditions that trigger the data policy.
   This example sets **Impact** to **High**.

Example

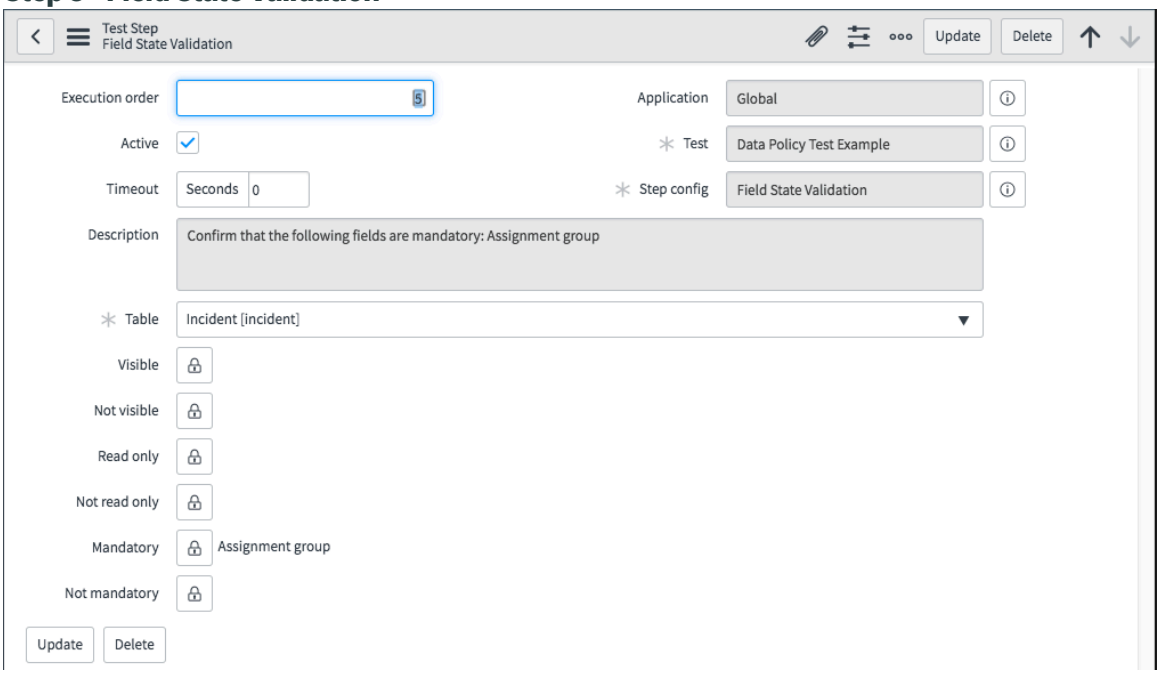


5. Validate that the data policy is enforced.

In this example, the test confirms that the data policy set **Assignment group** to **High** after the previous step set **Impact** to **High**.

Example

Step 5 - Field State Validation



Automated Test Framework use case: test a script include

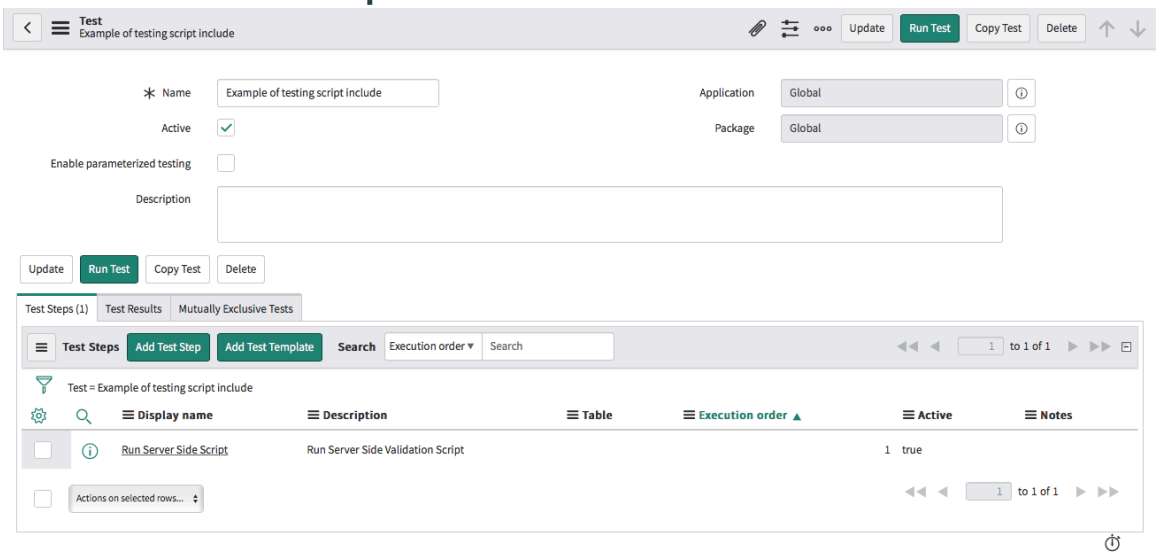
This use case illustrates testing a script include with the Automated Test Framework.

To test a script include with the Automated Test Framework, create a test that performs these steps:

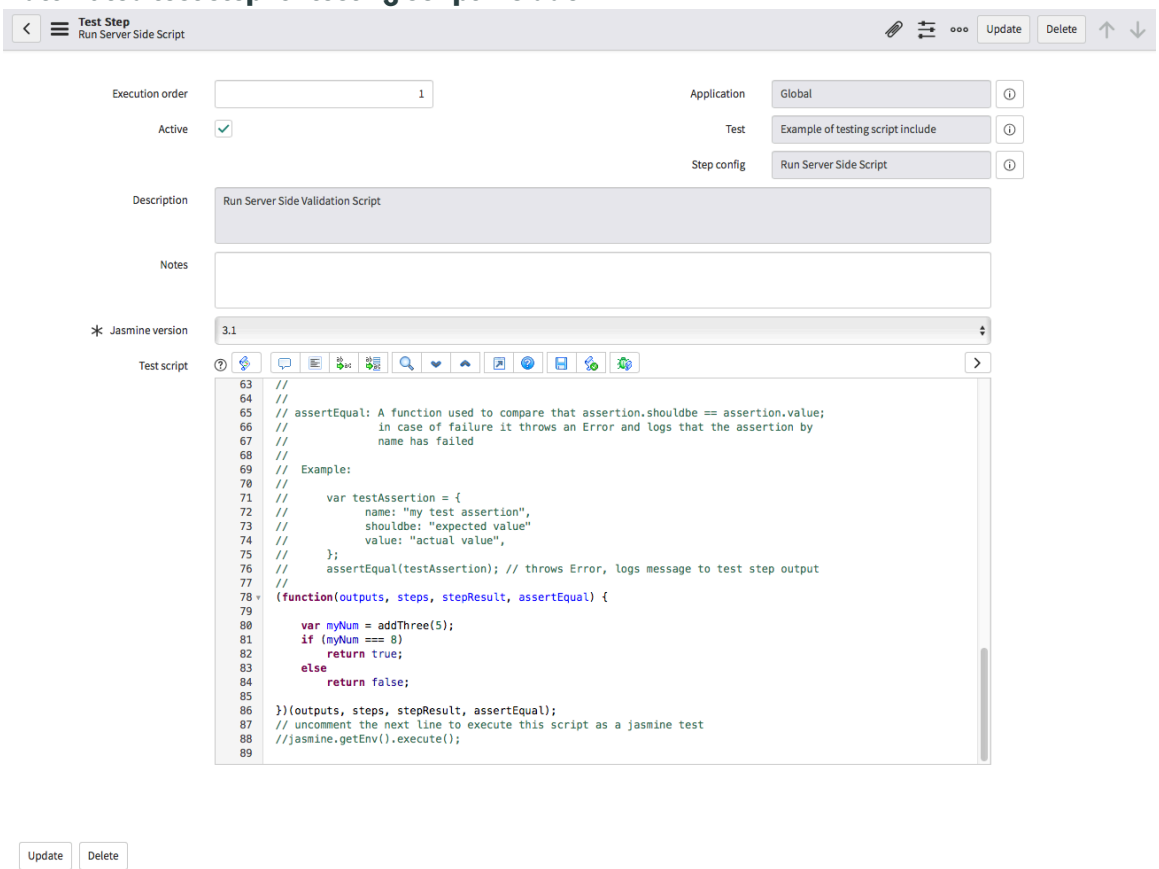
- a step that causes the system to execute the script include. Examples:
  - Run Server Side Script test step that calls the script include.
  - Form action step script – such as open a form, submit or form, or set a field value – that invokes the script include.
- A step that validates that the script include took the expected actions. The specific test step for this validation depends on what the script include is designed to do. Examples:
  - Field values validation, if the script sets a field value
  - Field state validation, if the script changes a field state
  - Record Query, if the script generates a new record

This example shows a test with one test step: Run Server Side Script. The script associated with this test step calls a script include that returns the value of its argument plus three. If the value returned from the script include is 8, the script include has worked as intended and the test passes.

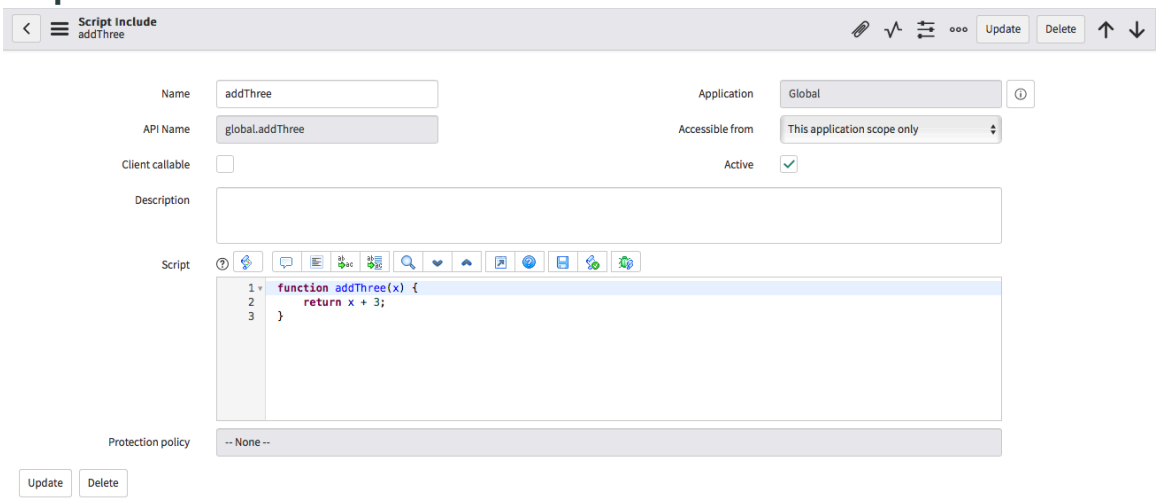
Automated Test to test a script include



Automated test step for testing script include



Script include to test with Automated Test Framework



Automated Test Framework use case: test a Service Catalog request

This use case illustrates testing a service catalog request with the Automated Test Framework.

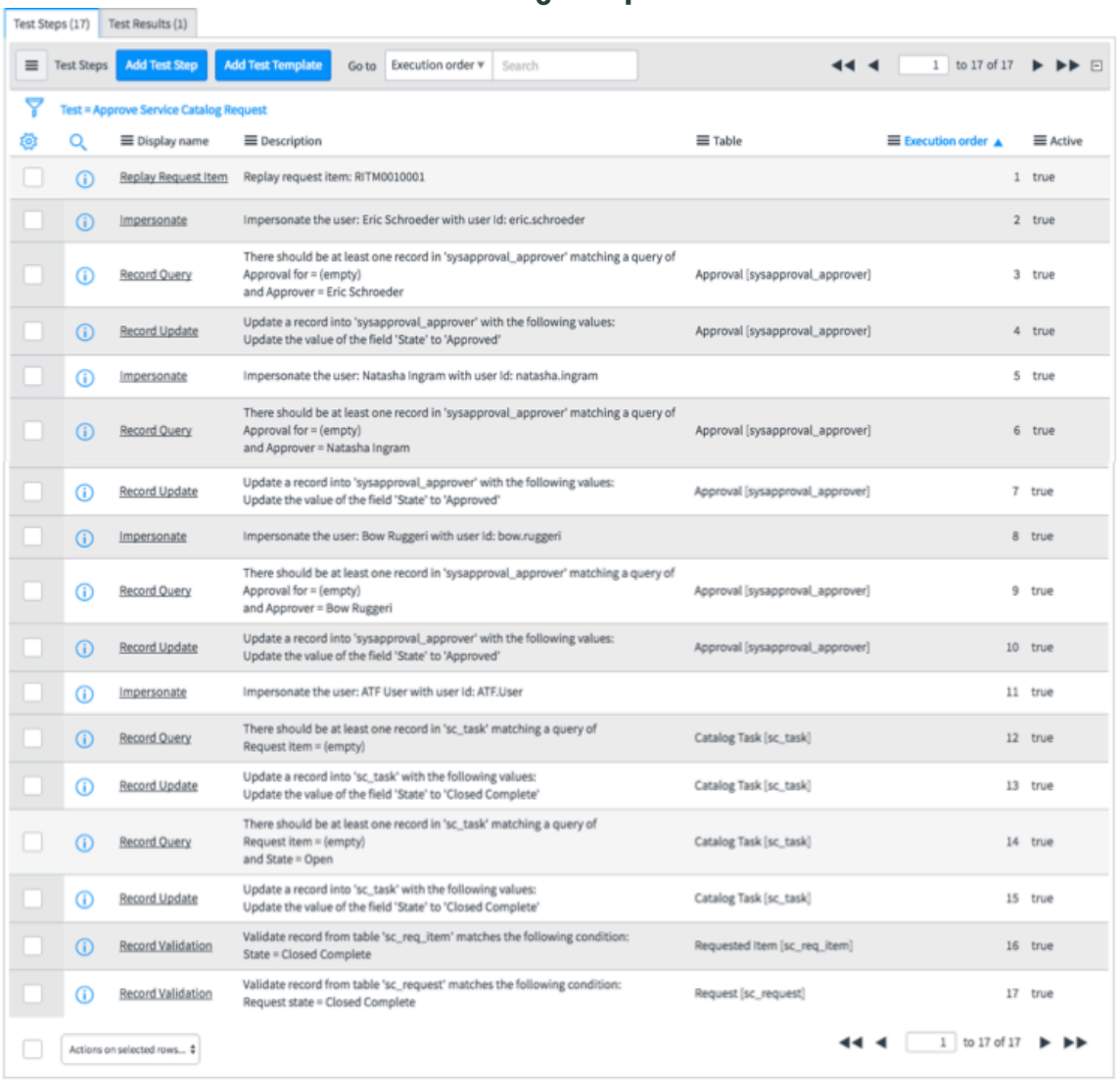
Before you begin

Role required: atf_test_admin

About this task

With the Replay Request Item test step, you can test the service catalog ordering process once a request exists and has a record in the request item table. In the Xanadu release, you cannot create an automated test for the process by which the user creates a new request.

Automated Test Framework: Service catalog example



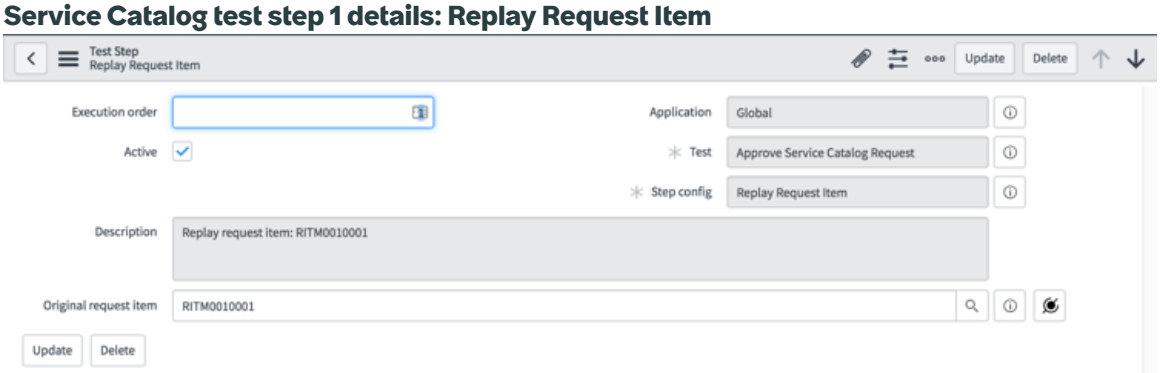
Procedure

1. Replay an existing service catalog request item.

   This test step inserts a new record in the [sc_request] table for the catalog request item RITM0010001.

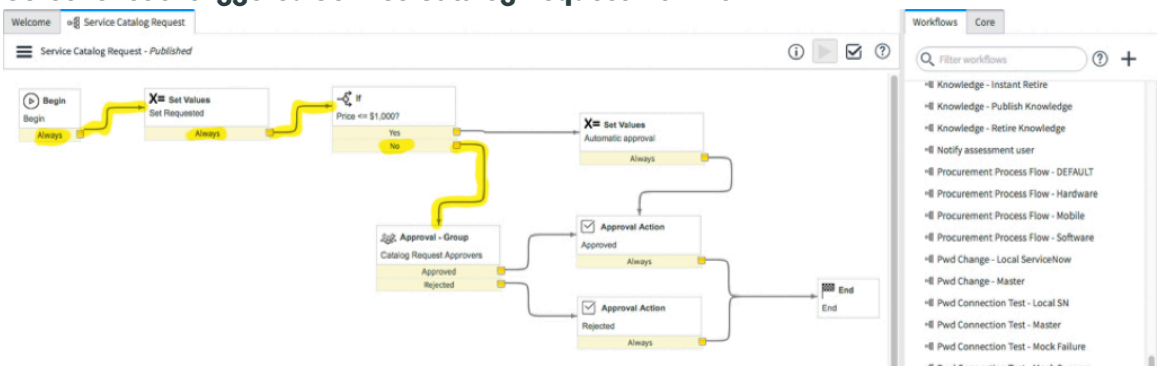
Example

Service Catalog test step 1 details: Replay Request Item



This insertion triggers the Service Catalog Request workflow, which checks the price of the item, determines that it exceeds $1000.00, and therefore generates approval records for users belonging to the Catalog Request Approvals group. In this example, only one user– Eric Schroeder – belongs to this group.

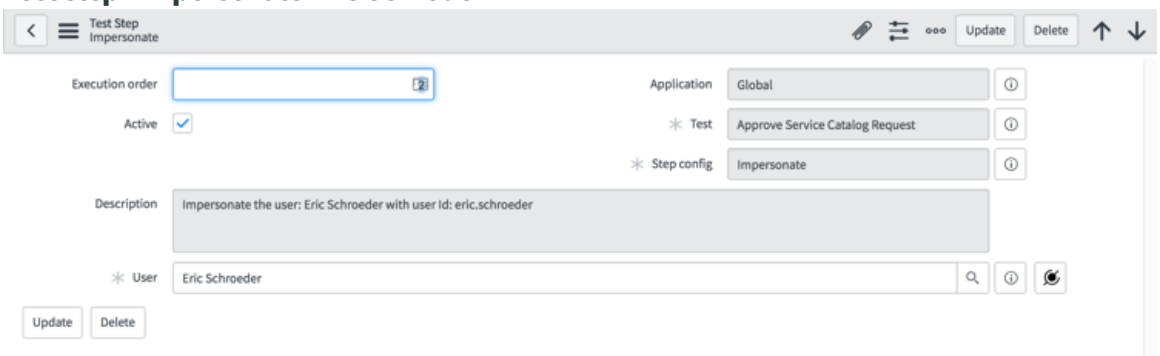
Screenshot of triggered Service Catalog Request workflow



2. Impersonate Eric Schroeder, the user who needs to approve this Service Catalog Request.

Example

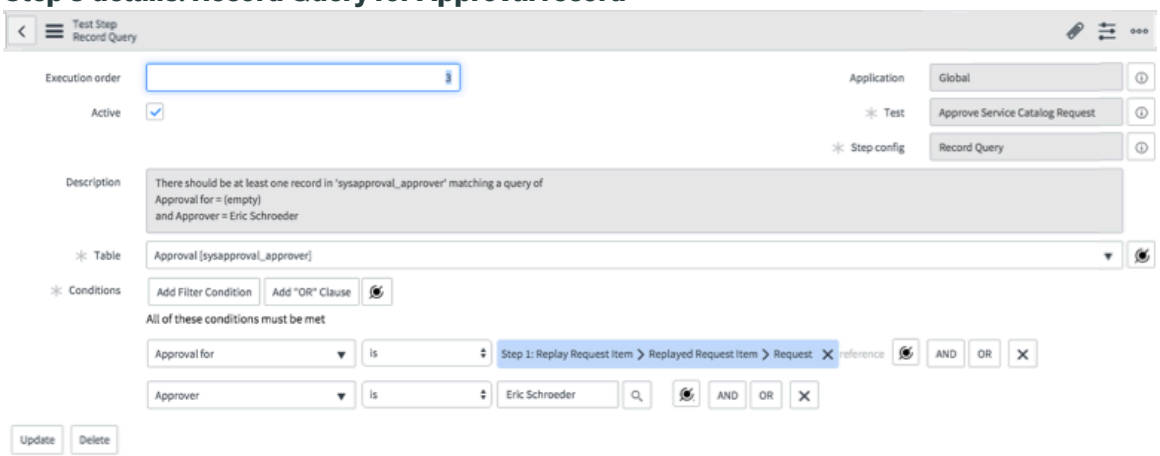
Test step - Impersonate Eric Schroder



3. Verify that the system created an approval record for Eric Schroeder and this request. Note that for the **Approval for** field, you assign the output value from Step 1.

Example

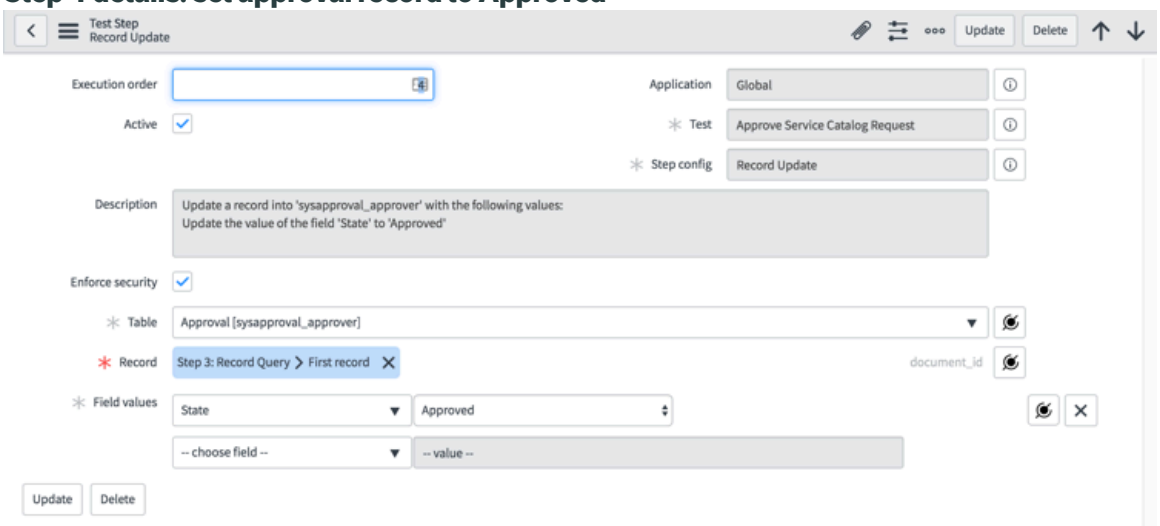
Step 3 details: Record Query for Approval record



4. Set the state of this approval record to Approved.

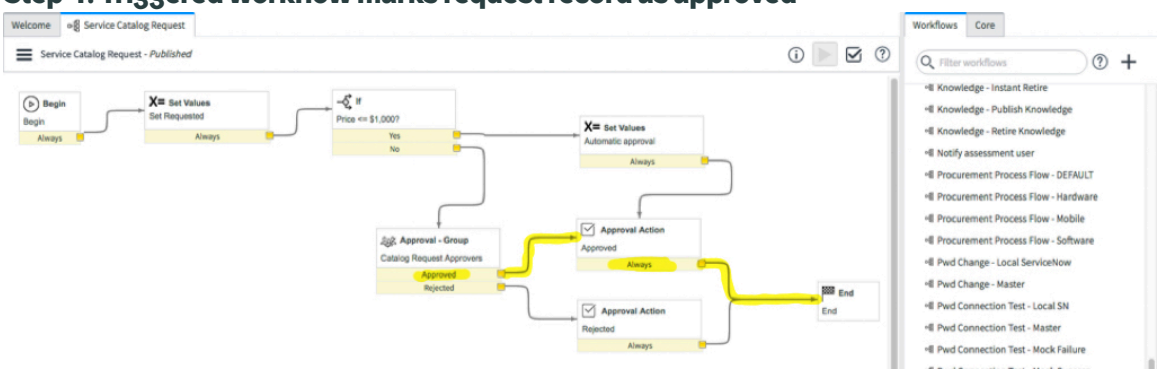
Example

Step 4 details: set approval record to Approved



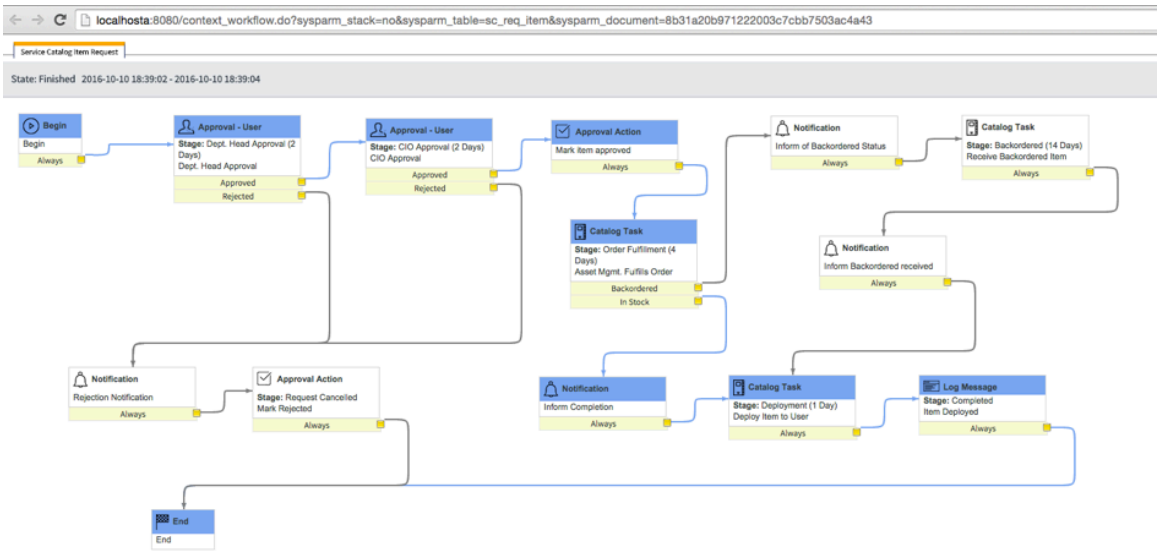
The Service Catalog Request workflow sees that all required approval records have the state of Approved and transitions to the Approval Action which marks the request record [sc_request] as Approved.

Step 4: Triggered workflow marks request record as approved



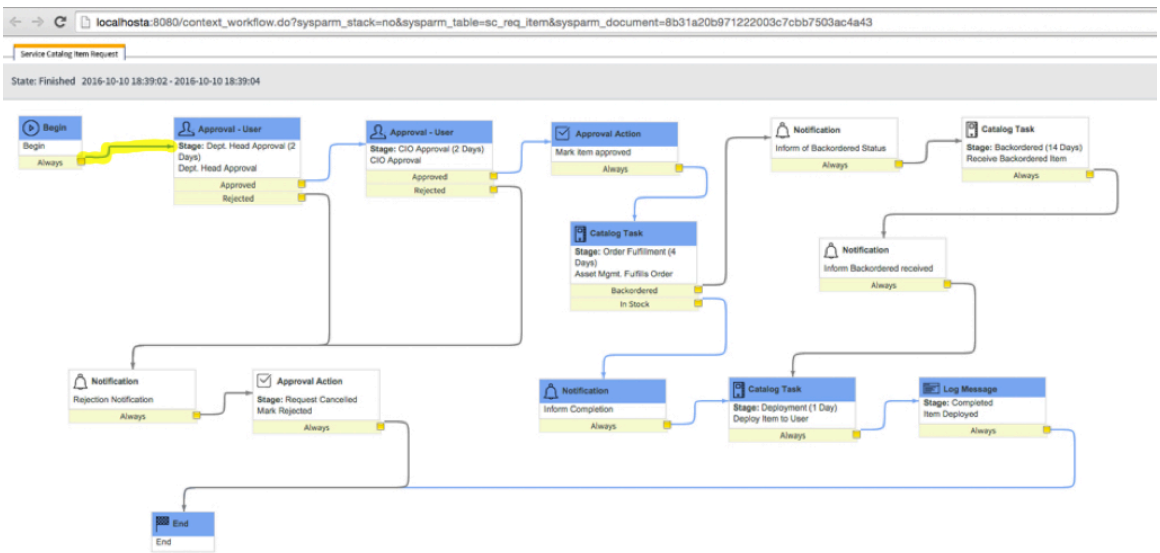
When the record in [sc_request] changes to the Approved state, an associated business rule generates request items [sc_request_item] for each item in the request. In this example, the request contains only one item, so the business rule inserts one record into the [sc_request_item] table. This insertion triggers the Service Catalog Item Request workflow.

Service Catalog Item workflow



The first activity in the Service Catalog Item Request workflow generates an approval record for the head of the department in which the requesting user works. In this example, the department head is Natasha Ingram.

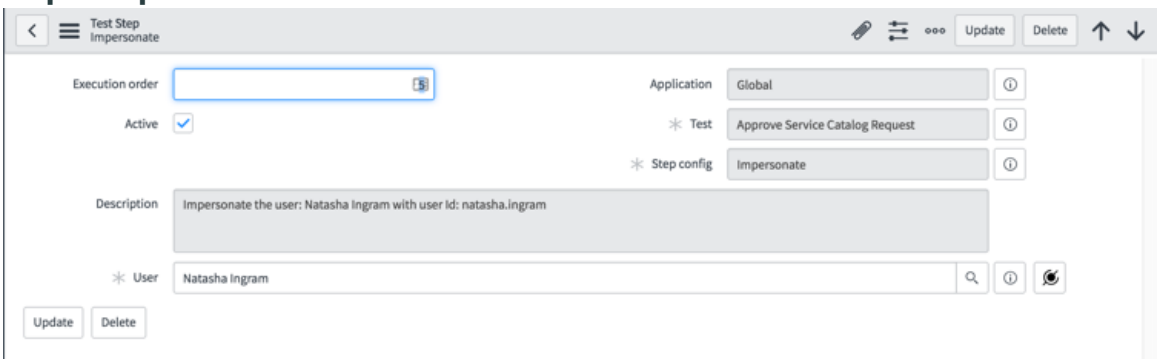
Service Catalog Item workflow: step 4



5. The workflow does not continue until the department head approves it, so the next test step impersonates Natasha Ingram.

Example

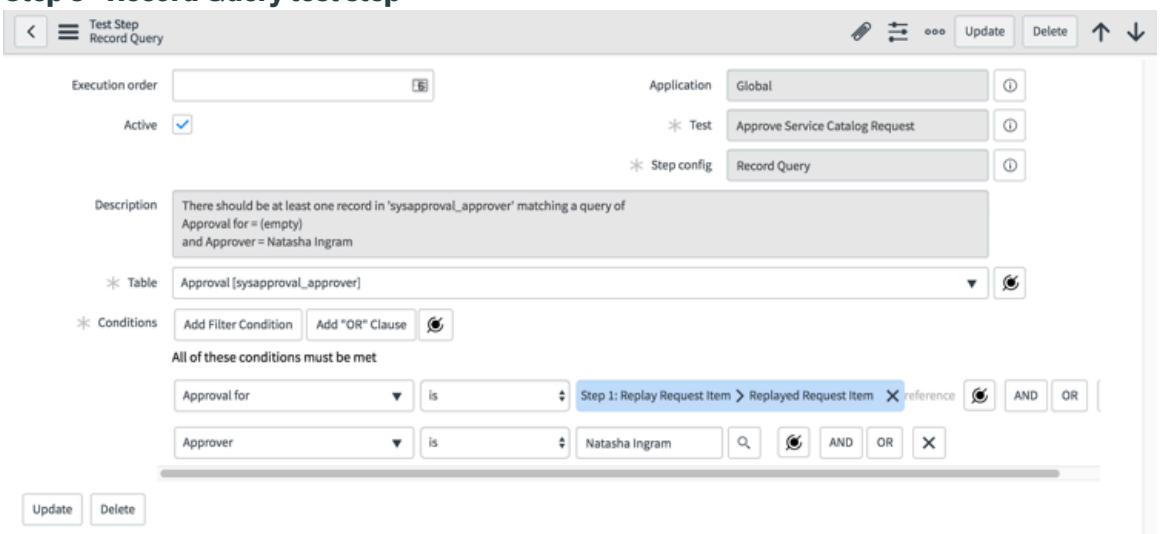
Step 5 - Impersonate User



6. Obtain the sys_id for the new approval record with the Record Query step.
   Note that Record Query creates an output variable with the sys_id of the first record returned from the query.

Example

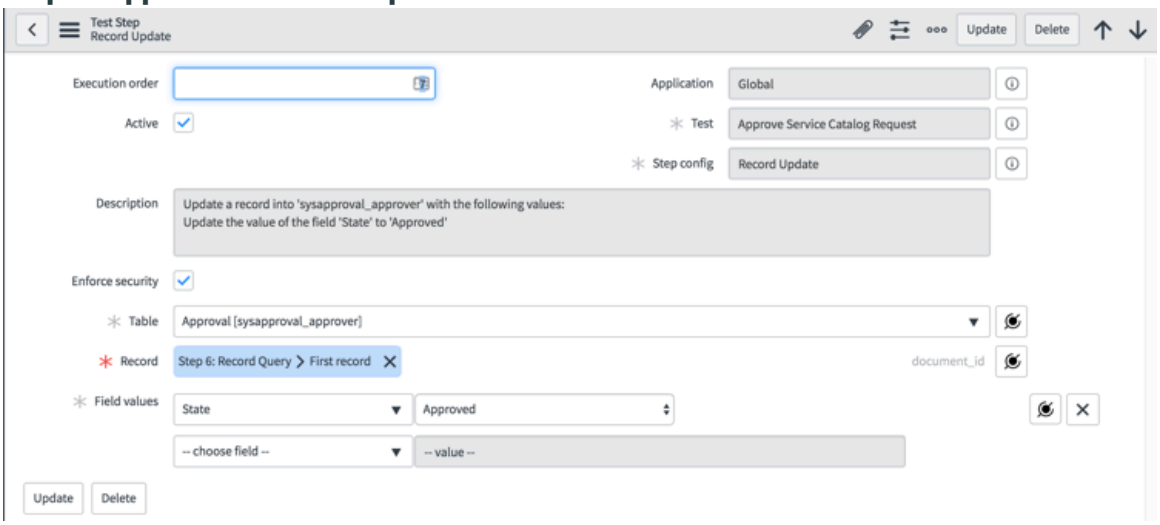
Step 6 - Record Query test step



7. Set the approval record to Approved.

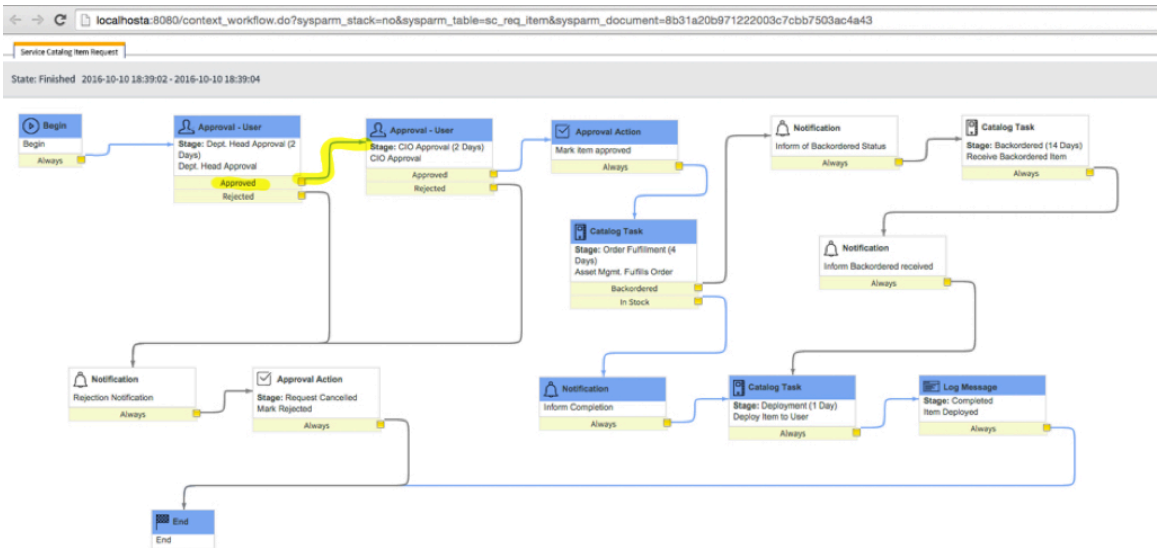
Example

Step 7 - Approval User test step



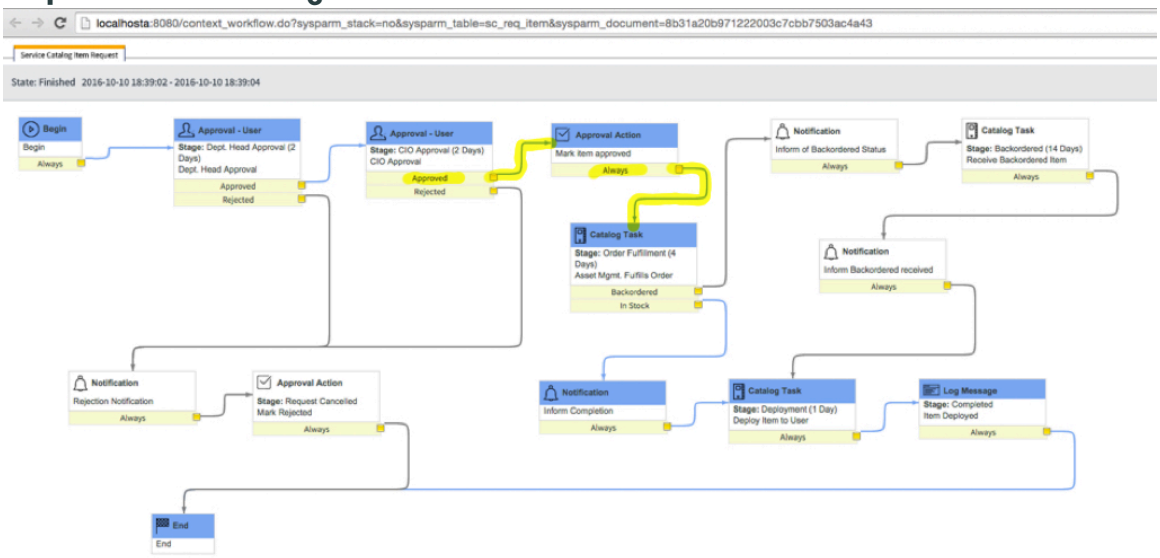
Note how Step 7 refers to the *`First record`* output variable from Step 6 to specify which record to approve. When the record is approved, the workflow transitions to the next Approval - User activity, which generates an approval record for the CIO. In this example, the CIO is Bow Ruggeri.

Step 7 details - Service Catalog Item workflow



8. Impersonate Bow Ruggeri.
9. Obtain the sys_id for the approval record for Bow Ruggeri.
10. Set the approval record to Approved.
    When the record is approved, the workflow transitions to the Approval Action activity which sets the record for this item in the [sc_request_item] table to *`Approved`*. The workflow transitions to the Catalog Task activity labelled Asset Mgmt. Fulfills Order. This Catalog Task activity generates a new record in the [sc_task] table that instructs a user in the Fulfillment group to order the item.

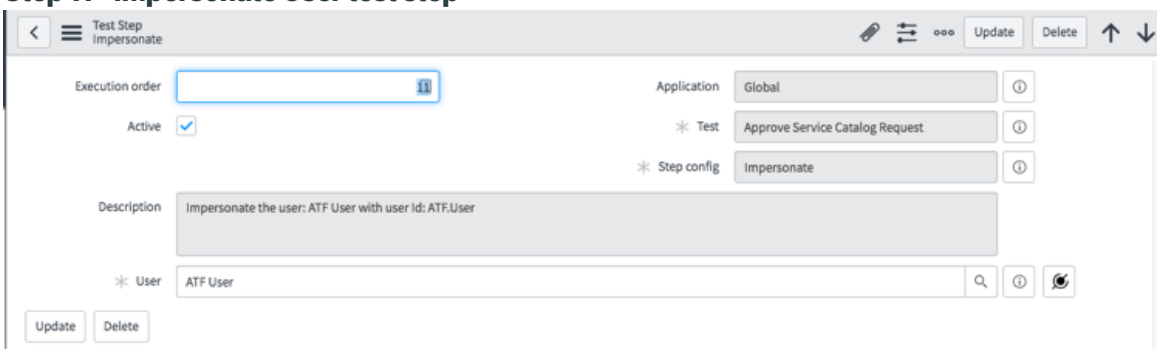
Step 10 - Service Catalog Item workflow



11. Impersonate a user in the Fulfillment group, in this example ATF.User.

Example

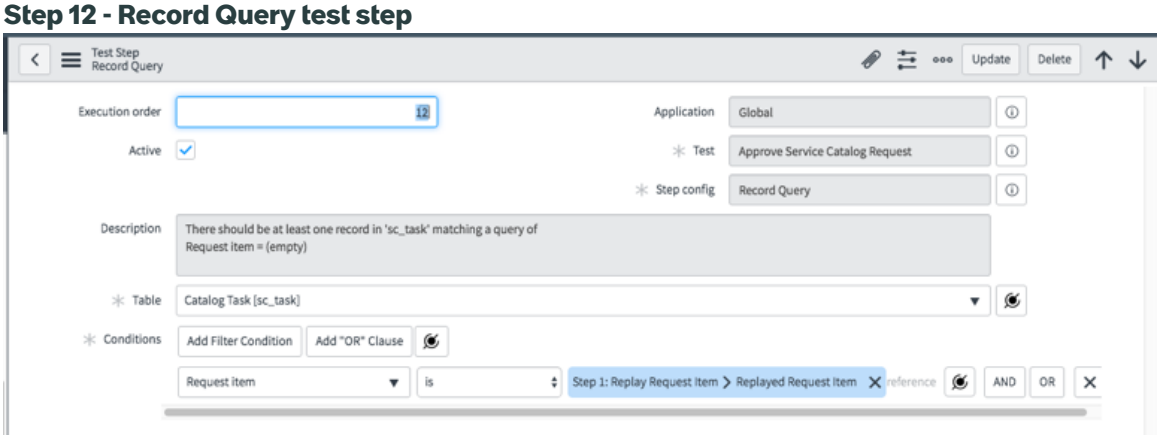
Step 11 - Impersonate User test step



12. Obtain the sys_id for the new catalog task with the Record Query step.
Note that Record Query creates an output variable with the sys_id of the first record returned from the query.

Example

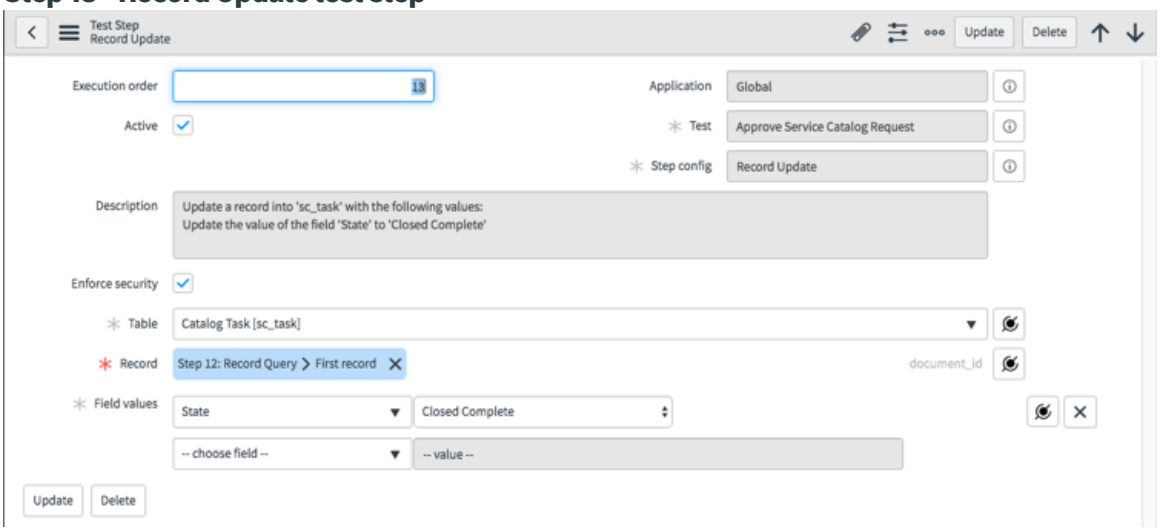
Step 12 - Record Query test step



13. Mark the [sc_task] record as *`Closed Complete`*.

Example

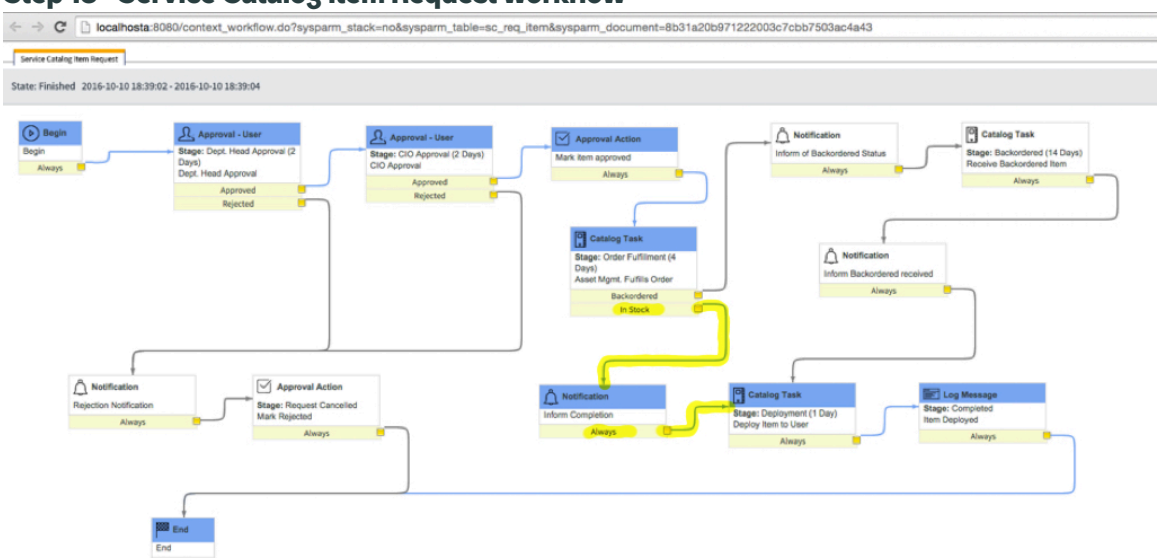
Step 13 - Record Update test step



Note how Step 13 uses the *`First record`* output variable from Step 12 to specify which record to mark as *`Closed Complete`*.

When the record is marked *`Closed Complete`*, the workflow exits the Catalog Task activity along the In Stock exit path.

Step 13 - Service Catalog Item Request workflow

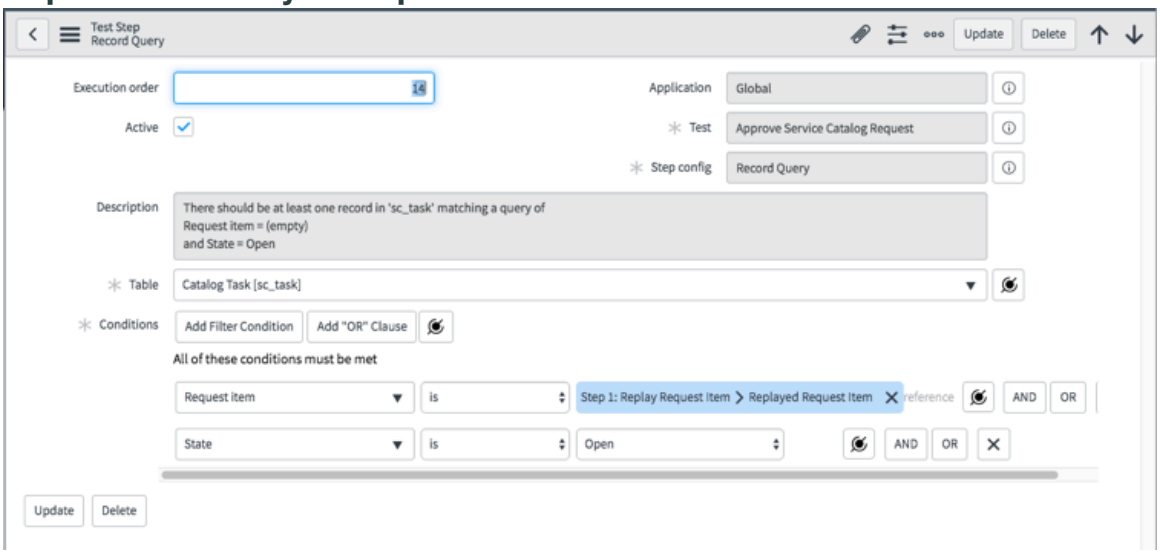


The workflow transitions to the Notification activity, then to the Catalog Task activity labelled Deploy Item to User.The Deploy Item to User Catalog Task activity inserts a new record into the [sc_task] table that instructs a user in the Deployment group to deliver the item.

14. Obtain the sys_id for the new catalog task with the Record Query step.
Note that Record Query returns an output variable with the sys_id of the first record returned from the query.

Example

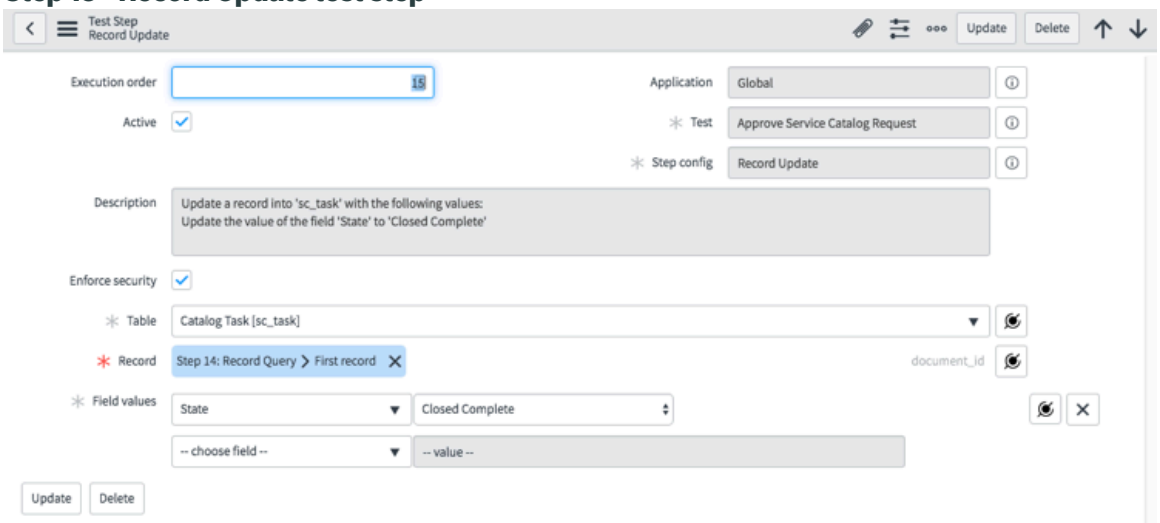
Step 14 - Record Query test step



15. Mark the [sc_task] record as Closed Complete.
    Note how Step 15 uses the *`First record`* output variable from Step 14 to specify which record to mark as Closed Complete.

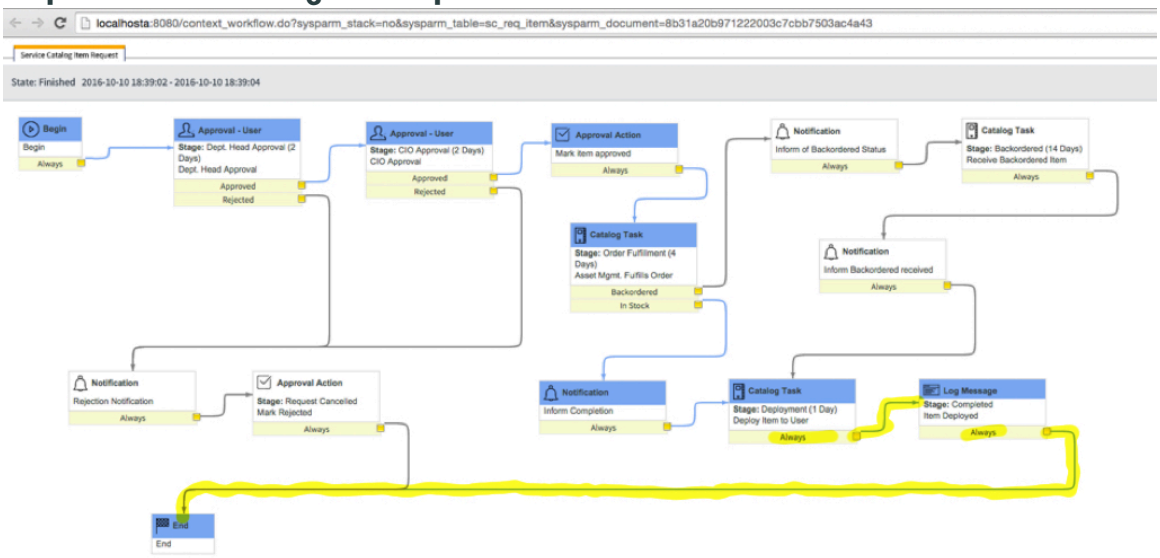
Example

Step 15 - Record Update test step



When the record is marked Closed Complete, the workflow exits the Catalog Task activity, logs a message, and exits.

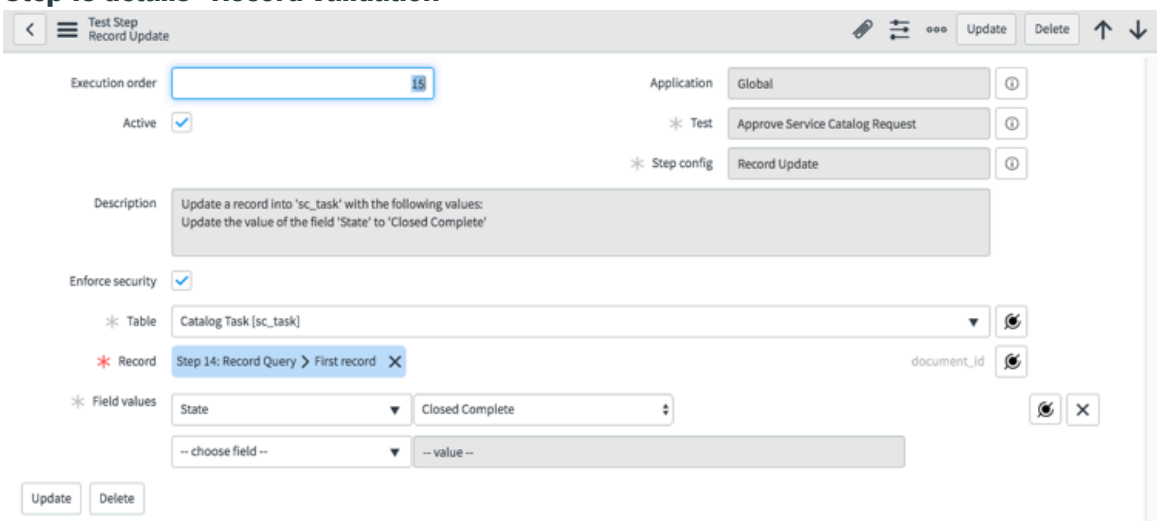
Step 15 - Service Catalog Item Request workflow



16. Verify that the request item in [sc_request_item] has the state Closed Complete.

Example

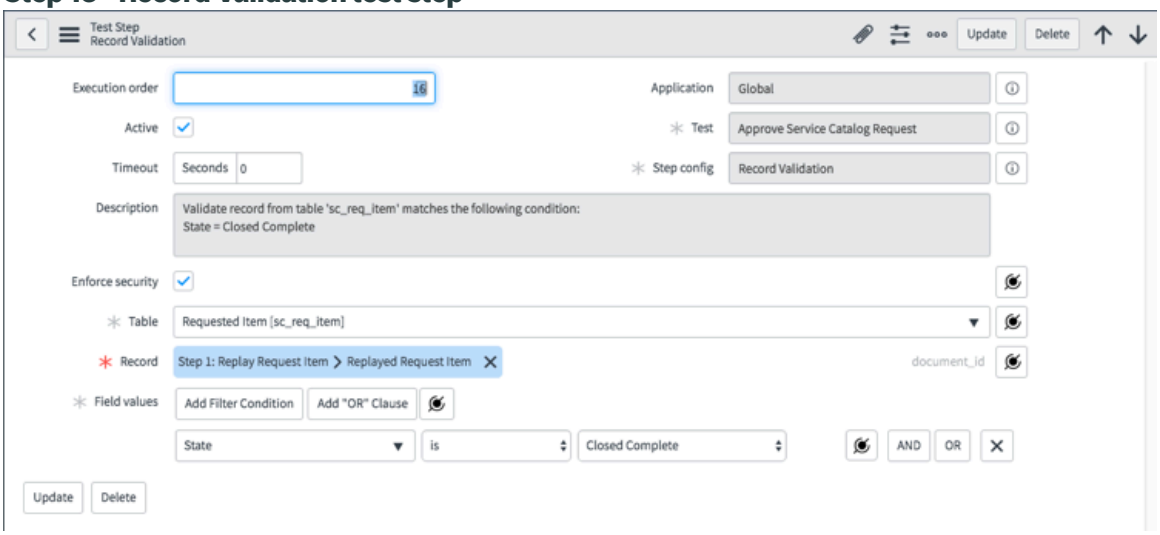
Step 15 details - Record Validation



17. Verify that the request [sc_request] has the state Closed Complete.

Example

Step 16 - Record Validation test step



Related topics

Replay Request Item

Automated Test Framework use case: retrieve an incident using REST-Inbound

The **Get Newly Created Resource via REST API Test** test is provided with the Automated Test Framework, and uses the REST - Inbound and assert steps.

Before you begin

Review the REST APIs and the Send REST Request- Inbound- REST API Explorer configuration step information before creating this test.

Role required: atf_ws_designer

About this task

This test creates an incident record, uses a REST- Inbound step to retrieve the record, and then uses assert steps to determine whether the request was successful.
You can use the Send REST Request- Inbound REST API Explorer step to build and test the request, or you can manually create the request using the Send REST Request- Inbound step. In either case, you must specify the basic authentication information on the Send REST Request- Inbound step form.

When creating your test, start with your REST API and determine what behavior you want to validate. You can then determine what test data to create. You can use other test step configuration categories to create, update, or delete records, and then use a Send REST Inbound step to retrieve, update, or delete the test data. Conversely, you can use the Send REST Inbound step to create records, and then use other test step configuration categories to validate that the records were created correctly.

This task steps you through creating the **Get Newly Created Resource via REST API Test** test.

Procedure

1. Create a test.

   a. Navigate to **Automated Test Framework** > **Tests**, click **New**.
   The **Test** new record form is shown.

   b. Enter a test name and a description, and click **Submit**.

   c. Click the test created in the previous step.
   The **Test** form is shown.

2. Create test data.

   a. Click the **Add Test Step** button.
   The **Add Test Step** form is shown.

   b. Click a test configuration category, and then click a test configuration.
   To replicate the **Get Newly Created Resource via REST API Test** test, click **Server** > **Record Insert**, and then **Next**.
   The test configuration form you selected is shown.

   c. Fill in the information needed for the test configuration you selected.
   To replicate the **Get Newly Created Resource via REST API Test** test, on the **Record Insert** form, specify the incident table, and add a value for the **Short description** field, and click **Submit**.

3. Create the REST request.

   a. Click the **Add Test Step** button.
   The **Add Test Step** form is shown.

   b. Click a test configuration category, and then click a test configuration.
   To replicate the **Get Newly Created Resource via REST API Test** test, click **REST** > **Send REST Request - Inbound REST API Explorer**, and then **Next**.
   The **REST API Explorer** is shown.

   c. Fill in the information needed for the REST request, and click **Send**.
   To replicate the **Get Newly Created Resource via REST API Test** test, fill in the fields.

Name	Value
Namespace	now
API Name	Table API
API Version	latest
tableName	incident

   When you do not specify the **Basic authentication**, the REST API Explorer uses your credentials.

   The **Create Automated Test Step** is shown after you click **Send**.

d. When ready, click **Create Automated Test Step**.
The Send REST Request - Inbound test step is created.

e. Click the Send REST Request - Inbound step, and in the **Basic authentication** field, specify a basic auth configuration.
If no configurations are available, you can create a basic auth configuration by clicking **New** on the **Basic Auth Configurations** form.

f. To replicate the **Get Newly Created Resource via REST API Test** test, in the **Path** field, click the contextual search button, and then click **Record Insert** > **Record**.

g. Click **Update**.

4. Create assert steps to verify the REST response.

a. Click the **Add Test Step** button.
The **Add Test Step** form is shown.

b. Click a test configuration category, and then click a test configuration.
To replicate the **Get Newly Created Resource via REST API Test** test, click **REST** > **Assert Status Code**, and then **Next**.
The **Assert Status Code** form is shown.

c. Enter the information needed for the form, and click **Submit**.
To replicate the **Get Newly Created Resource via REST API Test** test, in the **Operation** field, select **is** and in the **Status Code** field, enter **200**.

To replicate the **Get Newly Created Resource via REST API Test** test, repeat this step to create **Assert Response JSON Payload is Valid**, and **Assert JSON Response Payload Element** test steps.

Quick start tests

Copy and customize quick start tests provided by the ServiceNow AI Platform® to validate that your instance works after you make any configuration changes. For example, if you apply an upgrade or develop an application.

The tests can only produce a pass result when you run them with the default demo data that's provided with the application or feature plugin.

Note:
If your QST fails, it can be that the test doesn't match your instance customizations. Use the **Notes** field for each test step to update the test to pass with your data. If you have customized the business process associated with a test, revise the test to match your customizations.

To apply a quick start test to your instance-specific data, copy the quick start test and add your custom data.

Note:
You can copy either an individual quick start test or the entire quick start tests suite.

See Getting started with quick start tests for more information.

Activation

Each application or feature has its own plugin activation requirements for enabling quick start tests. See Available quick start tests by application or feature for activation information.

Managing copies

When you copy a test, Automated Test Framework populates the **Copied from** field with the name of the copied test. When an upgrade changes, a quick start test Automated Test Framework notifies test designers about the change in a notification on the test form. Test designers can revert the copied test to the upgraded version with the **Revert Copy to Base System** UI action.

Tip:
If a QST is updated, the previously copied versions of the QST does not have the recent changes. Review the old copies using **Copies to Review** for all the tests that have an associated warning message.

Available quick start tests by application or feature

Validate that your instance still works after you make any configuration change such as apply an upgrade or develop an application. Copy and customize the ServiceNow -provided quick start tests to pass when using your instance-specific data.

Danger:
By default, the system property that is used to run automated tests is disabled to prevent you from accidentally running these tests on a production system. To avoid data corruption or an outage, run tests only on development, test, and other non-production instances. See Enable or disable executing Automated Test Framework tests.

Agile Development 2.0

Agile Development 2.0 quick start tests require activating the Agile Development 2.0 plugin (com.snc.sdlc.agile.2.0) and the Agile Development 2.0 - ATF Tests plugin (com.snc.sdlc.agile.2.0.atf).

Agile Development 2.0 quick start test suite

Test	Description	Release version
Verify that global rank is populated when a story is created	Verify the global rank of a story after creation.	Madrid
Verify that closing a sprint with active stories is prevented	Verify that a sprint with active stories cannot be closed.	Madrid
Verify that sprints cannot overlap in the same group	Verify that sprints in the same group do not overlap.	Madrid
Verify that sprint points are updated	Verify that changes to stories produce accurate sprint point totals.	Madrid
Verify that only one sprint in a group can have the current state	Verify sprint statuses.	Madrid

Agile Development 2.0 quick start test suite (continued)

Test	Description	Release version
Verify sprint end date is after the sprint start date	Verify sprint start and end dates.	Madrid
Verify that any update on story rolls up to Epic	Verify that adding, estimating, removing, deleting, updating, or cancelling a story updates the epic-level roll-ups correctly.	Orlando
Verify changes to the scope of a current sprint do not alter the value of the Total Committed Points	Verify that the value of Total Committed Points does not change with change in the scope of a sprint after its state is changed to Current.	Orlando
Verify active flag is set false when Agile Story state is changed to Completed/ Cancelled	Verify that active flag of an Agile story is set to the following: • False, if the state is changed to Completed or Cancelled • True, for all other states	Orlando
Verify that updating the team/ group capacity overrides the capacity on all the future sprints	Verify that any update to the **Group capacity** field of the assignment group results in the following changes to the **Group capacity** field of the various sprints associated with this assignment group: • For the sprints that are in the Draft and Planning state: ○ The group capacity is updated to the new value. ○ The **Group capacity** field is editable. • For the sprints in the Current, Complete, or Cancelled state: ○ The group capacity remains the old value. ○ The **Group capacity** field is read-only. For the sprints in the Draft or Planning state, you can individually edit the group capacity of the sprint anytime later. This would not change the group capacity of the	Paris

Agile Development 2.0 quick start test suite (continued)

Test	Description	Release version
	assignment group associated with this sprint.	
Verify create and edit functionality of an epic backlog on the scrum program board	• Verify that you can create an epic backlog for scrum programs from the Backlog tab of Agile Board. • Verify that you can update an existing epic backlog for scrum programs from the Backlog tab of Agile Board. • The epics listed in the backlog must belong to the selected epic backlog.	Quebec
Verify a Demand is converted to a scrum story	If the PPM Standard plugin (com.snc.financial_planning_pmo) is active, verify that a Demand can be converted to an Agile 2.0 story using the **Create Story** related link on the Demand form. For more information on how to create an Agile 2.0 story from a demand, see Create an artifact from a demand.	Quebec
Verify a Demand is converted to a scrum epic	If the PPM Standard plugin (com.snc.financial_planning_pmo) is active, verify that a Demand can be converted to an Agile 2.0 epic using the **Create Epic** related link on the Demand form. For more information on how to create an Agile 2.0 epic from a demand, see Create an artifact from a demand.	Quebec

To learn more about Agile Development 2.0, see Agile Development 2.0.

Enterprise Architecture (formerly Application Portfolio Management)

Enterprise Architecture quick start tests require enabling the Enterprise Architecture – ATF Tests plugin (com.snc.apm.atf).

Enterprise Architecture: Create Business application and capability test suite

Test	Description	Release version
Enterprise Architecture: Create Business Application	Verify the creation of an application category and then the creation of a business application with users having apm_user role.	Madrid
Enterprise Architecture: Create Business Capability	Verify the creation of a parent and child business capability and verify its field values.	Madrid
Enterprise Architecture: Test relating Business Service, Business Application, and Software Models	Verify the creation of a business application, business service, using the existing software model, and a relationship between them.	Orlando
Enterprise Architecture: Test for Indicator Score and Application Score generation	Verify the creation of indicator, scoring profile, and generation of indicator scores and application scores.	Paris
Enterprise Architecture: Business Application with Information Object and Data Domain	Verify the creation of business application, information object, and addition of the CRUD operations in relation attributes.	Quebec

To learn more about Enterprise Architecture (formerly Application Portfolio Management), see Application Portfolio Management.

Assessments and Surveys

Assessments and Surveys quick start tests require activating the Automated Test Framework for Survey plugin (com.glide.automated_testing_impl. Survey).

Test Suite for Survey

Name	Description	Release version
Survey: Basic Platform Based Test	Create a survey using Platform UI actions.	Madrid
Survey: Platform test for Dynamic Validation	Validate a survey dynamically.	Madrid
Survey: Clone Action	Clone a survey and validate the records of the original survey and the cloned survey.	New York
Survey: Question Bank Flow	Verify the addition of a question bank to a survey.	New York
Survey: Survey Creator Work Flow	Survey creator can create a survey and assign to the user who can take the survey and submit it.	New York

Test Suite for Survey (continued)

Name	Description	Release version
Assessment: Assign assessment to assessor	Assign an assessment to an assessor and verify that the instance is created for the assessor.	Orlando
Assessment: Create assessment as survey creator	Create an assignment as a survey creator.	Orlando
Assessment: Scale and Template support	Create, publish, and assign the assessment. Then take the assessment.	Rome
Assessment: Basic test flow	Verify the basic flow of an assessment.	Orlando

To learn more about Assessments and Surveys, see Service administration.

Change Management

Change Management quick start tests require activating the Change Management - ATF Tests plugin (com.snc.change_management.atf).

CHG: Emergency Type Change Request test suite

Test	Description	Release version
Emergency Type Change Request workflow	Process an emergency change request from new to closed.	Madrid
On Hold for Emergency type Change Request	Validate the approval state of an on-hold emergency change request.	Madrid
Copy Change For Emergency type Change Request	Validate the state of a copied emergency change request.	Madrid
Reject By Approver for Emergency type	Validate the state of a rejected emergency change request.	Madrid
Revert to new for emergency type	Validate the state of an emergency change request after using the Revert to new UI action.	Madrid
Convert Emergency to Normal type	Validate the conversion of an emergency change request to a normal change request.	Madrid
Cancel Change Request For Emergency Type	Validate the state of a canceled emergency change request.	Madrid

CHG: Normal Type Change Request test suite

Test	Description	Release version
Normal Type Change Request Workflow	Process a normal change request from new to closed.	Madrid
Convert Normal to Emergency type	Validate the conversion of a normal change request to an emergency change request.	Madrid
Copy change on Normal Change Request	Validate the state of a copied normal change request.	Madrid
On hold for Normal type Change Request	Validate the approval state of an on-hold normal change request.	Madrid
State validation when Reject Normal type Change request by Approver.	Validate the state of a rejected normal change request.	Madrid
Revert to New Functionality for Normal Type Change Request	Validate the state of a normal change request after using the Revert to new UI action.	Madrid
Cancel Change Request For Normal type	Validate the state of a canceled normal change request.	Madrid

CHG: Standard Change Proposal test suite

Test	Description	Release version
Standard Change Proposal	Determine whether a user can successfully perform standard change proposal creation, approval, and template publishing processes.	Madrid

CHG: Standard Type Change Request test suite

Test	Description	Release version
Standard Change Request Workflow	Process a standard change request from new to closed.	Madrid
Convert Standard to Emergency Change Request	Validate the conversion of a standard change request to an emergency change request.	Madrid
Convert Standard to Normal Change Request	Validate the conversion of a standard change request to a normal change request.	Madrid

CHG: Unauthorized Change Request and Outage test suite

Test	Description	Release version
Unauthorized change request Workflow	Process a unauthorized change request from new to closed.	Orlando
Create Outage of type planned outage from change request	Validate the creation of an outage of type planned outage from a change request.	Orlando

CHG: Unauthorized Change Request and Outage test suite (continued)

Test	Description	Release version
Create Outage of type outage from change request	Validate the creation of an outage of type outage from a change request.	Orlando

CHG: Risk Conditions with Best practice plugin test suite

Test	Description	Release version
Moderate Risk with UI Action Property	Process a moderate risk with UI action property.	Orlando
Low Risk with UI Action Property	Process a low risk with UI action property.	Orlando
Leave Alone Risk with UI Action Property	Process a leave alone risk with UI action property.	Orlando
High Risk with UI Action Property	Process a high risk with UI action property.	Orlando
High Risk with Business Rule Property	Process a high risk with Business rule property.	Orlando
Moderate Risk with Business Rule property	Process a moderate risk with Business rule property.	Orlando

CHG: Change Request against Conflict Sources test suite

Test	Description	Release version
Change against Blackout window	Process a change request against a blackout window.	Orlando
Check conflicts for CI Already Scheduled	Validate the conflicts for CI already scheduled.	Orlando
Change against Conflict Sources	Validate the change request against conflict sources.	Orlando

CHG: Change Schedule Definition and Sharing test suite

Test	Description	Release version
Change Schedules Definitions on New	Process the creation of change schedules definitions from New button on Change Schedules landing page.	Orlando
Share Panel On Change Schedules Definition	Validate the share panel on change schedules definition.	Orlando
Share Change Schedule Definition	Validate the sharing of change schedules definitions.	Orlando
Create Standard Change via Service Portal	Create Standard Change from Service Portal	Xanadu

To learn more about Change Management, see Change Management.

Cloud Provisioning and Governance

Cloud Provisioning and Governance quick start tests require activating the following Cloud Provisioning and Governance plugins:

- Cloud Provisioning and Governance (com.snc.cloud.mgmt)
- Domain Support- Domain Extensions Installer plugin (com.glide.domain.msp_extensions.installer)
- Service Catalog- Domain Separation plugin (glideapp.servicecatalog.domain_separation)

Cloud Provisioning and Governance: Azure test suite

Test	Description	Release version
CMP: Add Azure Credentials	Add Azure credentials by inserting the fields into the Credentials table.	Orlando
CMP: Azure Service Account	Add Azure service account by inserting the fields into the service account table.	Orlando
CMP: Update datacenter type	Update the Cloud Service account with the datacenter type.	Orlando
CMP: Creating Resource Group	Creating Resource Group to provision Azure stack via ARM Template.	Orlando
CMP: Create Azure Logical datacenter.	Create Azure Logical datacenter for provisioning resources.	Orlando
CMP: Create "Contains: Contained by" relationship.	Create "Contains: Contained by" relationship between Resource Group and Azure datacenter.	Orlando
CMP: Create 'Hosted on' relationship.	Create hosted on relationship between logical datacenter and Cloud Service account.	Orlando
CMP: Create Cloud Account	Create cloud account by inserting the fields into the cloud account table.	Orlando
CMP: Cloud account and Logical datacenter association	Create an association between CMP cloud account and Logical datacenter.	Orlando
CMP: Create Catalog Item	Create Cloud Catalog Item with ARM Template.	Orlando
CMP: Create template version.	Create Cloud Template Version for Cloud Catalog Item.	Orlando

Cloud Provisioning and Governance: Azure test suite (continued)

Test	Description	Release version
CMP: ARM template body	Update Cloud Template Version with ARM template body.	Orlando
CMP: Open the 'ServiceNow Cloud Template Versions'	Open the 'ServiceNow Cloud Template Versions' GUI page.	Orlando
CMP: Activate Cloud Template Version	Activate Cloud Template Version to create resource block, blueprint, and catalog.	Orlando
CMP: Verify the Cloud Template Version.	Validate whether status is Success and state is active in the Cloud Template Version after Activation operation.	Orlando
CMP: Activate Cloud Catalog Item	Activate Cloud Catalog Item to display catalog order form to the cloud user.	Orlando
CMP: Add stack and resource group name in Catalog item	Set the default value for stack name and resource group in the catalog item.	Orlando
CMP: Verify whether stack is present	Validate whether stack is available with name "ATFARMStack".	Orlando
CMP: Open Cloud Catalog Item page	Open Cloud Catalog Item page in the Service Portal.	Orlando
CMP: Submit Catalog form	Submit the Catalog Form to start provisioning ARM stack.	Orlando
CMP: Verify whether stack status is active	Validate the status of a stack after provisioning the stack.	Orlando
CMP: Verify stack status after Day2 - Stop operation	Validate the status of the stack status changed from 'On' to 'Off' after the Stop operation.	Orlando
CMP: Verify stack status after Day2 - Start operation	Validate the status of the stack status changed from 'Off' to 'On' after the Start operation.	Orlando
CMP: Verify stack status after Day2 - Deprovision operation	Validate the status of the stack status changed from 'On' to 'Terminate' after the de-provisioning operation.	Orlando

Cloud Provisioning and Governance: AWS test suite

Test	Description	Release version
CMP: Add AWS Credentials	Add AWS credentials by inserting the fields into the credential table.	Orlando

Cloud Provisioning and Governance: AWS test suite (continued)

Test	Description	Release version
CMP: AWS Service Account	Add AWS service account by inserting the fields into the service account table.	Orlando
CMP: Update datacenter type	Update the cloud service account with the datacenter type.	Orlando
CMP: Create logical datacenter	Create Logical datacenter for provisioning the resources.	Orlando
CMP: Create hosted on relationship	Create hosted on relationship between logical datacenter and Cloud Service account.	Orlando
CMP: Create Cloud Account	Create cloud account by inserting the fields into the cloud account table.	Orlando
CMP: Cloud account and Logical datacenter association	Create an association between CMP cloud account and logical datacenter.	Orlando
CMP: Create Catalog Item	Create Cloud Catalog Item with the CFT Template.	Orlando
CMP: Create template version	Created Cloud Template Version for a Cloud Catalog Item.	Orlando
CMP: CFT template body	Update Cloud Template Version with the CFT template body.	Orlando
CMP: Open the ' ServiceNow Cloud Template Versions'	Open the ' ServiceNow Cloud Template Versions' GUI page.	Orlando
CMP: Activate Cloud Template Version	Activate Cloud Template Version to crate resource block, blueprint, and catalog.	Orlando
CMP: Verify the Cloud Template Version	Validate the Cloud Template Version whether status is Success and state is active after Activation operation.	Orlando
CMP: Activate Cloud Catalog Item	Activate Cloud Catalog Item to display the catalog order form to the cloud user.	Orlando
CMP: Add stack name in Catalog item	Set the default value for stack name in the catalog item.	Orlando
CMP: Verify whether stack is present	Validate whether stack is available with name "ATFCFTStack".	Orlando
CMP: Open Cloud Catalog Item page	Open the Cloud Catalog Item page in the Service Portal.	Orlando

Cloud Provisioning and Governance: AWS test suite (continued)

Test	Description	Release version
CMP: Submit Catalog form	Submit the Catalog Form to start provisioning CFT Stack.	Orlando
CMP: Verify whether stack status is active	Validate the status of the stack after provisioning the stack.	Orlando
CMP: Verify stack status after Day2 - Stop operation	Validate the status of the stack status changed from 'On' to 'Off' after the Stop operation.	Orlando
CMP: Verify stack status after Day2 - Start operation	Validate the status of the stack status changed from 'Off' to 'On' after the Start operation.	Orlando
CMP: Verify stack status after Day2 - De-provision operation	Validate the status of the stack status changed from 'On' to 'terminate' after the de-provisioning operation.	Orlando

To learn more about Cloud Provisioning and Governance, see Cloud Provisioning and Governance.

Coaching

Coaching quick start tests requires activation of the Coaching plugin (com.sn_coaching).

Coaching test suite

Test	Description	Release version
Coaching: Create an assessment manually when logged in as a coach.	As a coach, verify that you can create an assessment and assign it to trainees.	Orlando
Coaching: Add skills to an opportunity and verify those skills awarded to trainee.	Add skills to a coaching opportunity and verify that those skills have been awarded to trainees after they complete an assessment.	Orlando
Coaching: Complete an assessment as a virtual coach.	Verify that the virtual coach completes an assessment and provides feedback to the trainee when a virtual coach is attached to an opportunity.	Orlando
Coaching: Add skills to an assessment and verify those skills awarded to trainee.	Add skills to a coaching assessment and verify that those skills have been awarded to trainees after they complete an assessment.	Orlando
Coaching: Add skills to a recommendation and verify those skills awarded to trainee.	Add skills to a coaching recommendation and verify that those skills have been	Orlando

Coaching test suite (continued)

Test	Description	Release version
	awarded to trainees after they complete an assessment.	
Coaching: Move assessments from one state to another when logged in as a coach.	As a coach, verify that you can move an assessment from one state to another.	Orlando
Coaching: Attach a recommendation learning to an assessment.	Verify that a recommendation learning on an opportunity gets attached to an assessment when an assessment is generated.	Orlando
Coaching: When a coaching opportunity is inactive, assessments are not generated.	Verify that when a coaching opportunity is in inactive state, it does not generate assessments.	Orlando
Coaching: Submit Coaching survey as a Coach user.	Verify that coach can submit survey for a trainee.	Quebec
Coaching: Submit Coaching survey as a Traineer user.	Verify that trainee can submit survey for a coach.	Quebec

To learn more about coaching, see Coaching.

Communities

Communities quick start tests require activating the Customer Communities plugin (com.sn_customer_communities) and the Communities Demo Data plugin (com.sn_communities_demo).

Communities: Community test suite

Test	Description	Release version
Post question and validate feed	Verify that a question is posted in the community and validate whether it appears in the content feed.	Madrid
Post video	Verify that a video is posted in the community.	Madrid
Forum membership approval	Verify that a membership request to a forum is approved.	Madrid
Forum membership reject	Verify that a membership request to a forum is rejected.	Madrid
Approve a moderation task	Verify that a moderation task is approved.	Madrid
Reject a moderation task	Verify that a moderation task is rejected.	Madrid

Communities: Community test suite (continued)

Test	Description	Release version
Topic subscription and activity feed validation	Verify that a topic is subscribed to and the topic activities appear in the activity feed.	New York
Approve content approval workflow task on question	Verify the content approval workflow of a question.	New York
Question auto-subscription and activity feed validation	Verify that the author is automatically subscribed to the question and the question activities appear in the activity feed.	New York
Follow a user	Verify that a community user is able to follow another community user.	New York
Reject content approval workflow task on question	Verify the content approval workflow of a question when content is rejected.	New York
Gamification on video posting	Verify the gamification points gained when posting a video.	New York
Forum subscription and activity feed validation	Verify that a forum is subscribed to and the forum activities appear in the activity feed.	New York
Question subscription and activity feed validation	Verify that a question is subscribed to and the question activities appear in the activity feed.	New York
Post a video with 'Disable comment' option	Verify a video is posted with *`Disable comments`* option as selected.	Orlando
Pin a video as Featured	Verify that a video is marked as featured by community administrator in the forum and community home page.	Orlando

To learn more about Communities, see Communities.

Configuration Compliance

Configuration Compliance quick start tests require activating the Configuration Compliance application (sn_vulc) and loading the demo data.

Configuration Compliance quick start tests

Test	Description	Release version
Configuration Compliance Assignment Rules	Create an assignment rule.	Xanadu
Configuration Compliance - Reapply Group Rule	Reapply a test result group rule.	Xanadu
Configuration Compliance - Delete Group Rule	Delete a test result group rule.	Xanadu

To learn more about Configuration Compliance, see Configuration Compliance.

Configuration Management Database (CMDB)

Configuration Management Database (CMDB) quick start tests require activating the Configuration Management (CMDB) plugin (com.snc.cmdb) and the CMDB - ATF Tests plugin (com.snc.cmdb.atf).

CMDB BSM: Dependency Views test suite

Test suite to check functionality of Dependency Views APIs.

Test	Description	Release version
CMDB BSM: Dependency Views	Test functionality of Dependency Views APIs. These APIs retrieve Dependency Views map and associated map items such as context menu items, for a given CI sys_id and using itil user role.	New York

CMDB HEALTH: CMDB Health Dashboard test suite

Test suite to check whether CMDB CMDB Health Dashboard is functional at a basic level.

Test	Description	Release version
CMDB HEALTH: Health Job Status	Checks whether any CMDB Health dashboard jobs, which were started 30 or more days ago, are still in progress.	New York
CMDB HEALTH: CMDB Health Completeness/ Recommended	Checks whether the recommended metric (included in the CMDB Health completeness KPI) is fully functional. This test validates:	New York

CMDB HEALTH: CMDB Health Dashboard test suite

Test suite to check whether CMDB CMDB Health Dashboard is functional at a basic level.

(continued)

Test	Description	Release version
	• Creation of a health inclusion rule for the recommend metric. • Creation of a recommended field that satisfies the health inclusion rule. • Validate that the health inclusion rule is correctly applied to a test record with missing data in the recommended field.	

CMDB IRE: Identification Reconciliation Engine test suite

Test suite to check Identification and Reconciliation Engine (IRE) functionality.

Test	Description	Release version
CMDB IRE: IRE Validation	Validate CI identifiers and reconciliation definitions and check indexes for CI identifiers.	Madrid
CMDB IRE: Reconciliation Rule	Check operations on a reconciliation rule, in CI Class Manager, using itil and itil_admin roles. Operations include create, edit, and delete a reconciliation rule. Also, check for active and not active setting, and derived rules.	Paris
CMDB IRE: Identification Rule	Check operations on an identification rule, in CI Class Manager, using itil and itil_admin roles. Operations include create, edit, and delete an identification rule. Also, check for active and not active setting, and derived rules.	Paris

CMDB QB: Query Builder test suite

Test suite to verify CMDB Query Builder functions such as create query, read query, and execute query using two related user roles - cmdb_query_builder and cmdb_query_builder_read.

Test	Description	Release version
CMDB QB: Query Builder - cmdb_query_builder Role	Verify that cmdb_query_builder user role can save queries, and access and run all saved queries, in CMDB Query Builder.	New York
CMDB QB: Query Builder - cmdb_query_builder_read Role	Verify that cmdb_query_builder_read user role can access and run all saved queries, and cannot save any query, in CMDB Query Builder.	New York

CMDB REL: Relationship Editor and Formatter test suite

Test suite to verify functionality of Relationship Editor and Relationship Formatter.

Test	Description	Release version
CMDB REL EDITOR:Relationship Editor	Check addition of relations to a CI and deletion of relations from a CI using itil user role.	New York
CMDB REL FORMATTER:Relationship Formatter	Check accuracy of CI information, relationship types, relationships, associated records such as change tickets, and settings such as CMDB views (relationship filters), displayed for a specific CI in relationship formatter using itil user role.	New York

CMDB SDK: SDK REST API test suite

Test suite to verify functionality of CMDB SDK Rest APIs.

Test	Description	Release version
CMDB SDK: Query CMDB Metadata	Test querying CMDB metadata.	New York
CMDB SDK: Create a relation for a CI using REST APIs	Test creation of a relationship for a CI in the CMDB using the CMDB REST APIs.	New York

CMDB SDK: SDK REST API test suite

Test suite to verify functionality of CMDB SDK Rest APIs.

(continued)

Test	Description	Release version
CMDB SDK: Delete a relation for a CI using REST APIs	Test deletion of a relationship for a CI using CMDB REST APIs.	New York
CMDB SDK: Create a CI using REST API	Test creation of a CI using CMDB REST APIs.	New York
CMDB SDK: Query CMDB using REST APIs	Test querying the CMDB using CMDB REST APIs.	New York
CMDB SDK: Update a CI using REST APIs	Test updating of a CI using CMDB REST APIs.	New York
CMDB SDK: Query for a CI using REST APIs	Test querying a CI using CMDB REST APIs.	New York

To learn more about Configuration Management Database, see Configuration Management Database.

Continual Improvement Management

Continual Improvement Management quick start tests require activating the Continual Improvement Management Automated Tests plugin (com.sn_cim.atf)

Continual Improvement Management tests

Test	Description	Release version
CIM: Inbound integration from Assessments	Create an improvement initiative from an assessment.	Paris
CIM: Inbound integration from Incident Management	Create an improvement initiative from an incident.	Paris
CIM: Inbound integration from Problem Management	Create an improvement initiative from a problem.	Paris
CIM: Inbound integration from Survey Management	Create an improvement initiative from a survey.	Paris

To learn more about Continual Improvement Management, see Continual Improvement Management.

Customer Service Management

All Customer Service Management quick start tests require activating the Customer Service Management Demo Data plugin (com.snc.customerservice.demo). Some quick start tests also require activating the following plugins:

- Business Location (com.snc.business_location)
- Case Playbook for Complaints (sn_complaint)
- Case Playbook for Onboarding (sn_onboarding)

- Consumer Service Portal (com.glide.service-portal.consumer-portal)
- CSM Contributor User (com.snc_csm_contributor_user)
- CSM Extension for Proxy Contacts (com.snc.csm_proxy_contacts)
- Customer Service Household (com.snc.household)
- Customer Service Management for Orders (com.snc.csm.order)
- Customer Service Portal (com.glide.service-portal.customer-portal)
- Customer Service with Request Management (com.sn_cs_sm_request)
- Customer Service with Service Management (com.sn_cs_sm)
- Guided Decisions Experience (com.snc.guided_decisions_playbook_experience)
- Guided Decision - Next Best Action (com.snc.next_best_action)
- Major Issue Management (com.sn_majorissue_mgt)
- Proactive Customer Service Operations with Event Management (com.snc.proactive_cs_itom)
- Skill Determination (com.snc.skill_determination)
- Walk-Up for CSM (com.snc.walkup_for_csm)

CSM: Case Management test suite

Test	Description	Release version
CSM: Create Product Case	Create a case for a product.	Madrid
CSM: Assign Case to an Agent	Create a case and assign it to a customer service agent.	Madrid
CSM: Assign Asset on Case	Assign an asset to a case.	Madrid
CSM: Assign Entitlement	Assign an entitlement to a case.	Madrid
CSM: Escalate an Account	Escalate an account.	Madrid
CSM: Escalate a Case	Escalate a case.	Madrid
CSM: Create Special Handling Notes	Create special handling notes for a case.	Madrid
CSM: Close a Case	Close a case.	Madrid
CSM: Time Recording	Record the time worked on a case.	Madrid
CSM: Create CHG from Case	Create a change record from a case.	Madrid
CSM: Create Incident from Case	Create an incident record from a case.	Madrid
CSM: Create Order Case	Create a case for an order. **Note:** Requires Customer Service Management for Orders.	Madrid

CSM: Case Management test suite (continued)

Test	Description	Release version
CSM - Create Order Case as Customer from CSM Portal	Create an order case as a customer from the Customer Service Portal. **Note:** Requires the Customer Service Portal. Also requires that the test be run as admin.	Madrid
CSM: Create Problem from Case	Create a problem record from a case.	Madrid
CSM - Create Proactive Case by NOC Operator	Verify whether a proactive case is created. **Note:** Requires Proactive Customer Service Operations with Event Management.	New York
CSM - Employee creating case OBO customer	As an employee with the proxy contact role (sn_customerservice_proxy_contact), create a case from the self-service portal on behalf of a customer. **Note:** Requires the CSM Extension for Proxy Contacts.	New York
CSM - Lookup Type Skill Determination Rule Test	Create a lookup type skill determination rule. **Note:** Requires Skill Determination.	New York
CSM - Manager creating request OBO customer from CSM portal	As a user with the case manager role, create a case on behalf of a customer from the Customer Service Portal. **Note:** Requires Customer Service Request Integration.	New York

CSM: Case Management test suite (continued)

Test	Description	Release version
CSM - Simple Type Skill Determination Rule Test	Create a simple type skill determination rule. **Note:** Requires Skill Determination.	New York
CSM: Register New Case Type	Register a case type and verify the record is created.	Orlando
CSM - Project Manager create Project for an Account	Project Manager creates project for an account. **Note:** Requires Customer Project Management.	Orlando
CSM - Project Manager identify Project Contact	Project manager identifies customers to a project. **Note:** Requires Customer Project Management.	Orlando
CSM - Project Manager create project task and assign to a customer	Project manager creates project task and assigns to a customer. **Note:** Requires Customer Project Management.	Orlando
CSM - Create Case from Project	Create a case from a project.	Orlando
CSM - Create Case from Project Task	Create a case from a project task.	Orlando
CSM: Create Task from Case	Create a case task from a case.	Orlando
CSM: Agent Creating Request for Customer	As a customer service agent, create a request for a customer. **Note:** Requires Customer Service Request Integration.	Orlando
CSM - Agent Create Cases from a Project	As a customer service agent, create a case from a project.	Orlando

CSM: Case Management test suite (continued)

Test	Description	Release version
	Note: Requires Customer Project Management.	
CSM - Agent Create Cases from a Project task	As a customer service agent, create a case from a project task. **Note:** Requires Customer Project Management.	Orlando
CSM - Agent Create Change Requests for a project	As a customer service agent, create a change request for a project.	Orlando
CSM: Create Major Case and its Child Cases	Create a major case and the associated child cases for the customer accounts in the recipient list. **Note:** Requires Major Issue Management.	Orlando
CSM: Advanced Type Skill Determination Rule Test	Create an advanced type skill determination rule. **Note:** Requires Skill Determination.	Orlando
CSM - Service Contracts covered under Sold Product	Create sold products and service contracts and associate service contracts to a sold product. Verify the association between the active contracts and the sold product. **Note:** Requires Customer Service Install Base Management.	Orlando
AWA - Create New Service Channel	Create a new service channel in the Advanced Work Assignment application.	Orlando

CSM: Case Management test suite (continued)

Test	Description	Release version
	Note: Requires Advanced Work Assignment for CSM.	
CSM-ITOM - Create Child Cases for Proactive Major Case	Create a child case for a proactive major case using recipient list. **Note:** Requires Proactive Customer Service Operations with Event Management.	Orlando
CSM - Create Outage from Case	Validate if newly created outage is linked to a case.	Paris
CSM: Create Sold Product on Household	Create a sold product on a household and its member. **Note:** Requires Customer Service Install Base Management and Customer Service Household with **Load demo data** enabled.	Quebec
CSM: View Health Status of Install Base Item from Account and Case pages	Validates the functionality of the **Refresh Install Base Health** button on the Account and Case record pages.	Quebec
CSM: Assign Case Task to Case Task Agent	The customer service agent creates and assigns a case task to a case task agent.	Rome
CSM: Case Task Agent views assigned Case Task	Verifies that the case task agent can view an assigned case task.	Rome
CSM: Case Task Agent Completes Assigned Task	The case task agent completes an assigned case task.	Rome
CSM: Report a knowledge gap from a case in Agent Workspace	Verify that a knowledge gap related to a case was reported in Agent Workspace	Rome
CSM-Create a case using 'Create a case (POST)' API	Create a case using 'Create a case (POST)' API.	Rome

CSM: Case Management test suite (continued)

Test	Description	Release version
	Note: Requires a user authentication record assigned to the **Basic authentication** field in step 1 of the ATF.	
CSM-Query a case using 'Query a case (GET)' API	Query a case using 'Query a case (GET)' API. **Note:** Requires a user authentication record assigned to the **Basic authentication** field in step 1 of the ATF.	Rome
CSM: OCS Manager creating New OSP	Verify that a new Outsourced Service Provider (OSP) is created by the OCS internal manager. **Note:** Requires Outsourced Customer Service.	Rome
CSM: Case Creation by OCS Agent	Verify that an Outsourced Customer Service agent is able to create a case. **Note:** Requires Outsourced Customer Service.	Rome
CSM: Manage Consumer Profile Locations	Create and manage consumer locations, and map them to consumer profiles.	Washington DC
CSM: Address sharing across Accounts	Implement reusable addresses for accounts to support complex customer operations and business models.	Washington DC

CSM Configurable Workspace test suite

Test	Description	Release version
CSM Workspace: Case creation Workflow	Tests the creation of a case record by a customer service	Xanadu

CSM Configurable Workspace test suite (continued)

Test	Description	Release version
	manager in CSM/FSM Configurable Workspace.	
CSM Workspace: Case follow up workflow	Tests the requesting of information from a customer by a customer service manager in CSM/FSM Configurable Workspace.	Xanadu
CSM Workspace: Report knowledge gap workflow	Tests the creation of a knowledge feedback task for a case record by a customer service manager in CSM/FSM Configurable Workspace.	Xanadu
CSM Workspace: Case resolution workflow	Tests the resolution of a case record by a customer service manager in CSM/FSM Configurable Workspace.	Xanadu
CSM Workspace: Create Incident from Case	Tests the creation of an incident record from a case record by a customer service agent in CSM/FSM Configurable Workspace.	Xanadu
CSM Workspace: Create Problem from Case	Tests the creation of a problem record from a case record by a customer service agent in CSM/FSM Configurable Workspace.	Xanadu
CSM Workspace: Create Normal Change from Case	Tests the creation of a change record from a case record by a customer service agent in CSM/FSM Configurable Workspace.	Xanadu

CSM: Case Types - Complaint

Test	Description	Release version
Create a Complaint Case	Tests that a user can create a case that is of type complaint in the sn_complaint_case table.	Paris
Escalate a Complaint Case	Tests that a user can escalate a case that is of type complaint in the sn_complaint_case table.	Paris

CSM: Case Types - Onboarding

Test	Description	Release version
Create an Onboarding Case	Tests that a user can create a case that is of type onboarding in the sn_onboarding_case table.	Paris
Escalate an Onboarding Case	Tests that a user can escalate a case that is of type onboarding in the sn_onboarding_case table.	Paris

CSM: Operations Dashboard test suite

Test	Description	Release version
awa_admin_operations_dashboard	Verify whether user with role awa_admin is able to view the Advanced Work Assignment menu under Operations Dashboard and unassigned interactions and unassigned task work items modules.	Orlando
awa_manager_operations_dashboard	Verify whether user with role awa_manager is able to view the Advanced Work Assignment menu under Operations Dashboard and unassigned interactions and unassigned task work items modules.	Orlando

CSM Portal test suite

Test	Description	Release version
CSM - Create Product Case as Customer from CSM Portal	Create a product case as a customer from the Customer Service Portal. **Note:** Requires Customer Service Portal. Also requires that the test be run as admin.	Madrid
CSM - Create Product Case as Partner from CSM Portal	Create a product case as a partner from the Customer Service Portal.	Madrid

CSM Portal test suite (continued)

Test	Description	Release version
	Note: Requires Customer Service Portal. Also requires that the test be run as admin.	
CSM - Search on Homepage	Search for information from the Customer Service Portal. The search includes cases, Knowledge articles, and Community threads.	New York
CSM - Update Support Profile	Update a contact's profile from the Customer Service Portal.	New York
CSM - Provide requested info on case	From the Customer Service Portal, the contact can provide information for a case that was requested by the agent.	New York
CSM - Accept Proposed Solution On Case	Accept a proposed solution for a case from the Customer Service Portal.	New York
CSM - Provide Feedback on Survey	Provide feedback on a survey after a case is closed from the Customer Service Portal.	New York
CSM - View All Desktop Notifications	View all Customer Service Management specific desktop notifications.	Orlando
CSM - View Publications on CSM Portal	View publications on the Customer Service Management portal.	Orlando
CSM - Create Contact on CSM Portal	Create contacts on the Customer Service Management portal.	Orlando
CSM: Validate Outage widgets in CSM Portal	Validates various types of outages and the corresponding widgets shown on the Customer Service Portal home page and the Install Base page.	Quebec
CSM - Create case from Portal as Consumer Contributor	Creates a customer case from the portal by a user with the consumer contributor user role (sn_customerservice.consumer_contributor).	Rome

CSM Portal test suite (continued)

Test	Description	Release version
	Note: Requires Business Location, Customer Service Household, and CSM Contributor User.	
CSM - Create case from Portal as Account Contributor	Creates a customer case from the portal by a user with the account contributor user role (sn_customerservice.account_contributor). **Note:** Requires Business Location, Customer Service Household, and CSM Contributor User.	Rome
CSM - Add Related parties on Case and perform action on case as Related party	Verify the ability to add related parties to the case and perform actions on the case as related parties through the CSM Portal.	Rome

CSP Portal test suite

Test	Description	Release version
CSP - Create Product Case as Consumer from CSP Portal	Create a product case as a consumer from the Consumer Service Portal. **Note:** Requires the Consumer Service Portal. Also requires that the test be run as admin.	Madrid
CSP - Search on Homepage	Search for information from the Consumer Service Portal. The search includes cases, knowledge articles, and Community threads.	New York
CSP - Update Support Profile	Update a consumer's profile from the Consumer Service Portal.	New York
CSP - Register Your Product	Register a product from the Consumer Service Portal.	New York
CSP - Provide requested info on case	From the Consumer Service Portal, the consumer can	New York

CSP Portal test suite (continued)

Test	Description	Release version
	provide information for a case that was requested by the agent.	
CSP - Accept Proposed Solution On Case	Accept a proposed solution for a case from the Consumer Service Portal.	New York
CSP - Provide Feedback on Survey	Provide feedback on a survey after a case is closed from the Consumer Service Portal.	New York
CSP - View Publications on CSP Portal	View publications on Customer Service portal.	Orlando

Guided Decision - Next Best Action ATF test suite

Test	Description	Release version
Validate Next Best Action List- Guided Decisions	Validates the list of next best actions recommended based on the configured rules.	San Diego
Validate Next Best Action Ranking- Guided Decisions	Validates the list of next best actions recommended based on the configured rules.	San Diego

TC: Targeted Communications test suite

Test	Description	Release version
TC - Create Recipient List	Create a recipient list with the required parameters. Verify the new recipient list in the related list "Recipients".	Orlando
TC - Create Publication	Create a publication. The publication is published based on the publication date and verify if an user in the recipient list gets the publication.	Orlando

CSM Walkup Experience tests

Test	Description	Release version
CSM Walkup: Check-in as a walkup user, look at the queue, and submit a survey	As a walk-up user, perform check-in, look at the queue, and submit feedback through a survey.	Rome

CSM Walkup Experience tests (continued)

Test	Description	Release version
	Note: This test works when the Seattle location, available with demo data, is available.	

To learn more about Customer Service Management, see Customer Service Management.

Dashboards

Dashboards quick start tests require activating the Automated Test Framework - Responsive Dashboards plugin (com.glide.automated_testing_impl.dashboards). This plugin is active on zBoot of the instance.

Dashboards quick start tests

Test	Description	Release version
Responsive Dashboard Sharing	Confirm dashboard sharing by impersonating users.	Madrid
Responsive Dashboard Visibility	Confirm dashboard visibility by impersonating users.	Madrid

To learn more about Dashboards, see Dashboards.

DevOps

DevOps quick start tests are available when you install the DevOps application from ServiceNow Store.

DevOps test suite

Test	Description	Release version
DevOps Code Tool Flow	Verify the Workflow Studio flow for a DevOps coding tool.	Madrid
DevOps Orchestration Flow with CR	Verify the Workflow Studio flow for a DevOps orchestration tool that includes a change request.	Madrid
DevOps Orchestration Tool Flow	Verify the Workflow Studio flow for a DevOps orchestration tool.	Madrid
DevOps Plan Tool Flow	Verify the Workflow Studio flow for a DevOps planning tool.	Madrid

To learn more about DevOps, see DevOps Change Velocity.

Employee Center and Employee Center Pro

Employee Center quick start tests

Test	Description	Release version
Create Adhoc Delegation for Approval Task	Ensures an adhoc delegation for an approval task can be created from the My Tasks form in the Employee Center.	San Diego
Create Adhoc Delegation for HR Task	Ensures an adhoc delegation for an HR task can be created from the My Tasks form in the Employee Center.	San Diego
Employee Center - My Favorites	Verifies the favorite functionality in the Employee Center. Also verifies when a KB article or catalog item is added to the Favorite widget.	San Diego
Employee Center - Approval Hub Approve Request	Verifies the approval functionality in the Employee Center. Also verifies that when a request is approved, it appears in the user's **Completed** column.	San Diego
Employee Center - Approval Hub Reject Request	Verifies the rejection functionality. Also verifies that when a request is rejected, it appears in the user's **Completed** column.	San Diego
Employee center - Employee Profile Generation	Verifies that employee profiles are generated as per the employee definition.	Xanadu
Employee Center - Topic Page	Associates a KB article or catalog item to a topic and verifies they display on the topic page.	San Diego
Employee Center - Validate home page widgets	Validates the following widgets are present on the Employee Center home page: • Relevant for you • Popular topics • My Active Items • Quick tasks • Homepage Search	San Diego
Employee Center Pro - App launcher	Verifies the following on the My Application page:	San Diego

Employee Center quick start tests (continued)

Test	Description	Release version
	• Creates an application and verifies it is visible on the application page. • Creates a topic/category and associates the application and verifies it is visible under the topic and category.	
Employee Center Pro - Validate home page widgets	Validates the following widgets are present on the Employee Center Pro home page: • Content Experience • Recommended for you • Popular topics • Upcoming Events • My Active Items • Videos • App launcher • Homepage Search	San Diego
ESC: Employee can see ticket updates	Employees can view updates to their tickets.	Quebec
ESC: Post General HR Inquiry questions on ESC portal	Verifies an HR employee can create a General Inquiry case and post general HR inquiry questions on the ESC.	Quebec
ESC: Submit a Record Producer which crates Universal Request and HR Case	Verifies a user can submit an HR catalog item that creates a Universal Request. Also verifies the Universal Request and HR case are created and linked.	Quebec
ESC: Verify Standard Ticket page on ESC for HR Case	Creates a general inquiry case from the service portal and verifies it appears on the standard Ticket page.	Quebec
ESC: Verify widget contents in Catalog items	Verifies the widget content in a Catalog page.	Orlando
ESC: Verify widget contents in knowledge pages	Verifies the widget content in a Knowledge page.	Orlando

Employee Center quick start tests (continued)

Test	Description	Release version
HR: Search catalog items & KBs in ESC	Verifies the search functionality in the ESC.	Quebec
Taxonomy and Topic Creation	Creates taxonomy and topics and then adds content to the topic.	San Diego

Note:
Requires plugin activation of:

- Human Resources Scoped App: Core plugin (com.sn_hr_core)
- Employee Center Core [app-ex-employee-center-core]
- Employee Center [app-ex-employee-center]
- Employee Center Pro [app-ex-employee-center-pro-content]
- Content Publishing [app-content-publishing]
- Content Experiences [app-content-experiences]

Content Experiences

Test	Description	Release version
Content Experiences: Create and Publish a Campaign	Creates and publishes a campaign with portal content and verifies against the Content Experiences portal.	Orlando
Content Experiences: Portal preview	Verifies that a campaign manager is able to preview portal content using the portal preview.	Rome
Content Experiences: Preview individual content before Publishing a Campaign	Verifies a campaign manager can preview any individual content for a campaign.	San Diego

Note:
Requires plugin activation of:

- Human Resources Scoped App: Core plugin (com.sn_hr_core)
- Employee Center Core [app-ex-employee-center-core]
- Employee Center [app-ex-employee-center]
- Employee Center Pro [app-ex-employee-center-pro-content]
- Content Experiences [app-content-experiences]

Content Governance

Test	Description	Release version
Content Governance - New Content Request E2E Workflow	Provides end-to-end testing from creating a content request through publishing the content.	San Diego

Note:
Requires plugin activation of:

- Human Resources Scoped App: Core plugin (com.sn_hr_core)
- Employee Center Core [app-ex-employee-center-core]
- Employee Center [app-ex-employee-center]
- Employee Center Pro [app-ex-employee-center-pro-content]
- Content Publishing [app-content-publishing]
- Content Experiences [app-content-experiences]
- Content Governance [app-content-governance]

Content Publishing

Test	Description	Release version
Content Publishing: Audience	Tests the audience configurations for Content Publishing.	Orlando
Content Publishing: Create Banner type content in SCA	Ensures the Create Banner type can be created when using Streamlined Content Authoring (SCA) under the Content Publishing module.	Rome
Content Publishing: Create Styled content (Video) type content in SCA	Verifies an admin can create styled content that is a video type using Content Publishing.	Rome
Content Publishing: Schedule Portal Content	Creates and schedules test content for the Content Publishing demo portal.	Orlando
Create news content via Content Library	Creates and publishes a news article to the portal.	Xanadu

Note:
Requires plugin activation of:

- Human Resources Scoped App: Core plugin (com.sn_hr_core)
- Employee Center Core [app-ex-employee-center-core]
- Employee Center [app-ex-employee-center]
- Employee Center Pro [app-ex-employee-center-pro-content]
- Content Publishing [app-content-publishing]
- Content Experiences [app-content-experiences]

Essential SAFe

Essential SAFe quick start tests require activating the Agile - Scaled Agile Framework - Essential SAFe plugin (com.snc.sdlc.safe) and the Agile - Scaled Agile Framework - Essential SAFe - ATF Tests plugin (com.snc.sdlc.safe.atf).

Essential SAFe test suites

Test	Description	Release version
Essential SAFe: Feature tests	Verify feature global rank updates.	Madrid
Essential SAFe: Feature tests	For a SAFe feature, verify that: • Actual start date is populated after the state is changed to Implementation, Validation on Staging, or Deployment. • Actual end date is populated after the state is changed to **Released** or **Cancelled**. • Active flag is set to the appropriate value: ○ **False**, if the state is changed to **Released** or **Cancelled**. ○ **True**, for all other states.	Orlando
Essential SAFe: Feature tests	If the PPM Standard plugin (com.snc.financial_planning_pmo) is active, verify that a Demand can be converted to a SAFe feature using the **Create SAFe Feature** related link on the Demand form. For more information on how to create an SAFe feature from a demand, see Create an artifact from a demand.	Quebec
Essential SAFe: Program increment tests	Verify program increment date overlapping.	Madrid
Essential SAFe: Sprint tests	Verify the generation of ART sprints and team sprints as well as updates to sprint points and dates.	Madrid

Essential SAFe test suites (continued)

Test	Description	Release version
Essential SAFe: Sprint tests	Verify that any update to the **Group capacity** field of the assignment group results in the following changes to the **Group capacity** field of the various sprints associated with this assignment group: • For the sprints that are in the **Draft** or **Planning** states: ○ The group capacity is updated to the new value. ○ The **Group capacity** field is editable. • For the sprints in the **Current**, **Complete**, or **Cancelled** states: ○ The group capacity remains the old value. ○ The **Group capacity** field is read-only. For the sprints in the **Draft** or **Planning** state, you can individually edit the group capacity of the sprint anytime later. This would not change the group capacity of the assignment group associated with this sprint.	Paris
Essential SAFe: Story tests	Verify story global rank updates.	Madrid
Essential SAFe: Story tests	Verify that active flag of the SAFe story is set to the appropriate value: • **False**, if the state of the state is changed to Completed or Cancelled. • True, for all other states.	Orlando
Essential SAFe: Story tests	Verify that adding, estimating, removing, deleting, updating, or cancelling a SAFe story updates the SAFe feature-	Orlando

Essential SAFe test suites (continued)

Test	Description	Release version
	level and then the epic-level roll-ups correctly.	
Essential SAFe: Story tests	Verify that adding, updating, or deleting the feature on a SAFe story updates the Epic field on the SAFe story form.	Orlando
Essential SAFe: Story tests	If the PPM Standard plugin (com.snc.financial_planning_pmo) is active, verify that a Demand can be converted to a SAFe story using the **Create SAFe Story** related link on the Demand form. For more information on how to create a SAFe story from a demand, see Create an artifact from a demand.	Quebec
Essential SAFe: Team tests	Verify team association with an ART.	Madrid
Essential SAFe: Epic tests	For a SAFe epic, verify that: • Actual start date is populated after the state is changed to **Implementation**. • Actual end date is populated after the state is changed to **Complete**. • Active flag is set to the appropriate value: ○ **False**, if the state is changed to **Released** or **Cancelled** states. ○ **True**, for all other states.	Orlando
Essential SAFe: Epic tests	If the PPM Standard plugin (com.snc.financial_planning_pmo) is active, verify that a Demand can be converted to a SAFe epic using the **Create SAFe Epic** related link on the Demand form. For more information on how to create a SAFe epic agile	Quebec

Essential SAFe test suites (continued)

Test	Description	Release version
	from a demand, see Create an artifact from a demand.	
Essential SAFe: Program PI Objective tests	Verify the functionality of creating and updating PI objectives • Verify that you can create a program increment-level objective • Verify that you can update the created PI objective with the planned business value (PBV) and actual business value (ABV) • Verify that the percentage of business value achieved is computed as the percentage of ABV complete, for only the committed PI objectives, as compared to the PBV of the program in the PI	Rome
Essential SAFe: Team PI Objective tests	Verify the functionality of creating and updating team PI Objectives • Verify that you can create a team-level PI objective • Verify that you can update the created PI objective with the planned business value (PBV) and actual business value (ABV) • Verify that the percentage of business value achieved is computed as the percentage of ABV complete, for only the committed PI objectives, as compared to the PBV of the team in the PI	Rome

To learn more about Essential SAFe, see Essential SAFe.

Event Management

Event Management quick start tests require activating the Event Management plugin (com.glideapp.itom.snac).

Event Management quick start tests

Test	Description	Release version
EMSelfMonitoring	Track problems with the Event Management plugin after upgrade. To understand the details of issues found, open the All Alerts list and search for alerts that failed, whose source is `EMSelfMonitoring`, and that were opened or reopened since the upgrade. The exact time for filtering can be found in the error message.	Madrid

To learn more about Event Management, see Event Management.

Field Service Management

Field Service Management quick start tests are available when you enable the Field Service Management plugin (com.snc.work_management). Enable the demo data plugin (com.snc.work_management.demo) in a non-production instance to start using the quick start tests available with your application. Activate Field Service Contractor Management plugin (com.snc.fsm_contractor_management) to execute the External contractor related tests. You can also modify existing data and customize it to run the quick tests.

FSM: Field Service Management test suite

Test	Description	Release version
FSM: Create Initiate Qualify Dispatch and assign Work Order Task	• Create a work order. • Initiate and qualify a work order. • Dispatch a work order. • Assign a work order task to an agent.	Madrid
FSM: Part Sourcing	• Source any part to an agent's stock room from the work order task. • Create a transfer order line for the part sourcing. • Use this part for any work order to consume it using the **Part Usage** action.	Madrid

FSM: Field Service Management test suite (continued)

Test	Description	Release version
FSM: Part Usage	• Source any part to an agent's stock room from the work order task. • Use this part for any work order to consume it using the Part Usage action.	Madrid
FSM: Questionnaire	Create a questionnaire and associate it with a work order task.	Madrid
FSM: Field Service Configuration	Verify that the default configuration such as task assignment method, qualification requirement, PDF summary and agent's ability to accept or reject tasks is preserved.	New York
FSM: Planned Maintenance	Create a planned maintenance work order with weekly interval time for printer maintenance.	New York
FSM: Appointment Booking Configuration	Verify that the default configuration for appointment booking is preserved and that the point of sale service and catalog item exists in the system.	New York
FSM: Create Work or Personal Schedules	Create personal or work schedule for agents.	Orlando
FSM: Field Service Property Settings	Verify that the field service system properties preserve expected values.	Orlando
FSM: Work Groups	Verify that field service agents can be added to work groups.	Orlando
FSM: Dynamic Scheduling - Preferred Technician assignment with mandatory parts reservation	With Dynamic scheduling: • Preferred technician should be picked for task assignment. • Mandatory parts should be reserved in the agent stock room.	Paris

FSM: Field Service Management test suite (continued)

Test	Description	Release version
	Note: Activate Customer service management demo data plugin.	
FSM: Create Time Card	Verify that a time card is created for an agent in the work order task.	Quebec
FSM: Create incidental	Verify that an incidental is created for an agent in the work order task.	Quebec
FSM: Onboarding contractor company	Verify that a contractor company is onboarded with assignment group, external manager, and external agent. **Note:** Activate Field Service Contractor Management plugin.	Quebec
FSM: Onboarding contractor agent by external manager	Verify that an external contractor manager can onboard external agents from the Field Service Contractor Management portal.	Quebec
FSM: External contractor manager fulfil the work order task	Verify that a contractor manager or an agent is able to fulfil the assigned work order task.	Quebec
FSM: Assign work order task to Vendor group	Verify that a work order task is assigned to the external assignment group based on the defined criteria, such as task location and configuration parameters.	Quebec
FSM: Pause and Resume work order task	Verify that an agent can pause and resume work for a work order tasks.	Rome
FSM: Dynamic Scheduling - Assign technician with matching skills	With Dynamic Scheduling: Verify that a work order task is assigned to a technician who possesses all mandatory skills mentioned in the task.	Rome
FSM: Off boarding contractor agent by external manager	Verify that an external contractor manager can off board external agents from	Rome

FSM: Field Service Management test suite (continued)

Test	Description	Release version
	the Field Service Contractor Management portal.	
FSM: External contractor manager asset sourcing and usage	Verify that an external contractor manager is able to perform the following actions: • Request parts from stockroom. • Pick the part. • Close the transfer order. • Perform work using the part.	San Diego
FSM: External contractor agent fulfil the work order task	Verify that a contractor agent is able to fulfil the work order task that is assigned by the external contractor manager.	San Diego
FSM: Capacity Scheduling - Assign work to field service agents based on capacity	Verify that the rules and values mentioned in the capacity definition and capacity assignment module are evaluated to assign work to the agent based on capacity.	Tokyo
FSM: Onboarding external contractor agent as additional manager for the external assignment group	Verify that when onboarding an external agent for the contractor company, you can assign the additional manager role to the agent.	Tokyo
FSM: Dispatcher Workspace - Assign crews to work order tasks that require a group of agents to work on them	• Verify that a dispatcher with the crew moderator role can create a crew in the Dispatcher Workspace. • Verify that a dispatcher with the crew moderator role can assign a work order task to the planned crew in the Dispatcher Workspace	Tokyo
FSM: Work order task supports Multi-day task scheduling	Verify the Multiday task scheduling functionality: • Create Work order, work order task, and agent schedule records. • Select the Assign across schedule entries option in the work order task.	Tokyo

FSM: Field Service Management test suite (continued)

Test	Description	Release version
	• Assign an agent to the work order task for which the task duration is more than a day. • Verify that work order task is assigned successfully to the agent and the estimated end time is populated correctly.	
Verify creating a work order from a case and assigning a work order task to an agent from CSM/ FSM Configurable Workspace	• Verify you can create a work order from a case. • Qualify the work order and verify a work order task is created. • Assign the work order task to an agent and verify the agent it is assigned to is updated correctly.	Xanadu

Finance Close Automation

Finance Close Automation quick start tests are available when you install the Finance Close Automation application from ServiceNow Store.

FCA test suite

Test	Description	Release version
FCA: Applicable Months sub-set tests	Verify that the Applicable months can only be sub-set of functional workbook.	New York
FCA: dates validations	Verify that the date related calculations are correct.	New York
FCA: task relationships	Verify that dates and states are updated correctly for the tasks having relationships.	New York
FCA: relationship with parent tasks	Verify that dates and states are updated correctly for tasks that have relationship with parent tasks.	New York
FCA: Reset Task	Verify the FCA workflow when the task is reset.	New York
FCA: Reject Task	Verify the FCA workflow when the task is rejected.	New York

FCA test suite (continued)

Test	Description	Release version
FCA: Create JE Task	Verify that ERP source is available for a Journal Entry task type.	New York
FCA: Negative values as close day	Verify that both the positive and negative values are acceptable for close day and the planned dates are calculated correctly.	New York
FCA: Close day checks for Daily close Checklist	Close day verifications in daily close workbook.	New York
FCA: Daily close planned end date checks	Verify planned end dates in a daily close workbook.	New York
FCA: Kickstart errors	Verify that all the kickstart validations run.	New York
FCA: One final task verification	Verify that there is only one final task in each functional workbook.	New York
FCA: Confidential task	Verify that only the owner, reviewer, and approver of a confidential task can view the task.	New York
FCA: Unique checklist Verification	Verify that only one workbook is created for a specific day or period.	New York
FCA: Milestone and JE Validations	Verify that milestone and journal entry tasks don't have any child tasks.	New York

Financial Management

Financial Management quick start tests require activating the Financial Management Core - ATF Tests plugin (com.snc.financial_management.atf)

ITFM: Financial Modeling flow test suite

Test	Description	Release version
Verify FM Cost Allocation Flow	Verify the cost allocation flow in financial modeling.	Madrid

To learn more about Financial Management, see Financial Management.

Granular Delegation

Granular Delegation quick start tests require activating the Granular Delegation (com.glide.granular_service_delegation) plugin.

Test	Description	Release version
[Delegation] Admin - Adding a delegation rule with delegator and delegate user criteria	Ensures a delegate or delegator honors the user criteria assigned to an HR task.	Xanadu

GRC Audit Management

GRC: Audit Management quick start tests require activating the GRC: Audit Management plugin (com.sn_audit) and loading the demo data.

GRC: Audit Management test suite

Test	Description	Release version
GRC: Create Audit Engagement and Generate Audit Task	Validates audit engagement creation and associates entities to generate controls and test plans. Generates audit task which is associated to a test plan.	Paris
GRC: Create and process a milestone	Create a milestone in an engagement, notice that the due date cannot be in past and the completion date cannot be in future and for a milestone in open state the percent complete is 0 which changes in accordance with the milestone state change.	Paris
GRC: Cost and Resource plan rollup	Create an audit plan and associate an engagement to it, on adding cost plan and resource plan to this engagement, notice that these costs are rolled up to the plan. Any edits to these costs in engagement reflects in the plan.	Paris
GRC: Create Engagement Project Manual and automatic	On an engagement in the validate state, once the Enable advanced planning ui-action is performed notice an engagement project gets created and when the state of an engagement associated to an audit plan having "Advanced planning capabilities" is changed to validate notice that an	Paris

GRC: Audit Management test suite (continued)

Test	Description	Release version
	engagement project gets created automatically.	
GRC: Auditable Unit with Detailed Risk Assessment	Create an Auditable unit with method as "Detailed Risk Assessment" and request for Assessing the Risk Assessment by adding the Assessor once the assessor responds and Marks Assessment as Complete after performing the control assessment and residual assessment, the risk assessment fields should be auto updated.	Paris

To learn more about Audit Management, see Audit Management.

GRC Continuous Authorization and Monitoring

GRC: Continuous Authorization and Monitoring quick start tests require activating the Continuous Authorization and Monitoring plugin (com.sn_compliance) and loading the demo data.

GRC: Continuous Authorization and Monitoring Quick Start Tests test suite

Test	Description	Release version
GRC: System Owner create and validate responsibilities and roles for an AB and AP	System Owner creates and validates responsibilities and roles for an Authorization Boundary and Authorization Package. Information Owners and System User are pre-populated when selecting the Authorization Boundary.	Quebec (compatible with Paris and Orlando)
GRC: System Owner validate App Modules visibility	Verifies that the system owner persona is able to view the Continuous Authorization & Monitoring application menu and the following modules under that menu: • All Authorization Boundaries • All Authorization Packages • Information Type Library	Quebec (compatible with Paris and Orlando)

GRC: Continuous Authorization and Monitoring Quick Start Tests test suite (continued)

Test	Description	Release version
	• Control Overlays • Control Objectives • Controls • All Engagements	
GRC: System Owner Request First approval & My approvals module	System Owner requests an approval.	Quebec (compatible with Paris and Orlando)
SO: Create and validate responsibilities and roles for an AB and AP	Verifies if a system owner can create an Authorization Boundary by completing the fields on the Authorization Boundary form. Also verify if the same SO can create an Authorization Package from the form view.	Quebec (compatible with Paris and Orlando)

To learn more about Continuous Authorization and Monitoring, see Continuous Authorization and Monitoring.

GRC Policy and Compliance Management

GRC: Policy and Compliance Management quick start tests require activating the Policy and Compliance Management plugin (com.sn_compliance) and loading the demo data.

GRC: Policy and Compliance Management Quick Start Tests test suite

Test	Description	Release version
GRC: Create Controls	Validates control creation.	Madrid
GRC: Create Policy Exception and Flow	Create a policy exception and navigate through its states.	Paris
GRC: Policy Lifecycle	Create a policy and navigate through its states.	Paris

To learn more about Policy and Compliance Management, see Policy and Compliance Management.

GRC Risk Management

GRC: Risk Management quick start tests require activating the Risk Management plugin (com.sn_risk) and loading demo data.

GRC: Risk Management Quick Start Tests test suite

Test	Description	Release Version
GRC: Create Profile	Validate profile creation.	Madrid
GRC: Create Risk	Validates risk creation.	Madrid
GRC: Create Control	Validates control creation.	New York
GRC: Create Control Objective and Generate Controls	Validates control objective creation and associates profiles to generate controls.	New York
GRC: Create Issue	Validates issue creation	New York
GRC: Accept an Issue	Validates all the states of an issue till it is closed by accepting the issue.	New York
GRC: Remediate an Issue	Validates issue cannot be closed be with an open remediation task.	New York
GRC: ATF Flow for Indicator (Manual):	Create an indicator template with type manual and associate a control objective to it. Execute one of the indicators thus formed. Mark the state of the indicator task created as closed and result as failed. The associated control will become non-compliant and an issue will be generated.	New York
GRC: ATF flow for Indicator (Basic)	Create an indicator template with type basic and result as failed. Associate a control objective to it and give the supporting data. Execute one of the indicators thus formed. The associated control will become non- compliant and an issue will be generated.	New York
GRC: ATF flow for indicator (Script)	Create an indicator template with type script and enter a script and set the value of result.passed and result.value. Associate a control objective to it. Execute one of the indicators. The associated control will become non-compliant and an issue will be generated.	New York

To learn more about Risk Management, see Risk Management.

Hardware Asset Management

Hardware Asset Management quick start tests are available when you install the Hardware Asset Management (HAM) application from the ServiceNow Store.

Hardware Asset Management test suite

Test	Description	Release version
HAM - Hardware Normalization	Validates the various normalization status values based on the normalized Manufacturer, Product, and Model.	Orlando
HAM - Hardware Asset Disposal work flow	Validates asset disposal work flow.	Paris
HAM - Automating Asset deploy workflow	Validates asset deployment work flow.	Paris
HAM - Automating Asset Swap workflow	Validates Asset Exchange/ Swap work flow. **Note:** Requires demo data.	Paris
HAM - Standard Hardware Asset Request Flow	Validates the Standard Hardware Asset Request flow, which is a part of the Hardware Asset Management application.	Paris
HAM - Hardware Asset Refresh flow	Validates asset refresh workflow.	Quebec
HAM - Loaner Asset Allocation Flow	Validates loaner asset workflow.	Quebec
HAM - Lease Contract Asset Expiration	Validates leased contract asset expiration end to end workflow, performing return or extend of the assets covered in the lease contract.	Quebec
HAM - Asset RMA Flow	Validates the asset Return Merchandise Authorisation (RMA) workflow with Inventory User.	Rome
HAM - Asset RMA Flow (SP)	Validates the asset Return Merchandise Authorisation (RMA) workflow from Service Portal.	Rome
HAM - Loaner Asset Request Flow	Validates the Loaner asset allocation workflow with Inventory Admin user.	Rome

Hardware Asset Management test suite (continued)

Test	Description	Release version
HAM - Contract Renewal Flow	Validates the Contract Renewal flow with Contract Manager user.	Tokyo

To learn more about Hardware Asset Management, see Hardware Asset Management.

HR Service Delivery

HR Service Delivery case tests

Test	Description	Release version
Case Creation Configuration	Validates an HR service has a case creation configuration and is populated into the HR case form when creating an HR case for that HR service.	Xanadu
Create Document template block and block content	Verifies a document template block and document template content are created for block content.	Xanadu
Create HR Employee Document	Creates an employee document and ensures the Attachment [sys_attachment] table points back to the record in the Employee Document [sn_hr_ef_employee_document] table. Verifies that the employee document is accessible from the Employee Center or Employee Center Pro via the employee HR profile.	
Create Employee Document with Active Document Type	Validates the creation of: • One active document type • One inactive document type • Employee file with active document type **Note:** Human Resources Scoped App: Core (com.sn_hr_core) and Employee Document Management (com.sn_employee_document_management) must be activated.	Xanadu

HR Service Delivery case tests (continued)

Test	Description	Release version
[Delegation] Admin - Adding a delegation rule with delegator and delegate user criteria	Verifies that user criteria for a delegation rule works correctly,	San Diego
E2E test for Bulk Case Creation	Verifies that bulk cases can be created for selected users in a specific user segment.	San Diego
ER: Case Restriction	Verifies that restricted cases are only viewable by users with the sn_hr_er.confidential role.	San Diego
HR: Case updates visible to employees	Cases created and updated by an HR agent are visible to the Opened for or Subject person	Quebec
HR: Creation of 401k enrollment case from portal	Creates a 401(k) enrollment case from the Employee Center or Employee Center Pro. Verifies that the case creates successfully.	Madrid
HR: Creation of Employee Relations case from ESC portal	Creates a Disciplinary Issue case from the Employee Center or Employee Center Pro. Verifies that the case creates successfully.	New York
HR: Creation of Employment Verification Letter case from ESC portal	Creates an Employment Verification Letter case from the Employee Center or Employee Center Pro. Verifies that the case creates successfully.	Madrid
HR: Creation of Payroll case from ESC portal	Creates a direct deposit payroll setup case from the Employee Center or Employee Center Pro. Verifies that the case creates successfully.	New York
HR: Creation of Tuition Reimbursement request from portal	Creates a Tuition Reimbursement Request case from the Employee Center or Employee Center Pro.	Madrid

HR Service Delivery case tests (continued)

Test	Description	Release version
	Verifies that the case creates successfully.	
HR: Fulfillment Instructions	Verifies that fulfillment instructions are created for a specific condition for an opened for user. Also verifies the fulfillment instructions change when the details of the case changes.	Orlando
HR: Fulfillment Instructions conditions for COE specific fields	Verifies the conditions for COE specific fulfillment instruction. Fulfillment instructions update on an HR case based on the conditions defined in the fulfillment instructions.	Quebec
HR: General Benefits Inquiry Case Creation	Creates a General Benefits case using the native UI. It also verifies that the case opens and can be updated after case creation.	Madrid
HR: Payroll Discrepancy Case Creation	Creates a Payroll Discrepancy case using the native UI. It also verifies that the case opens and can be updated after case creation.	Madrid
HR: Reclassify Case Transfer	Tests when an HR case transfers from one COE (HR service) to a different COE using the reclassify (HR case number remains the same) method.	Orlando
HR: Response Template Configuration	Creates a response template for a payroll case and pastes the response in the worknotes.	Quebec
HR: Search catalog items & KBs in ESC	Verifies the search functionality in the Employee Center or Employee Center Pro.	Orlando
HR: Standard Case Transfer	Tests an HR case transfers from one COE (HR service) to a different COE using the	Orlando

HR Service Delivery case tests (continued)

Test	Description	Release version
	standard (creates HR case number and deletes old number) case transfer.	
HR: Suspend Case Flow	Tests the ability to suspend an HR case and later resume the case.	Rome
HR: Tuition Reimbursement Case Creation	Creates a Tuition Reimbursement HR case.	Orlando
HR: Workforce Administration Case Creation	Creates a Workforce Administration case, signs an employee verification letter, and generates the letter.	Orlando
JA: Publish & edit plan details	Verifies the ability to delete an existing to-do in an action plan.	San Diego
Journey Designer: Create stage and edit stage	Verifies that the user can create a new stage, add a task template to the stage, edit the stage name, and delete the stage.	Vancouver
Journey Designer: Add task templates from scratch and edit	Verifies that the user can add a task template using the Build from scratch option, then edit the task template.	Vancouver
Journey Designer: Add existing task templates and remove task template	Verifies that a task template can be added to a Journey template with the Use template option, and then removed.	Vancouver
Learning Core Task Assignment	Verifies a learning task from a learning catalog can be assigned to a user.	San Diego

Note:
Requires plugin activation of:

- Human Resources Scoped App: Core plugin (com.sn_hr_core)
- Employee Service Center [com.sn_hr_service_portal]

Lifecycle Events: Tests for verifying Lifecycle Events

Test	Description	Release version
Configure HR Service for Auto Case Closure	Verifies that Lifecycle event cases are automatically closed.	Orlando

Lifecycle Events: Tests for verifying Lifecycle Events (continued)

Test	Description	Release version
Create Employee request and associate it to LE case	Creates and validates that an employee request appears in the Lifecycle Event case action list.	Rome
Create Employee request for old ticket page	Validates if employee requests can be created from the legacy ticket page.	Xanadu
[Delegation] Admin - Adding a delegation rule with delegator and delegate user criteria	Verifies user criteria is working for delegation rules for a delegator and delegate user.	Xanadu
HR Lifecycle: Access Ticket page for a NewHire Onboarding Case in Employee Service Center	Verifies creation of a new hire and can access the Ticket page of the Employee Service Center.	Orlando
HR Lifecycle: Access ESC ticket page and todos page and complete assigned todos	Creates a Tuition Reimbursement Request case from Employee Service Center and the Complete NDA to-do from the subject person.	Orlando
HR LifeCycle: Request Onboarding Case Creation	Creates a Request Onboarding case using the native UI and updates the case after creation.	Madrid
HR Lifecycle: Trigger Rescind Workflow	Triggers the Rescind workflow for a New Hire Onboarding Lifecycle Event case.	Quebec
HR Lifecycle: Verify Requests page for Open and Closed cases	Verifies open and closed cases on the Requests page.	Orlando
HR Lifecycle: Verify search in Requests Page	Verifies the search functionality on the Requests page for open and closed cases.	Orlando
[LE]: Add Ad hoc task	Verifies an ad hoc task can be added for a Lifecycle Event case.	Xanadu

Note:
Requires plugin activation of:

- Human Resources Scoped App: Core plugin (com.sn_hr_core)
- Human Resources Scoped App: Lifecycle Events [com.sn_hr_lifecycle_events]
- Human Resources Scoped App: Lifecycle Events for Enterprise plugin [com.sn_hr_lifecycle_ent]
- Employee Service Center [com.sn_hr_service_portal]

KB: Knowledge block tests

Test	Description	Release version
Create knowledge block and attach block to article	Creates a knowledge block, publishes the block, and attaches the block to a knowledge article.	New York
Preview Knowledge Articles with Knowledge Blocks	Tests for preview of knowledge article that contains knowledge blocks.	New York

Note:
Requires plugin activation of Human Resources Scoped App: Core plugin (com.sn_hr_core)

To learn more about HR Service Delivery, see HR Service Delivery.

Incident Management

Incident Management quick start tests require activating the Incident Management - ATF Tests plugin (com.snc.incident.atf).

Incident Management test suite

Test	Description	Release version
INCIDENT MGMT: Incident Resolution SLA	Test to verify the Incident Resolution SLA baseline functionality.	Madrid
INCIDENT MGMT: Incident Response SLA	Test to verify the Incident Response SLA baseline functionality.	Madrid
INCIDENT MGMT: Copy Incident	Test to verify whether the fields from the original Incident are copied correctly to the new Incident.	Madrid
INCIDENT MGMT: Copy Incident from a Closed Incident	Test to verify that the **Copy Incident** UI action is visible for closed Incidents.	Madrid
INCIDENT MGMT: Create Standard Change from Incident	Test to verify the creation of a Standard Change from an Incident.	Madrid
INCIDENT MGMT: Create of Emergency Change from an Incident	Test to verify the creation of an Emergency Change from an Incident.	Madrid
INCIDENT MGMT: Create Normal Change from an Incident	Test to verify the creation of a Normal Change from an Incident.	Madrid
INCIDENT MGMT: Create Problem from an Incident	Test to verify the creation of a Problem from an Incident.	Madrid

Incident Management test suite (continued)

Test	Description	Release version
INCIDENT MGMT: Create Knowledge from an Incident	Test to verify the creation of a Knowledge from an Incident.	Madrid
INCIDENT MGMT: Incident State flow	Test to verify the state flow of an incident.	Madrid
INCIDENT MGMT: Reopening an Incident	Test to verify the reopen incident functionality.	Madrid
INCIDENT MGMT: Incident Assignment	Test to verify the incident assignment functionality.	Madrid
INCIDENT MGMT: Create child Incident using UI action and verify its fields	Test to verify the creation of a child Incident from an Incident through the **Create Child Incident** UI action. The test also verifies that the fields of the child Incident get copied correctly from the parent incident to the child incident.	Madrid
INCIDENT MGMT: Incident creation — Self Service	Test to verify the creation of an Incident using the **Create Incident** catalog item.	Madrid
INCIDENT MGMT: Parent and Child Incident state sync up	Test to verify that the state of a child Incident synchronizes with the parent Incident when the child Incident is created.	Madrid
INCIDENT MGMT: Parent and child Incident state sync up after reopening an Incident	Test to verify that the state of a child Incident synchronizes with the parent Incident when the parent Incident is reopened.	Madrid
INCIDENT MGMT: Verify creation of knowledge article from an Incident	Test to verify the creation of a knowledge article from an Incident using the **Create Knowledge** UI action on the Incident form. The UI action is visible when you activate the KCS Integration for Incident Management plugin (com.snc.incident.knowledge).	Orlando

To learn more about Incident Management, see Incident Management.

Incident Management in Service Operations Workspace

Incident Management in Service Operations Workspace quick start tests are available in the base system when you install or update to Service Operations Workspace ITSM Applications (sn-sow-itsm-cont) 6.0 version.

Incident Management in Service Operations Workspace test suite

Test	Description	Release version
SOW Incident: Create problem from incident	Test to verify the creation of a problem record from an Incident using the **Create Problem** UI action on the Incident form.	Xanadu (August store release, 6.0 version)
SOW Incident: Verify Assign to me button functionality	Test to verify the assignment of the incident record to the logged-in user using the **Assign to me** UI option on the Record information side panel of the Incident form.	Xanadu (August store release, 6.0 version)

To learn more about Incident Management in Service Operations Workspace, see Incident Management in Service Operations Workspace.

Integration Commons for CMDB

CMDB INT: CMDB Integrations Validation test suite

Test suite to verify the integrity of an integration using multiple tests.

Test	Description	Release version
CMDB INT: Set Test Session Application	Modify the run server-side script to set an application name so that you can test only one integration. Otherwise, all integrations installed will be tested.	Paris
CMDB INT: Test Against Source Analysis	Test an integration against the values in the CMDB Integration Source Analysis [sn_cmdb_int_util_cmdb_integration_source_analysis] table.	Paris
CMDB INT: Validate Application Feed	Validate all application feeds in an integration.	Paris
CMDB INT: Validate Discovery Source	Validate that the discovery source exists.	Paris
CMDB INT: Validate Entity Mappings	Validate all entity mappings of an integration.	Paris
CMDB INT: Validate Fields	Validate fields for CMDB Integrations.	Paris
CMDB INT: Validate Lookups	Validate CMDB integration lookups.	Paris
CMDB INT: Validate Mandatory Operations	Validate that all integrations for mandatory operations exist for mapped fields.	Paris

CMDB INT: CMDB Integrations Validation test suite

Test suite to verify the integrity of an integration using multiple tests.

(continued)

Test	Description	Release version
CMDB INT: Validate Operations	Validate all operations for an integration.	Paris
CMDB INT: Validate References	Validate CMDB integration references.	Paris
CMDB INT: Validate Related Entries	Validate all related classes against the data dictionary for related entries.	Paris
CMDB INT: Validate Relationships	Validate CMDB integration relationships.	Paris

To learn more about Integration Commons for CMDB, see Integration Commons for CMDB.

Investment Funding

The Investment Funding quick start tests require activating the Investment Funding - ATF Tests plugin (com.snc.investment_planning.atf).

Investment Funding quick start tests

Test	Description	Release version
Validation of top-down funding and unfunding	• Validate that the correct amount is allocated from top investment to child investments while funding. • Validate that the correct amount is returned back to the top investment from the child investment while unfunding.	Orlando
Validation of bottom-up funding	• Validate that the correct amount is requested by the child investment from the parent investment. • Validate that the correct amount is allocated back to the child investment from the parent investment.	Paris

To learn more about Investment Funding, see Investment Funding.

Knowledge Management

Knowledge Management quick start tests require activating the Knowledge Management Core plugin (com.glideapp.knowledge), the Knowledge Management Advanced Installer plugin (com.snc.knowledge_advanced.installer), the Knowledge Blocks plugin (com.snc.knowledge_blocks), and the Customer Service Management Demo Data plugin (com.snc.customerservice.demo).

KM: Knowledge Management test suite

Test	Description	Release version
KM : Create KCS Template Article with Approval Publish workflow	Verify the creation of a KCS template article with approval publish workflow. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Madrid
KM: Create MultiVersioned standard Article with approval publish workflow	Verify the creation of a multi-versioned standard article with approval publish workflow. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Madrid
KM: Article level subscription	Verify that users can subscribe to a knowledge article. **Note:** Requires the Knowledge Management Advanced Installer plugin.	New York
KM: User criteria covering canRead and canContribute for KnowledgeBase and canRead at Article level	Verify the creation of canRead and canContribute user criteria for knowledge base and canRead user criteria for article. **Note:** Requires the Knowledge Management Core plugin.	Madrid
KM: Create a KCS article from a case	Verify the creation of a KCS article from a case.	Madrid

KM: Knowledge Management test suite (continued)

Test	Description	Release version
	Note: Requires the Knowledge Management Advanced Installer and Customer Service Management Demo Data plugins.	
KM: Knowledge Base Level Subscriptions	Verify that users can subscribe to a Knowledge Base. **Note:** Requires the Knowledge Management Advanced Installer plugin.	New York
KM: AQI	Verify the creation, assignment, and review of an AQI checklist. **Note:** Requires the Knowledge Management Advanced Installer plugin.	New York
KM: Requestor performs search, view the article, provide feedback (Helpful No with Feedback Task)	Verify the search request, review, provision of feedback as not helpful, and creation of a feedback task for an article on the Knowledge Management Service Portal. **Note:** Requires the Knowledge Management Advanced Installer plugin.	New York
KM: Create and edit an article from Feedback Task form	Confirm the availability of the **Create Article** and **Edit Article** buttons on a Knowledge Feedback Task form. **Note:** Requires the Knowledge Management Advanced Installer plugin.	New York

KM: Knowledge Management test suite (continued)

Test	Description	Release version
KM: Search for pinned articles and checking the highest click rank	Verify the search request and that the click rank value of a pinned article was added to the Knowledge Searches (ts_query_kb) table. **Note:** Requires the Knowledge Management Advanced Installer plugin.	New York
KM: Create an Ownership Group and check edit permissions and subscriptions	Verify the assignment of an article to an ownership group, that all members of the ownership group are subscribed to the article, and have permission to edit. **Note:** Requires the Knowledge Management Advanced Installer plugin.	New York
KM: Create, search for, and view knowledge articles with knowledge blocks	Verify the creation of a knowledge block and its addition to a knowledge article. Also, verify that the knowledge block content is searchable. **Note:** Requires the Knowledge Blocks plugin.	Orlando
KM: Validate the Knowledge - Approval Retire workflow	Verify the success of Knowledge - Approval Retire workflow by retiring a published knowledge article. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Orlando
KM: Validate feedback task assignment to the ownership group manager	Verify the assignment of a feedback task to the ownership group manager of a knowledge article.	Orlando

KM: Knowledge Management test suite (continued)

Test	Description	Release version
	Note: Requires the Knowledge Management Advanced Installer plugin.	
KM: Validate that the article template selector includes knowledge bases and article templates	Verify that the article template selector includes all knowledge bases with contribute access and article templates applicable to each knowledge base. **Note:** Requires Knowledge Management Advanced Installer.	Orlando
KM: Validate the Knowledge - Approval Publish workflow	Verify the success of a Knowledge - Approval Publish workflow by recalling, rejecting, and then approving a knowledge article through the approval process. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Orlando
KM: Make an outdated version of an article the latest version	Verify that an outdated version of a knowledge article is made the latest version. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Orlando
KM: Validate that the mapped related articles appear in the Related articles widget	Verify that the related articles mapped to a knowledge article appear in the Related articles widget on the knowledge article view page. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Orlando

KM: Knowledge Management test suite (continued)

Test	Description	Release version
KM: Validate that the mapped related catalog items appear in the Related items widget	Verify that the related catalog items mapped to a knowledge article appear in the Related items widget on the knowledge article view page. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Orlando
KM: Validate that a versioned article is created, published, checked out, and retired	Verify the success of UI actions on a versioned knowledge article by creating, publishing, updating, and retiring the knowledge article in Agent Workspace. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Quebec
KM: Verify the scheduled publish feature for knowledge articles	Verify the scheduled publish feature for knowledge articles by creating an article in approval publish workflow, putting it for scheduled for publish and observing the change in workflow state from the**Scheduled for publish** state to the **Published** state. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Quebec
KM: Validate the Knowledge - Approval Publish workflow in Agent Workspace	Verify the success of a Knowledge - Approval Publish workflow by recalling and then approving a knowledge article through the approval process in Agent Workspace. **Note:** Requires the Knowledge Management Advanced Installer plugin.	Quebec

To learn more about Knowledge Management, see Knowledge Management.

Leader Hub

Leader Hub quick start tests require activating the Leader Hub plugin (sn_egd_lh) and loading demo data.

Leader Hub quick start tests

Test	Description	Release version
[Leader hub] - Validate LH flows	• Verifies that the following widgets are present on the At a glance page for user Maria Davies: ○ Org talent ○ Skill growth area ○ Growth engagement • Verifies the employee card for the leader who has accessed Leader Hub appears at the top of the organizational chart on the Org talent page. • Verifies that selecting the **View team details** button on a card for an employee navigates the leader to the Team details page. • Verifies that the Skills widget is present on the Org Skills page. **Note:** Requires demo data.	Xanadu

To learn more about Leader Hub, see Leader Hub.

Legal Request Management

Legal Request Management quick start tests require installing the following apps:

- Legal Request Management (sn_lg_ops)
- Legal Counsel Center (sn_lg_workspace)

Legal Request test suite

Test	Description	Release version
LRM: Privacy Request Test	Tests to validate submission and fulfillment of Privacy legal requests.	Utah

To learn more about Legal Request Management, see Legal Request Management.

Major Incident Major Incident Management

Major Incident Management quick start tests require activating the Incident Management - Major Incident Management plugin (com.snc.incident.mim).

MIM: Major Incident Management test suite

Test	Description	Release version
MIM: Create a Major Incident	Test to verify the creation of major incident from the application navigation module.	Orlando
MIM: Create a Major Incident Candidate	Test to verify the **Create a Major Incident Candidate** module from navigation menu.	Orlando
MIM: Create a Major Incident from a Major Incident Candidate	Test to verify the creation of a major incident from a major incident candidate.	Orlando
MIM: Major Incident Candidate creation when it matches Major Incident Management trigger rule	Test to verify the creation of a major incident candidate when conditions to create a major incident match the major incident management trigger rule.	Orlando
MIM: Propose a major Incident (Assignment group empty)	Test to verify auto assignment of major incident candidate when an incident is proposed as a major incident and the **Assignment group** and the **Assigned to** fields are empty.	Orlando
MIM: Propose a major Incident (Assignment group is not empty)	Test to verify the auto assignment of a major incident when an incident is proposed as major incident and the **Assignment Group** and the **Assigned to** fields are not empty.	Orlando
MIM: Promote a candidate to major Incident (Assignment group is empty)	Test to verify auto assignment of major incident when a candidate is promoted to a major incident and the **Assignment Group** is empty.	Orlando
MIM: Promote a candidate to major Incident (Assignment group is not empty)	Test to verify auto assignment of major incident when a candidate is promoted to a major incident and the **Assignment group** is not empty.	Orlando

MIM: Major Incident Management test suite (continued)

Test	Description	Release version
MIM: Reject a Major Incident Candidate	Test to verify the rejection of a major incident candidate.	Orlando
MIM: Demote a Major Incident	Test to verify whether a major incident gets demoted to an incident.	Orlando
MIM: Major Incident closure	Test to validate the major incident closure functionality.	Orlando
MIM: State sync up with ICP and ICT	Test is to verify the state sync up with Incident Communication Plan and Incident Communication Task.	Orlando
MIM: ICP attached to an Incident based on conditions and its state sync up with ICT	Test is to verify Incident Communication Plan attached to an Incident based on defined conditions.	Orlando
MIM: Resolving MI and PIR	Test is to verify resolving a Major Incident and Post Incident Report.	Orlando
MIM: Major Incident workbench layout verification	Test is to verify Major Incident workbench layout verification.	Orlando
MIM: Verify communication task from MI workbench	Test is to verify communication task from MI workbench.	Orlando

To learn more about Major Incident Management, see Major Incident Management.

On-Call Scheduling

Activate the On-Call Scheduling plugin (com.snc.on_call_rotation) to run the On-Call Scheduling quick start tests.

On-Call: On-Call scheduling ATF Suites

Test	Description	Release version
On-Call: Verify request time-off when PTO approval preferences is Not allowed	Verify whether you can request a time-off when PTO approval preferences is set to **Not allowed**	Orlando
On-Call: Create overlapping shifts without selecting a template	Verify whether you can create overlapping shifts without selecting a template.	Orlando
On-Call: Create overlapping shifts when allow shift overlap is set to No	Verify whether you can create overlapping shifts when **Allow overlap** is set to **No**.	Orlando

On-Call: On-Call scheduling ATF Suites (continued)

Test	Description	Release version
On-Call: Request time-off when PTO approval preferences is with approval	Verify whether you can request a time-off when PTO approval preference is **With approval**.	Orlando
On-Call: Assign shift managers for maintaining on-call schedules.	Verify whether shift managers can maintain on-call schedules.	Orlando
On-Call: Search on-call schedules by user name	Verify whether you can search for on-call schedules by user name.	Orlando
On-Call: Search on-call schedules by group name or schedule name	Verify whether you can search for on-call schedules by group name or schedule.	Orlando
On-Call: Roster details for overlapping shifts with the escalation rule 'Escalate to incoming shift'	Verify the roster details for overlapping shifts when the escalation rule is set to **Escalate to incoming shifts**.	Orlando
On-Call: Calendar preview with timezone options	Verify whether the calendar preview is available with the timezone options.	Orlando
On-Call: Create overlapping shifts by selecting a template	Verify whether you can create overlapping shifts by selecting a template.	Orlando
On-Call: Show active shifts		Orlando
On-Call: Make shift draft/ publish form shift form	Verify whether you can publish/unpublish a shift and alternate between making the rota **Draft** and **Publish** state from the form.	Orlando
On-Call: Roster Details for overlapping shifts with the escalation rule 'Escalate to outgoing shift'	Verify the roster details for overlapping shifts when the escalation rule is set to **Escalate to outgoing shifts**.	Orlando
On-Call: Roster Details for overlapping shifts with the escalation rule 'Escalate to all shifts'	Verify the roster details for overlapping shifts when the escalation rule is set to **Escalate to all shifts**.	Orlando
On-call: Verify On-Call workbench	Test to verify the On-Call workbench layout.	Orlando
On-call: Hide or show shifts	Verify whether you can hide or show shifts from On-Call calendar	Orlando

To learn more about On-Call Scheduling, see On-Call Scheduling.

Metric Intelligence

Metric Intelligence quick start tests require activating the Metric Intelligence plugin (com.snc.sa.metric).

Metric Intelligence quick start tests

Test	Description	Release version
OI: Health Metrics Collection	An individual test that verifies the following: • There is at least one Metric Intelligence Metrics extension that is running. • There are entries for the 'Health Metrics' in the Metric to CI table. • For each of the Metric Intelligence Metrics extensions that are currently running, that there is data stored in MetricBase. If the test initially fails, wait until the Metric Intelligence Metrics extension runs for at least 10-15 minutes. Then try again.	Madrid

To learn more about Metric Intelligence, see Metric Intelligence.

Predictive Intelligence

Predictive Intelligence quick start tests require activating the Predictive Intelligence [com.glide.platform_ml] plugin. In order to execute critical upgrade tests on existing machine learning solutions, you need to create a basic authorization profile named ml_atf in the Basic Auth Configurations table (sys_auth_profile_basic.list). To run the tests successfully, the user attached to the ml_atf authorization profile must have the ml_admin role.

Predictive Intelligence: Classification and Similarity Solution Prediction test suite

Test	Description	Release version
PI: Presence of ML model artifacts persisted in glide	Verify all the trained ML model artifacts are persisted in glide (sys_attachments table) after model training/instance cloning so that ML prediction calls are successful.	New York
PI: Valid setup of ML user (sharedservice.worker) in glide	Validate if the ML user in glide (sharedservice.worker) is active and not logged out	New York

Predictive Intelligence: Classification and Similarity Solution Prediction test suite (continued)

Test	Description	Release version
	so that model training is successful.	
PI: Glide upgrade test for Classification solution	Validate that the classification model prediction on existing active models is producing the same class membership and confidence value results after a glide upgrade.	New York
PI: Glide upgrade test for Similarity solution	Validate that the similarity model prediction API calls on active models are successful after a glide upgrade.	New York

Problem Management

Problem Management quick start tests require activating the Problem Management Best Practice — Madrid plugin (com.snc.best_practice.problem.madrid) and the Problem Management — ATF Tests plugin (com.snc.problem.atf). For all state related test, the Problem Management State Model (com.snc.best_practice.problem.madrid.state_model) plugin needs to be active.

PRB MGMT: Problem Management test suite

Test	Description	Release version
PRB MGMT: Cancel a Problem when the state of the Problem is Assess	Verify that when a Problem in the **Assess** state is canceled, the state of the Problem changes to **Closed** with **Resolution code** as **Canceled**.	Madrid
PRB MGMT: Cancel a Problem when the state of the Problem is Root Cause Analysis	Verify that when a Problem is in the **Root Cause Analysis** state and is canceled, the state of the Problem changes to **Closed** with **Resolution code** as **Canceled**.	Madrid
PRB MGMT: Mark a Problem as Duplicate when the state of the Problem is Assess	Verify that when a Problem is in the **Assess** state and is marked as duplicate, the state of the Problem changes to **Closed** with **Resolution code** as **Duplicate**.	Madrid
PRB MGMT: Mark a Problem as Duplicate when the state of the Problem is Root Cause analysis	Verify that when a Problem is in the **Root Cause Analysis** state and is marked as duplicate, the state of the Problem changes to **Closed** with **Resolution code** as **Duplicate**.	Madrid

PRB MGMT: Problem Management test suite (continued)

Test	Description	Release version
PRB MGMT: Accept Risk of Problem (problem.acceptrisk.move_to_closed=false, Fix in Progress)	Verify that when a Problem state is **Fix in Progress** and the risk is accepted, then the Problem state changes to **Resolved** with **Resolution code** as **Risk Accepted**. **Note:** The test is valid when Problem property **Accept Risk moves the Problem to Closed state instead of Resolved state** *(problem.acceptrisk.move_to_closed)* is false.	Madrid
PRB MGMT: Accept Risk of Problem (problem.acceptrisk.move_to_closed=true, Fix in Progress)	Verify that when a Problem state is **Fix in Progress** and the risk is accepted, then the Problem state changes to **Closed** with **Resolution code** as **Risk Accepted**. **Note:** The test is valid when Problem property **Accept Risk moves the Problem to Closed state instead of Resolved state** *(problem.acceptrisk.move_to_closed)* is true.	Madrid
PRB MGMT: Accept Risk of Problem (problem.acceptrisk.move_to_closed=false, Root Cause Analysis)	Verify that when a Problem state is **Root Cause Analysis** and the risk is accepted, then the Problem state changes to **Resolved** with **Resolution code** as **Risk Accepted**. **Note:** The test is valid when Problem property **Accept Risk moves the Problem to Closed state instead of Resolved state** *(problem.acceptrisk.move_to_closed)* is false.	Madrid
PRB MGMT: Accept Risk of Problem(problem.acceptrisk.move_to_closed=true, Root Cause Analysis)	Verify that when a Problem state is **Root Cause Analysis** and the risk is accepted, then	Madrid

PRB MGMT: Problem Management test suite (continued)

Test	Description	Release version
	the Problem state changes to **Closed** with **Resolution code** as **Risk Accepted**. **Note:** The test is valid when Problem property **Accept Risk moves the Problem to Closed state instead of Resolved state** *(problem.acceptrisk.move_to_closed)* is true.	
PRB MGMT: Reanalyze Problem which is closed-Risk Accepted from state Root Cause Analysis	Verify that when a Problem is reanalyzed after it is **Closed** with the **Resolution code** as **Risk Accepted**, Problem state changes to **Root Cause Analysis**.	Madrid
PRB MGMT: Create Emergency Change from Problem	Verify the creation of Emergency Change from a Problem.	Madrid
PRB MGMT: Create Normal Change from Problem	Verify the creation of Normal Change from a Problem.	Madrid
PRB MGMT: Problem State Management	Verify problem state management.	Madrid
PRB MGMT: Reanalyze a Problem from Complete	Verify that when a Problem is reanalyzed after it is **Closed** with the **Resolution code** as **Fix Applied**, Problem state changes to **Root Cause Analysis**.	Madrid
PRB MGMT: Reanalyze a Problem which is canceled from state Assess	Verify that when a Problem is reanalyzed after it is **Closed** with the **Resolution code** as **Canceled**, Problem state changes to **Root Cause Analysis**.	Madrid
PRB MGMT: Reanalyze a Problem which is canceled from state Root Cause Analysis	Verify that when a Problem is reanalyzed after it is **Closed** with the **Resolution code** as **Canceled**, Problem state changes to **Root Cause Analysis**.	Madrid
PRB MGMT: Reanalyze Problem which is closed-Risk Accepted from state Fix in progress	Verify that when a Problem is reanalyzed after it is **Closed** with the **Resolution code** as **Risk Accepted**, Problem	Madrid

PRB MGMT: Problem Management test suite (continued)

Test	Description	Release version
	state changes to **Root Cause Analysis**.	
PRB MGMT: Create a Known Error article from Problem	Verify creation of Known Error article from a Problem.	Madrid
PRB MGMT: Risk Accept reason on Incident	Verify the **Risk Accepted reason** is copied to the Incidents which are awaiting resolution of a Problem.	Madrid
PRB MGMT: Communicate Fix	Verify the communicate fix functionality.	Madrid
PRB MGMT: Communicate Workaround	Verify the communicate workaround functionality.	Madrid
PRB MGMT: Fix information on Incident	Verify that when a Problem is resolved, the state of the Incidents that are awaiting resolution of the Problem changes to **Resolved**. The fix notes of the Problem are copied to the Incidents.	Madrid
PRB MGMT: Problem task (Type:General) state management	Verify Problem task state management of a general type Problem.	Madrid

To learn more about Problem Management, see Problem Management.

Project Portfolio Management

Project Portfolio Management quick start tests require activating the PPM Standard - ATF Tests plugin (com.snc.financial_planning_pmo.atf).

PMO: Financial Tests for verifying cost rollups and demand to project conversion test suite

Test	Description	Release version
PMO: Verify cost plan roll up to project/demand and program	Validate the total planned cost rollup from project and demand to program.	Madrid
PMO: Verify cost plan roll up to project/demand, program and portfolio	Validate the total planned cost rollup from project and demand to portfolio.	Madrid
PMO: Verify financials of Project created from Demand - Simple Financials	Validate the financial tab fields of a project created from a demand.	Madrid
PMO: Verify financials of Project created from Demand	Validate the budget, cost plan, and benefit plan of a project created from a demand.	Madrid

PMO: Financial Tests for verifying cost rollups and demand to project conversion test suite (continued)

Test	Description	Release version
- With budget, cost plans, benefit plans		
PMO: Verify ETC/EAC at cost plan breakdown and project level	Validate the EAC and ETC values at cost plan breakdown and project level.	Tokyo

PMO: Innovation Management Tests for verifying idea life cycle

Test	Description	Release version
Validate state changes of Idea	Validate that the state of idea is changed when an idea is accepted, rejected, or any other task is created from an idea.	San Diego

PMO: Project Management tests for validating basic life cycle and project rollups test suite

Test	Description	Release version
Validate PPM Cycle from Ideation to demand to project closure	Validate the flow of creating an idea, converting the idea to a demand, and then converting the demand to a project. **Note:** This test fails if the PPM Standard Multicurrency (com.snc.ppm_multicurrency) plugin is active.	Madrid
Validation of State and Date Rollup for Automatic Project	Validate the date and state rollup from tasks for a project of type Automatic.	Madrid
Validation of State and Date rollup for Manual Project	Validate the date and state rollup from tasks for a project of type Manual.	Madrid
Validation of project percent complete when all tasks are Closed Incomplete	Validate the project percent complete when all the tasks are closed as Closed Incomplete.	Orlando
Validate Project is not 100 percent complete if it has atleast one task as closed incomplete	Validate the project percent complete is not 100% when at least one of the tasks is closed as Closed Incomplete or Closed Skipped with task % complete less than 100%.	Orlando

PMO: Project Management tests for validating basic life cycle and project rollups test suite (continued)

Test	Description	Release version
Validate waterfall project does not show Agile Planning Board	Validate that the waterfall projects cannot use the following: • Add stories or epics • Add agile phase • Access Agile board from the project	Orlando
Validate dates are rolled up from existing projects to program	Validate that the start and end dates of the project are rolled up to the program to which the project belongs.	Paris
Validate dates are rolled up from new projects to program	Validate that the start and end dates of a new project are rolled up to the program to which the project belongs.	Paris
Add projects and demands with risks or issues to program	Validate that the projects and demands with risks or issue records associated with them are added to the program.	Paris
Add projects and demands with benefit plans to program	Validate that the projects and demands with benefit plans associated with them are added to the program.	Paris
Add projects and demands with cost plans to program	Validate that the projects and demands with cost plans associated with them are added to the program.	Paris
Validate program dates on addition of existing demands	Validate that the start and end date of the program are adjusted on addition of an existing demand.	Paris
Validate program dates on addition of new demands	Validate that the start and end date of the program are adjusted on addition of a new demand.	Paris
Validate program state rollup	Validate that the program state is rolled up from state of all the projects in the program.	Paris
Verify Demand task due date column field value empty	Validate that the **Due date** field is empty for a new demand task.	Rome
Verify baseline is created on Project & Demand when demand is converted to project	Validate that a project and demand baseline is created when a demand is converted to a project.	Rome
Verify timecard financial appears on Demand when time card submitted against demand task	Validate that the actual cost of the demand is updated when a time card is processed for a demand task.	Rome
Validate demand approved date	Validate that the approved dates for demand and the project created from the demand are the same.	Tokyo

PMO: Project Management tests for validating basic life cycle and project rollups test suite (continued)

Test	Description	Release version
Validate project preference "close project milestone tasks when they change to work state"	Validate that the milestone tasks are closed when their state is changed to Work in Progress after setting the project preference to "close project milestone tasks when they change to work state".	Tokyo
Validate project preference "Enable move project for WIP projects"	Validate that the project start date is changed appropriately for a project in WIP state when the project start date is changed after setting the project preference to "Enable move project for WIP projects".	Tokyo
Validate project preference "Rollup project start date from tasks"	Validate that the start date of a project task does not roll up to the project when the project planned dates are different than the project task start date after deselecting the project preference "Rollup project start date from tasks".	Tokyo

Child Test Suite: Validation of Move Project Action

Test	Description	Release version
Validate Planned start date of a project can be shifted by using the Move project action	Validate that the Planned start date of a project in Planning or Open state can be updated to a later or earlier date than the current planned start date using the Move project related link.	Orlando
Validate Move Project is disabled when Project is selected for execution	Validate that the Move Project option is not available if the Project is in Execution phase.	Orlando
Validate Move Project functionality with sprint dates populated for an agile phase	Validate that the sprint start and end dates are cleared when the project is moved using the Move Project related link. **Note:** This test is available only when Agile Development 2.0 plugin (com.snc.sdlc.agile.2.0) is activated.	Orlando
Validate Move Project functionality with external dependencies and related entities	Validate that the external dependencies and related entities are also shifted and redrawn when the project is moved using the Move Project related link.	Orlando

Child Test Suite: Validation of Move Project Action (continued)

Test	Description	Release version
Validate Move Project functionality with different project states	Validate that the Move Project does not work when the project is in Work In Progress or Closed Complete state.	Orlando

Child Test Suite: Verify RIDAC flow of a Project

Test	Description	Release version
Verify RIDAC flow for Risk	Validate the RIDAC flow for a risk associated with a project.	Quebec
Verify RIDAC flow for Issue	Validate the RIDAC flow for an issue associated with a project.	Quebec
Verify RIDAC flow for Decision	Validate the RIDAC flow for a decision associated with a project.	Quebec
Verify RIDAC flow for Action	Validate the RIDAC flow for an action associated with a project.	Quebec
Verify changes in fields of risk form	Validate that any change is the Risk form fields such as Risk rank, Risk value, and Probability is updated successfully upon submission.	Quebec

Child Test Suite: Verify RIDAC flow of a Demand

Test	Description	Release version
Verify RIDAC flow for Risk	Validate the RIDAC flow for a risk associated with a demand.	Quebec
Verify RIDAC flow for Issue	Validate the RIDAC flow for an issue associated with a demand.	Quebec
Verify RIDAC flow for Decision	Validate the RIDAC flow for a decision associated with a demand.	Quebec
Verify RIDAC flow for Action	Validate the RIDAC flow for an action associated with a demand.	Quebec
Verify changes in fields of risk form	Validate that any change is the Risk form fields such as Risk rank, Risk value, and Probability is	Quebec

Child Test Suite: Verify RIDAC flow of a Demand (continued)

Test	Description	Release version
	updated successfully upon submission.	

PMO: Resource Management tests for verifying the resource plan flows test suite

Test	Description	Release version
Verify user resource plan flow from Planned to Canceled state	Validate that the resource plan of a project can be moved to canceled state from planned state.	Madrid
Verify group resource plan flow from Planned to Complete state	Validate that the resource plan of a project can be moved to complete state from planned state.	Madrid
Verify role resource plan flow from Planned to Allocated state	Validate that the resource plan of a project can be moved to allocated state from planned state.	Madrid
Verify aggregated cost of all resource plans roll up to the corresponding project or demand fields	Validate that the aggregated cost of all resource plans on a project or demand roll up to the Planned Cost and Allocated Cost fields and the Resource Cost section of respective projects and demands.	Orlando
Verify Copy Resource plan option	Validate that the Copy Resource plan option creates an exact copy of the source resource plan in the Planning state	Orlando
Verify resource plan aggregate roll up from project/ demand to program	Validate that the aggregated cost of all resource plans on a project or demand roll up to the total planned cost of the associated program.	Orlando
Verify records on completion of a resource plan	Validate the changes in a resource plan on completion: • The state of the resource plan is updated to Completed. • If the completion date is earlier than the resource plan end date, the end date of the resource plan is updated with the completion date. If the completion date entered is later than the resource plan end date, the resource plan end date is retained. • All the requested and allocation records for the resource plan for the period after the completion date are deleted. If there are any actual hours logged against an allocation, that allocation is not deleted. For those allocation records,	Orlando

PMO: Resource Management tests for verifying the resource plan flows test suite (continued)

Test	Description	Release version
	the allocated hours become zero and the actual hours are retained. • The available and allocated hours for resources are updated in the aggregate tables.	
Verify records on completion of a resource plan with Planned Duration as allocation type	Validate the following on completion of a resource plan with Planned Duration as allocation type: • The state of the resource plan is updated to Completed. • Allocations are not deleted. • End date of the allocation is updated to the completion date.	Orlando
Verify the RP replan Capability	Validate that when a cancelled resource plan is re-planned, the state of the resource plan changes to Planning.	Orlando
Verify whether change in resource plan is reflected in corresponding cost plan	Validate that when a resource plan is updated, the corresponding cost plan is updated accordingly. For example, if the total planned cost is 500 USD, and the planned hours is 10, and you change the planned hours to 20, the total planned cost is updated to 1000 USD.	Paris
Resource-Test the default population of resource plan start & end date	Validate the following on creating a user or group resource plan from the related list of a demand: • If a demand is created without a start date and end date, the user or group resource plan has task as demand and no start and end date. • If a demand is created with a start date and end date, the user resource plan has task as demand and the start date and end date as added for the demand.	Paris
Validate that actual hours in operational resource plan and time card are equal	Validate that when a time card category is mapped with an operational work type, on submitting the time card for the operational resource plan associated with that work type, the actual hours in the resource plan and the time card are equal.	Quebec
Verify Resource plan auto population for Operational plans	Validate that the operational resource plans associated with a time card are automatically retrieved on the time card when time is logged.	Rome
Verify resource plan auto population for non-operational resource plans	Validate that the resource plans associated with a project, project task, or demand for a time card are automatically retrieved on the time card when time is logged.	Rome

To learn more about Project Portfolio Management, see Project Portfolio Management.

Project Currency test suite

Project currency quick start tests require activating the PPM Standard Multicurrency – ATF Tests plugin (com.snc.ppm_multicurrency.atf).

Product currency test suite tests

Test	Description	Release version
Verify cost in project currency on cost plan	Validate the calculation of cost line breakdown with budget reference rate and verify roll up to cost plan and also for the project in project currency.	Orlando
Verify benefit in project currency on benefit plan	Validate the calculation of benefit line breakdown with budget reference rate and verify roll up to benefit plan and also for the project in project currency.	Orlando

To learn more about PPM Standard, see PPM Standard.

Reporting

The Reporting quick start test Automated Test Framework - Reporting plugin (com.glide.automated_testing_impl.report) is active by default or instance reboot.

Note:
Reporting quick start tests do not test report access from dashboards. To test dashboards, see Quick start tests for Dashboards.

Reporting quick start tests

Test	Description	Release version
Report Visibility	Confirm whether reports are still visible to users whom they are shared with.	Madrid

Software Asset Management Foundation plugin Software Asset Management

Software Asset Management Foundation plugin Software Asset Management quick start tests require activating the Software Asset Management Professional plugin (com.snc.samp). Some quick start tests require activating the following additional plugins.

- Software Asset Management - Spend Detection (com.sn_sam_spend)
- Software Asset Management Professional for Microsoft (com.snc.samp.microsoft)
- Software Asset Management Professional for SAP (com.sn_samp_sap)
- Software Asset Management Professional for Oracle (com.snc.samp.oracle)
- Software Asset Management - SaaS License Management Integrations (sn_sam_saas_int)

Software Asset Management test suite

Test	Description	Release version
SAM - Mapping Custom PPN/ DMAP to newly published PPN and Content	Validates the replacement of a custom publisher part number (PPN) with a new Software Asset Management Content Service PPN.	San Diego
SAM - Oracle PaaS BYOL	• Validates the addition of the new Serverless Hardware [cmdb_ci_serverless_hardware] table, which stores information about PaaS devices. • Validates the license compliance of Oracle Database servers in Amazon Web Services (AWS) PaaS environments. **Note:** Requires the Software Asset Management Professional for Oracle (com.snc.samp.oracle) plugin and the CMDB CI Class Models store application.	Rome
SAM - Software Product Lifecycles records on Software Model	Validates that the Software Lifecycle tab on the Software Model form is showing records. **Note:** Requires demo data and the Software Asset Management Professional for Microsoft (com.snc.samp.microsoft) plugin.	Quebec
SAM - BYOL	• Validates the purchase date on the Software Entitlement form • Validates the addition of newly added column legacy_license on the License Metric Results [samp_license_metric_result]	Quebec

Software Asset Management test suite (continued)

Test	Description	Release version
	and License Position Report [samp_license_position_report] tables • Validates the addition of newly added columns, cloud_license_type and cloud_license _type_source in the Software Installations [cmdb_sam_sw_install] table **Note:** Requires the Software Asset Management Professional for Micorosft (com.snc.samp.microsoft) plugin.	
SAM - Validate CIS Suites	Validates reconciliation of Core Infrastructure Server (CIS) suites along with downgrade rights. **Note:** Requires demo data and the Software Asset Management Professional for Microsoft (com.snc.samp.microsoft) plugin.	Paris
SAM - PerCoreForMSCluster	Verifies the reconciliation functionality for Microsoft per core with cluster. **Note:** Requires the Software Asset Management Professional for Microsoft (com.snc.samp.microsoft) plugin.	Paris
SAM - Validate upgrade/ downgrade during Reconciliation for Microsoft publisher	Validates upgrade and downgrade rights during reconciliation for Microsoft products.	Paris

Software Asset Management test suite (continued)

Test	Description	Release version
	Note: Requires demo data and the Software Asset Management Professional for Microsoft (com.snc.samp.microsoft) plugin.	
SAM - Generate demand to consolidate SaaS applications	Validates generation and submission of a demand on SaaS applications. **Note:** Requires the Software Asset Management - Spend Detection (com.sn_sam_spend) plugin and the PPM standard (com.snc.financial_planning_pmo) plugin.	Paris
SAM - Software Model and Software Entitlement checks for SAP Engines	Verifies that the Software Model and Software Entitlement forms change when the product is an SAP engine. **Note:** Requires the Software Asset Management Professional for SAP (com.sn_samp_sap) plugin.	Paris
SAM - Downgrade Rights on Software Model	Validates that the downgrade rights pushed from the content service are correctly populated on the Downgrade Rights related list on the software model form.	Orlando
SAM - Multi-core pack validation on Software Entitlement	Validates the functionality of new fields for a multi-core pack on software entitlements.	Orlando

Software Asset Management test suite (continued)

Test	Description	Release version
	Note: Requires the Software Asset Management Professional for Microsoft (com.snc.samp.microsoft) plugin.	
SAM - Downgrade Rights on Software Entitlement	Validates that the downgrade rights pushed from the content service are correctly populated on the Downgrade Rights related list on the software entitlement form.	Orlando
SAM - Software Spend Transaction	Validates the creation of a Software Spend Transaction. **Note:** Requires the Software Asset Management - Spend Detection (com.sn_sam_spend) plugin.	Orlando
SAM - Software Model and Software Entitlement	Tests that a user can create a software model and software entitlement and validates those records.	New York
SAM - Software Installation and Discovery Model	Tests that a user can create a software installation and discovery model and validates those records.	New York
SAM - Software Entitlement Creation Using Custom PPN	Creates a custom software product, a custom DMAP for the custom product, a custom Part Number for the custom DMAP, a software entitlement using the custom Part Number, and verifies that a software model is automatically created.	New York
SAM - Software Model Checks for SAP Named Users	Tests that the software model form changes when the publisher is SAP and the product is Named Users.	New York

Software Asset Management test suite (continued)

Test	Description	Release version
	Note: Requires the Software Asset Management Professional for SAP (com.sn_samp_sap) plugin.	
SAM - Software Model Checks for SaaS	Tests that the Software Model form changes when a SaaS product is selected. **Note:** Requires the Software Asset Management - SaaS License Management Integrations (sn_sam_saas_int) plugin.	New York
SAM - Validate Fields on SaaS Software Products	Tests that the **Subscription software** and **Ignore installs** fields are present on the Software Product form. **Note:** Requires the Software Asset Management - SaaS License Management Integrations (sn_sam_saas_int) plugin.	New York

To learn more about Software Asset Management, see Software Asset Management.

Security Incident Response

Security Incident Response quick start tests require activating Security Incident Response plugin (com.snc.security_incident) and loading the demo data.

Security Incident Response tests

Test	Description	Release version
SIR: Create Security Incident	Determine whether a user can successfully create a security incident from the security incident form.	Xanadu

Security Incident Response tests (continued)

Test	Description	Release version
SIR: Create Security Incident via Security Incident Catalog	Determine whether a user can successfully create a security incident from the catalog.	Xanadu
SIR: Security Incident life cycle	Validate the response tasks of the Policy Violation workflow.	Xanadu
SIR: Threat Lookup	Validates the Threat Lookup capability.	Xanadu
SIR: PIR Assessments OOTB configuration	Use this test to validate PIR assessments and base system configurations.	Xanadu
SIR: PIR Assessments conditional configuration	Verify that security incidents matching the mandatory conditional rule are not closed without completing the post incident assessment. Verify that the security incidents matching the optional conditional rule can be closed without completing the post incident assessment. Verify that assessments are not generated for the security incidents that do not match any rule.	Xanadu
SIR: PIR run time verification	Verify that PIR reports are configured and attached to the security incidents as per the new design.	Xanadu
SIR: PIR design time setup verification	Verify that the security incident is mapped with the report template based on the administrator configuration.	Xanadu
SIR: Link Security Incident to a existing Major Security Incident	Link a Security Incident to an existing Major Security Incident and validate data from Security Incident rolled up to Major Security Incident.	Xanadu
SIR: Promote Security Incident as Major Security Incident	Promote a Security Incident as Major Security Incident and validate data from Security Incident rolled up to Major Security Incident.	Xanadu
SIR: Propose Security Incident as Major Security Incident	Propose a Security Incident as Major Security Incident and validate data from Security	Xanadu

Security Incident Response tests (continued)

Test	Description	Release version
	Incident rolled up to Major Security Incident.	
Verify that only Allowed Members can access the security incident once Enforce Restriction is ON	Verify that only the allowed members can access the security incident once the Enforce Restriction is enabled.	Xanadu
Verify that only Allowed Groups can access the security incident once Enforce Restriction is ON	Verify that only the allowed groups can access the security incident once the Enforce Restriction is enabled.	Xanadu
Validate Read Access	Validate the view access.	Xanadu
Validate Write Access	Validate the edit access.	Xanadu

To learn more about Security Incident Response, see Security Incident Response.

Service Level Management

Service Level Management quick start tests require activating the Service Level Management - ATF Tests plugin (com.snc.service_level_management.atf).

Service Level Management quick start tests

Test	Description	Release version
SLM: Service Level Management Task SLA completed workflow	Tests that a user can create a task SLA and complete the workflow.	Orlando
SLM: Service Level Management Task SLA cancelled workflow	Tests that a user can create a task SLA and cancel the workflow.	Orlando
SLM: Service Level Management Task SLA timer REST API	Tests that the SLA Timer API response matches data expected by SLA Timer Seismic component.	Paris

To learn more about Service Level Management, see Service Level Management.

Service Mapping

Service Mapping quick start tests require activating the Service Mapping (com.snc.service-mapping) plugin and loading the demo data.

Service Mapping quick start tests

Test	Description	Release version
SM OOTB: Tests SM application visibility by roles	Validate the visibility of Service Mapping applications and modules	Madrid

Service Mapping quick start tests (continued)

Test	Description	Release version
	for different roles. For example, the test verifies that a user logged in with the service_mapping_user role cannot access the Administration module under Service Mapping.	
SM OOTB: Service map verification	Use this test template to create custom tests for verifying that the topology of the most significant services is unchanged. Configure values under Test Run Data Sets, to identify the name of the service to test and the nodes that you expect to find in this service. For the node name values, enter the node attribute exactly as it appears on the map, for example, "Apache server." If the attribute name for a node is truncated or shows that a node is a CI cluster, configure this node name value to reflect the way it appears on the map, for example, "11x ExchangeF" for a group of 11 Exchange FrontEnd servers. **Note:** You cannot use this test to verify the service content beyond the top, unexpanded level.	Orlando, updated Paris
SM OOTB: Check UI accessibility after an upgrade	Run this test to check that the Service Mapping UI is fully functional after an upgrade.	Orlando

To learn more about Service Mapping, see Service Mapping.

Service Portfolio Management Premium

Service Portfolio Management Premium quick start tests require activating the Service Portfolio Management Premium plugin (com.snc.spm).

Service Portfolio Management Premium - ATF Tests test suite

Test	Description	Release version
Portfolio Editor: Not Portfolio Owner, Read Only Taxonomy Access	Ensure a portfolio editor can only read associated non-owned portfolio taxonomies.	Orlando
Portfolio Editor: Cannot create new Portfolios	Ensure a portfolio editor cannot create new portfolios.	Orlando
Portfolio Editor: Valid Portfolio Owner Taxonomy Access	Ensure a portfolio owner has access to taxonomies within owned portfolios.	Orlando
Service Editor: Access	Ensure a service editor can only edit owned services and offerings.	Orlando
Portfolio Editor: Can Modify Owned Portfolios	Ensure a portfolio editor can modify and update owned portfolios.	Orlando
Create a Portfolio	Ensure a portfolio admin can create a new portfolio.	Orlando
Portfolio Editor: Cannot Modify Non-Owned Portfolios	Ensure a portfolio editor cannot modify non-owned portfolios.	Orlando
Service Workflow	Verify that a service cannot move forward to Catalog phase without a service portfolio, taxonomy node, and service offering attached to it. Verify a service cannot move backward from the Catalog phase to the Pipeline phase.	Orlando
Service Portfolio (Normal)	Create a service portfolio, taxonomy layer, and taxonomy nodes.	Orlando
Create a Portfolio + 3 Taxonomy Layers	Create a service portfolio with three taxonomy layers.	Orlando
Service Portfolio (Exception) - Taxonomy Layer Definition Set 01	Create a service portfolio, taxonomy layer, and taxonomy nodes.	Orlando
SPM: Create a Portfolio	Create a service portfolio.	Paris
SPM: Create a Portfolio + 3 Taxonomy Layers	Create a service portfolio, with three taxonomy layers.	Paris
SPM: Service Portfolio (Normal)	Create a service portfolio, taxonomy layer, and taxonomy nodes.	Paris

Service Portfolio Management Premium - ATF Tests test suite (continued)

Test	Description	Release version
SPM: Service Portfolio (Exception) - Taxonomy Layer Definition Set 01	Create a service portfolio, taxonomy layer, and taxonomy nodes.	Paris
SPM: Portfolio Editor: Can Modify Owned Portfolios	Ensure a portfolio editor can modify portfolios that they own.	Paris
SPM: Portfolio Editor: Cannot Modify Non-Owned Portfolios	Ensure a portfolio editor cannot modify portfolios that they do not own.	Paris
SPM: Portfolio Editor: Cannot create new Portfolios	Ensure a portfolio editor cannot create new Portfolios.	Paris
SPM: Portfolio Editor: Valid Portfolio Owner Taxonomy Access	Ensure portfolio owners have access to taxonomies that they own.	Paris
SPM: Portfolio Editor: Not Portfolio Owner, Read Only Taxonomy Access	Ensure a portfolio editor can only read the taxonomies of portfolios they do not own.	Paris
SPM: Service Editor: Access	Ensure a service editor can only edit services and offerings that they own or are a delegate of.	Paris
SPM: Service Viewer: Access	Ensure a service viewer has access to view services.	Paris
SPM: Service Workflow	Verify a service cannot move forward to Catalog phase without a service portfolio, taxonomy node, and service offering attached to it. Verify a service cannot move backward from Catalog to Pipeline phase.	Paris

To learn more about Service Portfolio Management Premium, see Service Portfolio Management Premium.

Skills Management

Skills Management quick start tests require activating the Skills Management plugin (com.snc.skills_management).

Skills Management test suite

Test	Description	Release version
Skills MGMT: User skill level inheritance when user is part of multiple groups	Verify that the user is assigned the highest skill level when the user belongs to multiple groups that have been assigned the same skills with different skill levels.	New York
Skills MGMT: Add skills to lowest level category	Verify that skills can be added to the lowest level category.	New York
Skills MGMT: Create a child category under a parent category	Verify that a lower-level category can be created when the flag for **Add skills** is unchecked in the parent category.	New York
Skills MGMT: Create a skill category	Verify that a skill category can be created on the skill category form.	New York
Skills MGMT: Create skill level type and skill levels	Define the skill level type and different skill levels for the type.	New York
Skill MGMT: Skill level inheritance from group to user	Verify that skill levels can be inherited from a group to the users of the group and that the **Inherited** and **Skill level inherited** fields are set to true.	New York
Skills MGMT: Create a skill from Manage IT Skills user interface.	Verify that you can create a skill from the Manage IT Skills user interface.	Orlando
Skills MGMT: Add skills and skill levels to users in the Manage IT Skills user interface.	Verify that you can add skills and associated skill levels to users in the Manage IT Skills user interface.	Orlando
Skills MGMT: Select a skill and add users to the skill in the Manage IT Skills user interface.	Verify that you can select a skill and add the skill and associated skill levels to one or more users in the Manage IT Skills user interface.	Orlando
Skills MGMT: Select a skill that does not have a skill level and add users to that skill in the Manage IT Skills user interface.	Verify that you can select a skill that does not have a skill level and add the skill to one or more users in the Manage IT skills user interface.	Orlando

To learn more about Skills Management, see Skills Management.

Test Management 2.0

Test Management 2.0 quick start tests require activating the Test Management 2.0 plugin (com.snc.test_management.2.0). and the Test Management 2.0 - ATF Tests plugin (com.snc.test_management.2.0.atf).

Test Management 2.0: Test version test suite

Test	Description	Release version
Create test version should create test	Validate test creation and version.	Madrid
Should be able to mark test version as ready when it contains verification steps	Validate test state when test has verification steps.	Madrid
Should not able to mark test version as ready when it does not contain verification step	Validate test state when test does not have verification steps.	Madrid
Marking a test version as ready should retire other test version in ready state	Validate test state when marking test ready.	Madrid

Test Management 2.0: Test results rollup test suite

Test	Description	Release version
When test run closed, should update execution suite progress	Validate execution state progress.	Madrid
Should not be able to assign a test not in ready state	Validate test assignment.	Madrid
Test progress should roll up for test plan and test cycle	Validate test progress for test plan and test cycle.	Madrid

To learn more about Test Management 2.0, see Test Management 2.0.

Universal Request

Universal Request quick start tests require activating the Universal Request (com.snc.universal_request), Universal Request: Integration for Incident management (com.snc.incident.universal_request), and Human Resources Scoped App: Core (com.sn_hr_core) plugins.

Universal Request quick start tests

Test	Description	Release version
UR: Transferring an HR case to an Incident	Validate if an HR case is transferred to an incident and an incident is created.	Quebec
UR: Restrict or Unrestrict universal request	Validates the following: • The fields are restricted and hidden when a routing agent restricts a UR. • Only agents with `sensitiveinfo_agent` role are allowed to unrestrict the request.	Xanadu

Universal Request quick start tests (continued)

Test	Description	Release version
	• And, after the request is unrestricted all fields are visible. • sys-audit log access is available for agents.	
UR: Universal Request creation with RITM	Verifies a catalog request for any item from self-service can be created when Universal Request Certified is enabled.	San Diego

Vendor Management Workspace

Vendor Management Workspace quick start tests require activating the Vendor Manager Workspace plugin (com.snc.vlm.vmw) and loading demo data.

Vendor Management Workspace - Tests test suite

Test	Description	Release version
VLM: Create a Vendor	Track problems with the Vendor Manager Workspace plugin after upgrade. Create a new vendor.	New York

To learn more about Vendor Management Workspace, see Vendor Management Workspace.

Vendor Manager Workspace and Third-party Risk Management

GRC: Vendor Manager Workspace and Third-party Risk Management quick start tests require activating the Vendor Manager Workspace and Third-party Risk Management plugin (com.sn_vdr_risk_asmt) and loading demo data.

GRC: Vendor Risk Management and Third-party Risk Management Quick Start Tests

Test	Description
GRC: Create Engagement Risk Assessment	Creates and submits an engagement risk assessment to an engagement.
GRC: Vendor Risk Scoring - Cancel assessment for engagement	Verifies a request to recalculate a risk rating is automatically created on an Engagement and Third party after one assessment for the Engagement is cancelled.
GRC: Vendor Issue ATF	Creates and submits a third-party risk issue form and third-party risk task form.
GRC: Create Vendor Risk Assessment	Creates and submits a vendor risk assessment to a vendor.

GRC: Vendor Risk Management and Third-party Risk Management Quick Start Tests (continued)

Test	Description
GRC: Vendor Portal - Answer and Return Assessment	Vendor contact answers and submits assessment in the Service Vendor Portal.
GRC: Vendor Tiering Assessment	Selects and submits an assessment to respective assessors after changing the duration.

Verify that TPRM still works after you make configuration changes such as applying an upgrade or developing an application. Copy and customize the quick-start tests to pass when using your instance-specific data.

Vulnerability Response

Vulnerability Response quick start tests require activating the Vulnerability Response application (sn_vul) and loading the demo data.

Vulnerability Response tests

Test	Description	Release version
VR: Create Remediation Target Rule	Create a Remediation Target Rule.	Madrid
VR: Create Vulnerability Group Rule	Create a Vulnerability Group Rule.	Madrid
VR: Create Vulnerable Item via Form	Determine whether a user can successfully create a vulnerable item from the Vulnerable Item form.	Madrid
VR: Vulnerability Group Life Cycle	Determine whether a user can successfully resolve a vulnerability group.	Madrid
VR: Vulnerable Item life cycle	Determine whether a user can successfully move a vulnerable item through its life cycle, and also determine whether a closed vulnerable item can be reopened.	Madrid
VR: Rollup Calculator	Determine whether the rollup risk calculator can provide an overall risk score for an entire group of vulnerable items using the scores for all the vulnerable items in a vulnerability group.	New York
VR: Vulnerability Response Assignment Rules	Determine whether a sample set of assignment rules can successfully auto-assign vulnerable items to	New York

Vulnerability Response tests (continued)

Test	Description	Release version
	an assignment group for remediation.	
VR: Vulnerability Calculators	Test the vulnerability calculators.	New York
VR: CI Lookup - Qualys	Create a new lookup rule with method "field_matching" called "Lookup By Network Adapter" for Qualys. Determine whether a configuration item is successfully matched in the Discovered Item table by network adapter and IP address with the new lookup rule.	Orlando
VR: Create Normal and Emergency Change Request	Determine whether the user can successfully create normal and emergency change requests from a vulnerability group.	Orlando
VR: Split Vulnerability Group	Determine whether the user can successfully split a vulnerability group.	Orlando
VR: Update VG state when a CHG is cancelled.	Determine whether the State field on a vulnerability group successfully transitions when a change request that is associated with the vulnerability group is cancelled.	Orlando
VR: Update VG state when a CHG transitions to Review.	Determine whether the State field on a vulnerability group successfully transitions when a change request that is associated with it moves to the Review state.	Orlando
VR: CI Lookup - Rapid7	Test CI lookup using the existing Rapid7 Vulnerability Integration lookup rule, **IP Address**.	Orlando
VR: CI lookup - Qualys	Test CI lookup by creating a new lookup rule for the Qualys Vulnerability Integration	Orlando
VR: Exception Approval Workflow for VI	Create an exception request and verify that the approval process is working.	Orlando

Vulnerability Response tests (continued)

Test	Description	Release version
VR: False Positive Approval Workflow for VI	Create a false positive exception request and verify that the approval process is working.	Orlando
VR: Application Vulnerability Response (AVR)	Determine whether your rules and calculators are working correctly. Verify that updates are working.	Orlando
Remediation target rules: VI import test	Tests VR remediation target rules during import.	Paris
VR: Classification Rule	A sample classification rule that automatically classifies a vulnerability.	Tokyo

To learn more about Vulnerability Response, see Vulnerability Response.

Walk-up Experience

Walk-up Experience quick start tests require activating the Walk-up Experience plugin (com.snc.walkup) and loading demo data.

Walk-up Experience - Tests test suite

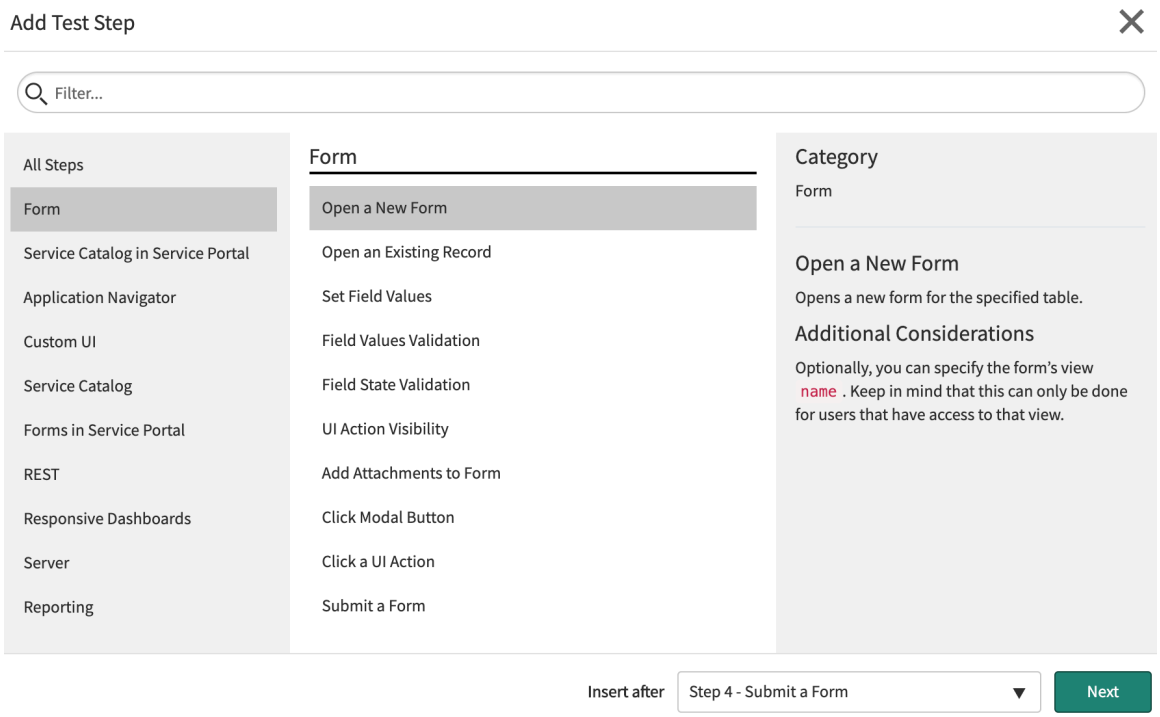
Test	Description	Release version
Create a location	Ensure users with the Walk-up administrator [sn_walkup.walkup_admin] role can access all existing walk-up queues, create new queue locations, and configure queues appropriately.	Orlando
Onsite checkin (ITIL User)	Verify that users with the ITIL role can check into an onsite queue location.	Orlando
Onsite checkin (ESS User)	Verify that users with the ESS role can check into an onsite queue location.	Orlando
Onsite checkin (Guest)	Verify that guest users can check into an onsite queue location.	Orlando

To learn more about Walk-up Experience, see Walk-up Experience.

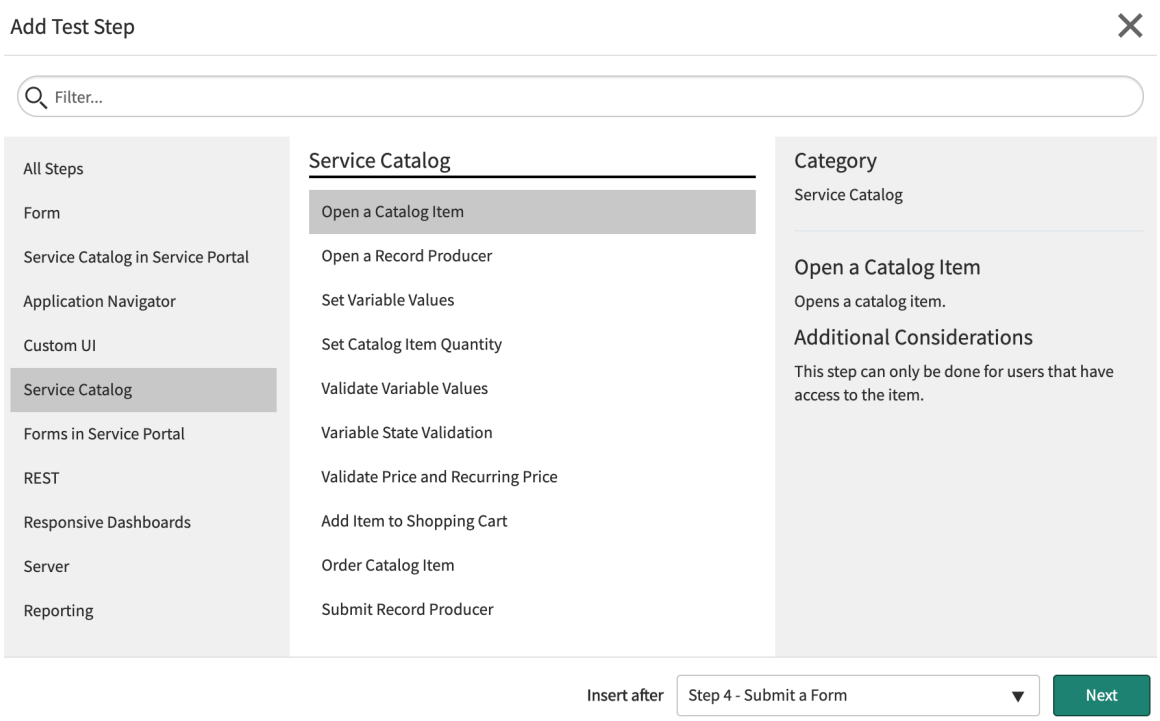
Test step categories

Find test steps for a particular user interface or ServiceNow AI Platform feature.

The Add Test Step panel lists test categories in the left pane. You can directly search for a specific test step by its name in the search field at the top of the panel. You can also select **All Steps** in the left pane to list all available test steps in the middle pane. A description of the selected step appears in the right pane.



You can select a specific test category to list only the test steps available for that test category in the middle panel. A description of the selected step appears in the right panel.



Service Catalog in Service Portal category

Validate catalog item transactions and requester flows from Service Portal.

Activation of the Automated Test Framework for Service Catalog in Service Portal

These ATF test steps require activation of the Automated Test Framework Service Catalog Service Portal (com.glide.automated_testing_impl.service_catalog_portal) plugin, which is active by default on new instances. Administrators may need to activate the plugin on instances upgraded from earlier versions.

Support for parametrized tests

Service Catalog in Service Portal step configurations support parametrized tests. For more information on parametrized tests, refer to Parameterized tests.

Open a Record Producer (SP)

Open a record producer in the Service Portal.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Portal	Portal for which you want to test this step.
Page	Service Catalog page associated with the test step.
Record Producer	Record producer that you want to open. **Note:** You should have access to the record producer.
Query Parameters	URL query parameters for the page, such as *sys_id*.

Open a Catalog Item (SP)

Open a catalog item in the Service Portal.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Portal	Portal for which you want to test this step.
Page	Service Catalog page associated with the test step.
Catalog Item	Catalog item that you want to open. **Note:** You should have access to this catalog item.
Query Parameters	URL query parameters for the page, such as *sys_id*.

Open an Order Guide (SP)

Open an order guide in the Service Portal.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.

Inputs (continued)

Field	Description
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Portal	Portal for which you want to test this step.
Page	Service Catalog page associated with the test step.
Order Guide	Order guide that you want to open. **Note:** You should have access to this order guide.
Query Parameters	URL query parameters for the page, such as *sys_id*.

Add row to multi-row variable set (SP)

Add a row to a multi-row variable set included in the current catalog item in Service Portal. You can use this step configuration only when the current catalog item contains a multi-row variable set.

Use this step after the Open a Catalog Item(SP) step, Open a Record Producer(SP) step, or Open an Order Guide (SP) step. If a contextual value has been used for the Open a Catalog Item(SP) step, Open a Record Producer(SP) step, or Open an Order Guide (SP) step, set the catalog item in this step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Notes	Notes about the test step.

Inputs (continued)

Field	Description
Catalog Item	Catalog item whose multi-row variable set requires an addition of a row. If an item is not already selected, you can either search for the item, or insert a reference to the contextual value of the item from a list of available parameters.
Multi-Row Variable Set	Multi-row variable set for which a row should be added.

Save current row of multi-row variable set (SP)

Save the current row of a multi-row variable set included in the current catalog item in Service Portal. You can use this step configuration only when the current catalog item contains a multi-row variable set. Use this step after the Add row to multi-row variable set (SP) step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Notes	Notes about the test step.
Assert Type	Criteria for the test to pass. **Successfully saved row to a multi-row variable set** Test passes only if the current row of the multi-row variable is saved. **Cannot save a row to multi-row variable set** Test passes only if the current row of the multi-row variable is not saved.

Set Variable Values (SP)

Set variable values for a catalog item or record producer in the Service Portal. For a catalog item, use this step after using the Open a Catalog Item (SP) step, and before using the Order Catalog Item (SP) step.

For a record producer, use this step after using the Open a Record Producer (SP) step, and before using the Submit Record Producer (SP) step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Item	Catalog item or record producer for which you want to set variable values.
Multi-Row Variable Set	Multi-row variable set for which variable values should be set. **Note:** Use the Add row to multi-row variable set (SP) step configuration prior to the current step configuration.
Variable Values	List of variables and the values that you want to set for them. **Note:** You can set the value for multiple variables.

Note:
Custom variables and custom variable with labels are not supported for Set Variable Values step configuration.

Validate Variable Values (SP)

Validates variable values of a catalog item or record producer in Service Portal. For a catalog item, use this step after using the Open a Catalog Item (SP) step, and before using the Order Catalog Item (SP) step.

For a record producer, use this step after using the Open a Record Producer (SP) step, and before using the Submit Record Producer (SP) step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Item	Catalog item or record producer whose variables should be validated.
Multi-Row Variable Set	Multi-row variable set for which variable values should be validated. **Note:** Use the Add row to multi-row variable set (SP) step configuration prior to the current step configuration.
Catalog Conditions	Conditions for variable validation. If the conditions are met, the test passes. **Note:** The label of a variable associated with a variable set reflects the variable set name. The format is *`variable_set_name » variable_name`*.

Note:
Custom variables and custom variable with labels are not supported for Validate Variable Values step configuration.

Variable State Validation (SP)

Validates the state of variables in Service Portal. Possible variable states are mandatory, not mandatory, read only, not read only, visible, and not visible.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Item	Catalog item or record producer whose variables should be validated.
Multi-Row Variable Set	Multi-row variable set for which variable states should be validated. **Note:** Use the Add row to multi-row variable set (SP) step configuration prior to the current step configuration.
Visible	List of the catalog item variables that must be visible for the step to pass.
Not visible	List of the catalog item variables that must be hidden for the step to pass.
Read only	List of the catalog item variables that must be read-only for the step to pass.
Not read only	List of the catalog item variables that must not be read-only for the step to pass.
Mandatory	List of the catalog item variables that must be mandatory for the step to pass.
Not mandatory	List of the catalog item variables that must not be mandatory for the step to pass.

Note:
Custom variables and custom variable with labels are not supported for Variable State Validation step configuration.

Validate Price and Recurring Price (SP)

Validate the price and recurring price of a catalog item in Service Portal. Use this step after using the Open a Catalog Item (SP) step, and before using the Order Catalog Item (SP) step. This step is not applicable for a record producer.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Price	Price of the catalog item for the step to pass.
Recurring price	Recurring price of the catalog item for the step to pass.
Recurring price frequency	Recurring price frequency of the catalog item for the step to pass.

Navigate within Order Guide (SP)

Navigate within an order guide.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of

Inputs (continued)

Field	Description
	the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Guide Step	Order guide step to which you want to navigate: • **Describe Needs** • **Choose Options** • **Summary**
Assert Type	Criteria for the test to pass. **Navigation Successful** Test passes only if the navigation is successful. **Navigation Failed** Test passes only if the navigation fails.

Set Catalog Item Quantity (SP)

Set the quantity for a catalog item in Service Portal. This step is not applicable for a record producer. Use this step after using the Open a Catalog Item (SP) step, and before using the Order Catalog Item (SP) step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Quantity	Quantity of the catalog item that you want to order.

Validate Order Guide Items (SP)

Validate items included in the order guide.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Catalog Items	Catalog items that you want to validate. **Note:** You should have access to these catalog items.

Review Order Guide Summary (SP)

Review the order guide summary in the Service Portal.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.

Inputs (continued)

Field	Description
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Catalog Items	Catalog items that you want to review. **Note:** You should have access to these catalog items.
Price	Price of the catalog item for the step to pass.

Review Item in Order Guide (SP)

Review individual items in the order guide and choose whether or not to include the item.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of

Inputs (continued)

Field	Description
	the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Catalog Item	Catalog item that you want to review. **Note:** You should have access to this catalog item.
Included	Selected if the catalog item should be included in the order guide, otherwise unselected.

Add Item to Shopping Cart (SP)

Add the current catalog item to the shopping cart in Service Portal. Use this step after using the Open a Catalog Item (SP) step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Assert Type	Criteria for the test to pass. **Successfully added item to Shopping Cart** Test passes only if the catalog item is successfully added to the shopping cart. **Cannot add item to Shopping Cart** Test passes only if the catalog item cannot be added to the shopping cart.

Outputs

Field	Description
cart_item_id	The sys_id of the added catalog item.

Add Order Guide to Shopping Cart (SP)

Add an order guide to the shopping cart.

Input

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Assert Type	Criteria for the test to pass. **Cannot add order guide to shopping cart** Test passes only if the order guide cannot be added to the shopping cart. **Successfully added order guide to shopping cart** Test passes only if the order guide is successfully added to the shopping cart.

Order a Catalog Item (SP)

Click **Order Now** for the current catalog item in the Service Portal. Use this step after using the Open a Catalog Item (SP) step.

After this step, you cannot use any other steps on the catalog item. If the two-step checkout is false, a request is generated for the catalog item. If the two-step checkout is true, you are redirected to the cart preview page.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Assert type	Criteria for the test to pass. **Successfully ordered Catalog Item** Test passes only if the catalog item is successfully ordered. **Cannot order Catalog Item** Test passes only if the catalog item cannot be ordered.

Outputs

Field	Description
table	The table to which the submitted request belongs.
record_id	The sys_id of the submitted request.

Submit an Order Guide (SP)

Click **Order Now** to order an order guide. Do not add more than one record producer to the order guide.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Assert Type	Criteria for the test to pass. **Cannot order the Order Guide** Test passes only if the order guide cannot be ordered. **Successfully ordered the Order Guide** Test passes only if the order guide is successfully ordered.

Outputs

Field	Description
table	The table containing the submitted order guide.
record_id	The sys_id of the submitted order guide.

Submit Record Producer (SP)

Submit the current record producer in the Service Portal. Use this step after using the Open a Record producer (SP) step. After this step, you cannot use any other steps on the record producer.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Assert Type	Criteria for the test to pass. **Successfully submitted Record Producer** Test passes only if the record producer is submitted successfully. **Cannot submit Record Producer** Test passes only if the record producer cannot be submitted.

Outputs

Field	Description
table	The table containing target record of the record producer.

Application Navigator category

Verify the functionality of menus and modules in the application navigator.

Application Menu Visibility

Verifies the visibility, or lack thereof, of selected application menus in the application navigator (left navigation bar). For example, you create a test that first impersonates a user, then verifies that specified application menus (such as Self-Service and Reports) are visible, or are not visible, to that user.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.

Inputs (continued)

Field	Description
Active	Option to activate this test step for use.
Application	The application scope in which the system runs this test or test suite.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Navigator	Navigator version to assert against • Core UI (UI16): If you are creating new steps, you will have Core UI by default. • Next Experience: If you have Next Experience enabled, Next Experience is the default navigator. If it is disabled, Core UI is the default navigator.
Visible assert type	Specifies how application menus selected in the **Visible application menus** field should be tested for visibility in the application navigator. • **At least these application menus are visible**: At minimum, all the selected application menus are visible in the application navigator. • **Only these application menus are visible**: Only the selected application menus are visible in the application navigator.
Visible application menus	Application menus whose visibility in the application navigator is being verified.
Not visible assert type	Specifies how application menus selected in the **Not visible application menus** field should be tested for lack of visibility in the application navigator. • **At least these application menus are not visible**: At minimum, all the selected application menus are not visible in the application navigator. • **Only these application menus are not visible**: Only the selected application menus are not visible in the application navigator.
Not visible application menus	Application menus whose lack of visibility in the application navigator is being verified.

Related topics

Create an application menu

Create an application menu

Application menus define the main content that users can access in the application navigator. You can configure which applications appear in the application navigator.

Before you begin

Role required: admin

About this task

When creating an application menu, consider grouping like modules into a consolidated application menu. When editing an existing menu, add more useful modules to the application menu, and remove unneeded ones.

Procedure

1. Navigate to **All** > **System Definition** > **Application Menus**.
2. Click **New**.
3. Complete the form.

Field	Description
Title	Defines the display name of the application menu.
Roles	Restricts access to the specified roles. All users can view the application menu when it is active.
Category	Specifies the menu category that defines the navigation menu style (default value is **Custom Applications**).
Hint	Defines the text that appears in a tooltip when a user points to this application menu.
Active	Select the check box to activate the application menu. Only active application menus appear in the application navigator.
Description	Provide a more detailed explanation of what this application does.
Other fields	
Order	Defines the relative position of the application menu in the application navigator. If you do not specify an order, the default order of the menu category is used.
Default device type	This field is not used. You can define application menus for mobile devices in a separate table.

Note:
You may need to configure the form to see all fields.

4. Click **Submit**.
5. Create modules to appear in the application menu.
   Only application menus that contain modules appear in the application navigator.

Create a module

Modules are the children, or the second tier navigation options to the applications in the application navigator. Modules often link to other pages or records in the platform. You can configure which modules appear in the application navigator using Application Menus module.

Watch this five-minute video to learn about adding application menus and modules to the application navigator. Shows how to add application menus and modules to the application navigator.

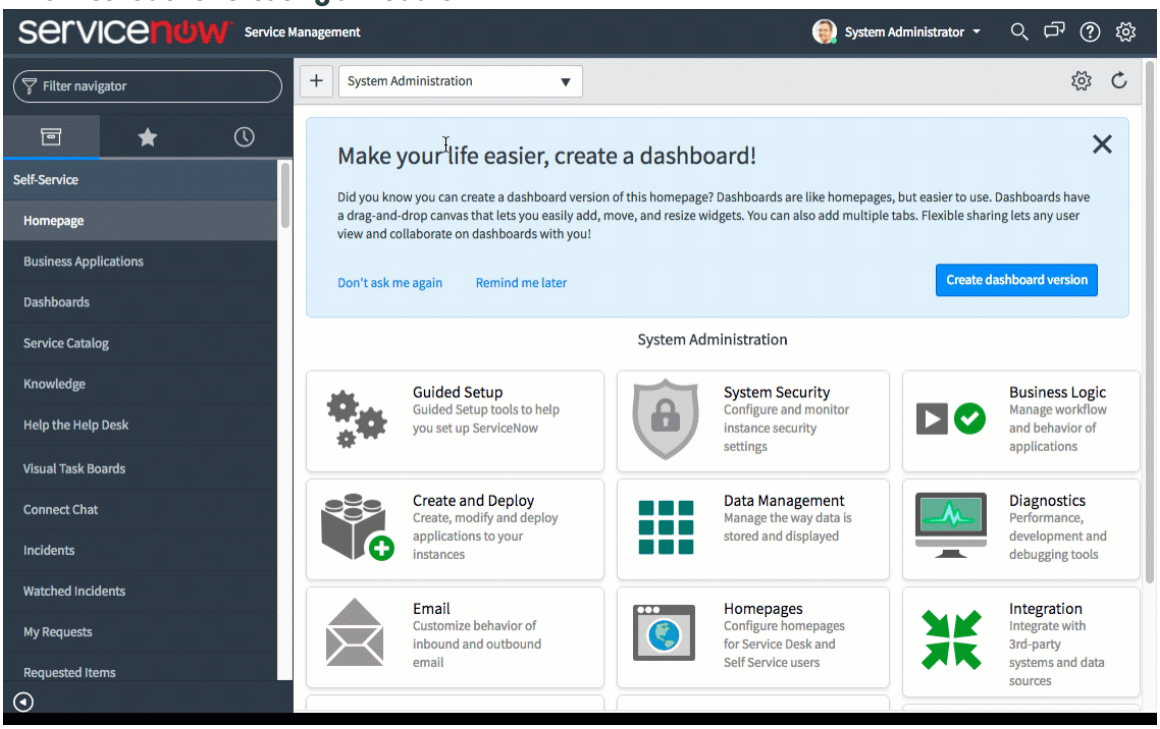
Before you begin

Role required: admin

Procedure

1. Open the application menu record using one of the following methods.
   - Navigate to **System Definition** > **Application Menus** and select the application menu from the list.
   - Point to the application menu and click the edit application (pencil) icon.

Two methods for creating a module



2. Scroll down to the **Modules** related list and click **New**.
3. Define the module by completing the fields on the Module form.

Module form

Field	Description
Title	Defines the module name. Choose a title that clearly identifies the module.
Application menu	Specifies the name of the application menu under which the module appears.
Hint	Defines the tooltip that appears when a user points to the module name. **Note:** Module Hints are deprecated in Core UI
Order	The order in which the module displays relative to other modules.

4. On the **Visibility** tab, complete the fields.

Visibility tab

Field	Description
Roles	Restricts module access to the specified roles. If this field is left blank, the module is visible to all users who have access to the application menu.
Active	Defines whether the module appears in the application navigator.
Override application menu roles	Allows users to access this module even if they do not have permission to view the containing application menu. Users must still meet the role requirements for this module.

5. On the **List Type** tab, complete the fields.

   The fields displayed depend on the **Link type** you choose. All module URIs must be encoded. If you supply arguments to the module URI, either you or ServiceNow. For more information about encoding module URIs, see Encoding module URIs.

List Type tab

Field	Description
Link type	Specifies what type of link this module opens. You must specify additional information based on the link type. See Module link types.
Table	Specifies the table used by the module. **Note:** The list shows only tables and database views that are in the same scope as the module.
Filter	Conditions for the items presented in the module, for example, **Active is true**.
Argument	String appended to the URI to create the module URI. Can be a sysparm_query. These values must be encoded either by you or ServiceNow. For more information about encoding module URIs, see Encoding module URIs.
Order	Specifies the order in which the modules appear under the application.

Module link types

The **Link type** field on the Module form specifies what type of link the module opens.

Module link types

Link Type	Description
Assessment	Links to the assessment-based survey you select in the **Assessment** reference field. See Create a survey module.
Content Page	Displays the content page you select in the **Content page** reference field. See Create a content page.

Module link types (continued)

Link Type	Description
Documentation Link	Links to a documentation page and opens in a new tab or window. This link type is used with embedded metadata in documentation topics. To open an internal document from a module, use the **URL (from Arguments)** module link type.
Homepage	Displays the homepage you select in the **Homepage** reference field.
HTML (from Arguments)	Places HTML in the application navigator. This link type is used for more complicated links, where a flat URL is not customizable enough. **Note:** • The **HTML (from Arguments)** link type is supported in the legacy UI15 and UI11 interfaces only. In Core UI, use the **URL (from Arguments)** link type instead. • Enter a value for the **Arguments** field.
List Filter	Displays an unpopulated list view for the table you select in the **Table** field. Allows users to specify a filter without loading the list first. Use the **Filter** field to define the default filter for the list. Use the **View name** field to specify a View management.
List of Records	Displays the list view for the table you select in the **Table** field. Use the **Filter** field to define the default filter for the list. Use the **View name** field to specify a view.
Map Page	Displays the map page you select in the **Map page** reference field.
New Record	Displays a form for creating a record in the table you select in the **Table** field. • Use the **View name** field to specify a view. • Use the **Arguments** field to apply a template. See Create a module for a template.
Run a Report	Runs the saved report you select in the **Report** field.
Script (from Arguments)	Runs a script, as defined in the **Arguments** field. **Note:** Enter a value for the **Arguments** field.
Search Screen	Link that displays a blank form for searching records in the table. Use the **View name** field to specify a view. **Note:** Use the parameter *&sysparm_result_view=view_name* to define the view the results are rendered in. All searches use a **[starts with]** query to search for matching text. Other query types are not supported in search screens.
Separator	Creates a division between modules. Enter a name in the **Title** field to add a section name that users can collapse or expand.

Module link types (continued)

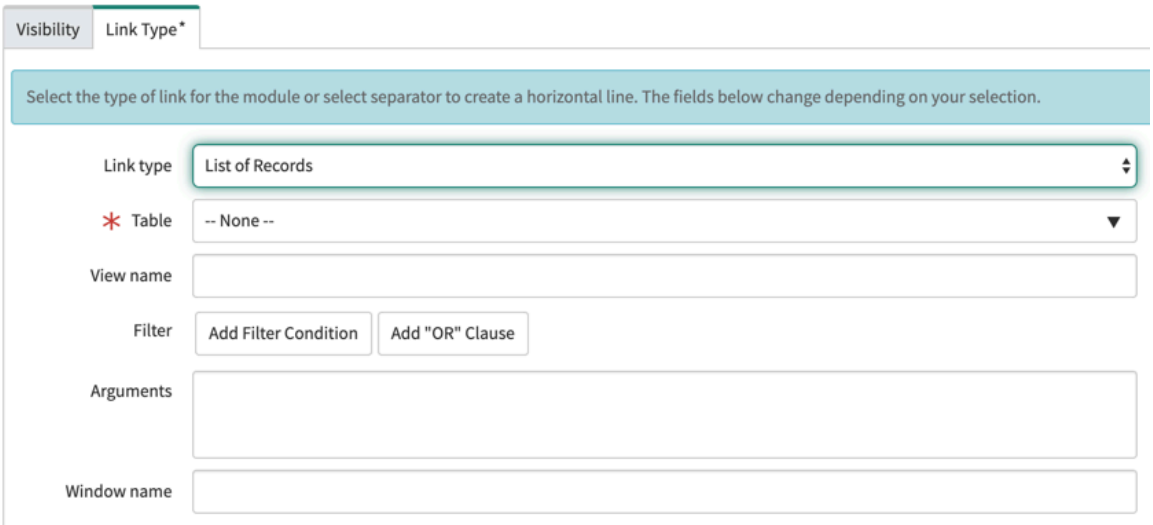
Link Type	Description
Single Record	Displays a form for a single record on the table. Use the **View name** field to specify a view.
Survey	Links to the legacy survey you select in the **Survey** reference field. Use the **Survey overwrite** check box to determine whether the survey can be taken multiple times. **Note:** The **Survey** link type is for use with legacy surveys only, which assessment-based surveys replace. Select the **Assessment** link type to link the module to an assessment-based survey.
Timeline Page	Displays the timeline page you select in the **Timeline Page** reference field. See Timeline pages.
URL (from Arguments)	Opens any URL, as defined in the **Arguments** field. [Optional] Use the **Window name** field to define a link that opens in a new window. **Note:** • For internal links, always use a relative link such as `./catalog_home.do?sysparm_view=catalog_default` or `catalog_home.do?sysparm_view=catalog_default`. Do not use an absolute link to a ServiceNow instance. It creates problems when you move an update set from a development instance to a production instance because the URL still references the development instance. • Enter a value for the **Arguments** field.

Encoding module URIs

Clicking a module name in the navigation pane executes a URI that opens the module's page in the content pane. All the characters in module URIs must be URL-encoded or the link breaks.

Note:
If you're upgrading to the New York release or later from a pre-New-York release, your module UIs may break if they do not follow the conventions presented in this topic.

When you create modules, you have the option of adding arguments and filter conditions that sort and/or reduce the number of results displayed in the content pane. When you click a module name in **System Definitions** > **Application Definitions**, you can see those conditions and arguments on the **Link Type** tab.



The argument definition in **Arguments** and filter conditions defined in **Filter** become part of the module's URI and must be URL-encoded. ServiceNow automatically URL-encodes filter conditions and appends them to the module URI using sysparm_query. For example, adding the filter condition, **Active is true** appends `sysparm_query=active%3Dtrue` to the module's URI; %3D is the URL-encoding for the equals sign (=).

The following table shows when you must URL-encode the argument in the **Arguments** field and when ServiceNow URL-encodes the argument.

Rules for encoding arguments

Has a filter condition?	Argument definition starts with	Who encodes the argument?	How argument is handled
No	^	ServiceNow	Removes the caret (^) from the argument, encodes it and uses sysparm_query to append it to the module URI.
No	&	You	Removes the ampersand (&) from the argument and appends it to the module URI.
No	Anything else	ServiceNow	Encodes the argument and uses sysparm_query to append it to the module URI.
Yes	^	ServiceNow	URL-encodes the filter definition and the argument and uses sysparm_query to append the combination to the module URI.
Yes	Anything else	You	URL-encodes the filter definition and uses sysparm_query to append it and the (unaltered) arguments to the module URI.

You can turn on (the default) and off the URL-encoding requirement for module UIs using the *`glide.ui.encode_module_uri`* property.

Examples

The following examples demonstrate when you have to URL-encode the argument definition in **Arguments**:

- There are no filter conditions and the argument definition in **Arguments** starts with an ampersand, for example,`&sysparm_fixed_query=assigned_to=javascript:gs.user_id()`.

  This argument breaks the module URI because the equals sign and the colon are not URL-encoded, and the ampersand prevents ServiceNow from URL-encoding the argument. URL-encode the argument: `&sysparm_fixed_query=assigned_to%3Djavascript%3Ags.user_id()`.

- There are filter conditions and the argument definition in **Arguments** does not start with a caret (^), for example, `sysparm_name=Barnes & Noble's`.

  This argument breaks the module URI because the ampersand and spaces are not URL-encoded. URL-encode the argument: `sysparm_name=Barnes%20%26%20Nobel's`

Module Visibility

Verify the visibility, or lack thereof, of selected modules in the application navigator (left navigation bar). For example, create a test that first impersonates a user, then verifies that specified modules (such as Homepage and My Requests) are visible, or are not visible, to that user.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	The application scope in which the system runs this test or test suite.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Navigator	Navigator version to assert against • Core UI (UI16): If you are creating new steps, you will have Core UI by default. • Next Experience: If you have Next Experience enabled, Next Experience is the default navigator. If it is disabled, Core UI is the default navigator.
Visible assert type	Specifies how modules selected in the **Visible modules** field should be tested for visibility in the application navigator. • **At least these modules are visible**: At minimum, the modules selected in the **Visible modules** field are visible in the application navigator. • **Only these modules are visible**: Only the specific modules selected in the **Visible modules** field are visible in the application navigator.

Inputs (continued)

Field	Description
Visible modules	Modules whose visibility in the application navigator is being verified.
Not visible assert type	Specifies how modules selected in the **Not visible modules** field should be tested for lack of visibility in the application navigator. • **At least these modules are not visible**: At minimum, the modules selected in the **Not visible modules** field are not visible in the application navigator. • **Only these modules are not visible**: Only the specific modules selected in the **Not visible modules** field are not visible in the application navigator.
Not visible modules	Modules whose lack of visibility in the application navigator is being verified.

Related topics

Create a module

Navigate to Module

Open a module from the application navigator, as if a user had clicked it. The module must be visible to the currently executing user to navigate to it.

Note:
Not all pages are currently testable. Wherever the module takes you is your responsibility.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Module	Module that should be opened. To navigate to the selected module, the module must be visible to the currently executing user in the application navigator. The following modules are not supported and cannot be tested:

Inputs (continued)

Field	Description
	• Modules that are separators • Modules that do not link to a specific page, but instead execute client-side JavaScript (such as Studio and the Script Debugger) • Modules that link to external websites, such as the ServiceNow documentation site (servicenow.com/docs) • Modules that reload or redirect the entire page

Related topics

Create a module

Custom UI category

Validate the behavior of page components on custom user interfaces.

Set Component Values (Custom UI)

Set component values on a custom UI page.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Component values	Page components to be tested and the value each page component should have for the step to succeed.

Inputs (continued)

Field	Description
	Note: The **Value** field defaults to the last retrieved value of the selected component.

Assert Text on Page (Custom UI)

Assert that the specified text is or is not on a custom UI page.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Text	Text to be compared with the text on the page.

Note:

- The text to be searched is case-sensitive.
- Starting with the Rome release, this test step also works for Workspace for any new tests.
- The Assert Text on Page test step also accepts text containing white spaces.

Component Value Validation (Custom UI)

Validate a component value on a custom UI page.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Component values	Page components to be tested and the value each page component should have for the step to succeed. **Note:** The **Value** field defaults to the last retrieved value of the selected component.

Click Component (Custom UI)

Click a component on a custom UI page.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You

Inputs (continued)

Field	Description
	can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Component	Component to be tested in this step.

Component State Validation (Custom UI)

Validate the state of a specified component on a custom UI page.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Component	Component to be tested in this step.

Inputs (continued)

Field	Description
Assert type	Type of assert: • **Disabled** • **Enabled**

Open Service Portal Page

Open a portal page. Test designers must first open a page in a portal before testing UI components on the page.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Portal	Select a portal.
Page	Select the page to open.
Query parameters	Add any required query parameters for the page. For example, to open a record in the form page, enter the table and sys_id parameters. For more information about Service Portal query parameters, see Page navigation by URL.

Form category

Validate the functionality of fields and UI actions on a form.

Note:
Test steps that include the **Form UI** field give you the option to select an available UI. For any available workspace, navigation between tabs is not supported. Use the **Open a New Form** or **Open an Existing Record** step to reopen a form.

It is to be noted that Agent Workspace has been removed from the Platform. If you have any Agent Workspace tests or test suites, they won't work anymore.

Open a New Form

Open a form to a new record in the specified table and **Form UI**.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Table	Name of the table for the new form.
View	Name of the view in which you want this form to open. The testing user must have access to that view. If the name is not a valid form view, the form opens in its default view.

Inputs (continued)

Field	Description
	Note: This field appears when **Standard UI** is chosen from **Form UI**.

Open an Existing Record

Open a form to an existing record in the specified table and **Form UI**.

Note: Using an existing record may cause unexpected behavior for this test. See Automated Test Framework design considerations for more information.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Table	Name of the table for the new form.

Inputs (continued)

Field	Description
Record	Record ID of the record that you want to open and name of the document that you want to open.
View	Name of the view in which you want this form to open. The testing user must have access to that view. If the name is not a valid form view, the form opens in its default view. **Note:** This field appears when **Standard UI** is chosen from **Form UI**.

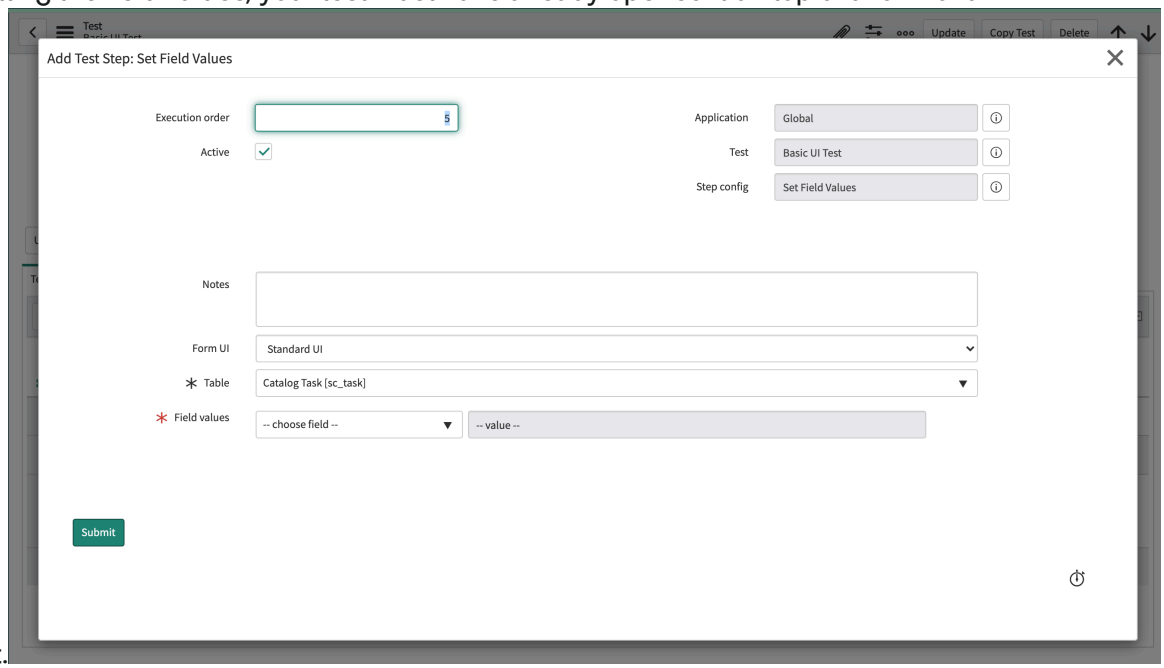
Set Field Values

Set the fields on the current form to the specified values.

To run this step, your test must have already opened a form using either the **Open a New Form** or **Open an Existing Record** step. It is recommended to not run this step directly after a **Submit a Form** or **Click a UI Action** step. This is because they can redirect your test to a different page based on the navigation stack configuration on your instance or the script defined in the clicked UI action. Unless you are certain that the UI action will take you to a specific page, you should explicitly use an **Open a New Form** step after **Submit a Form** or **Click a UI Action** to ensure that the test is on the form as expected. Ensure that the test keeps passing consistently when added to a suite.

The **Field Values Validation**, **Set Field Values**, **Field State Validation**, and **UI Action Visibility** steps can appear in any order.

Note:
This step waits for the form to load before setting field values. This step doesn't support reference qualifiers, neither at test design time nor at test runtime. A modal form appears either on top of another form or a list. To submit a modal form after setting the field values, your test must have already opened it on top of a form or a list.



Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace

Inputs (continued)

Field	Description
	selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Table	Name of the table for the new form.
Set field value	Assigns a value to a field on an open form.

Field Values Validation

Validate field values on the current form.

To run this step, your test must have already opened a form using either the **Open a New Form** or **Open an Existing Record** step. It is recommended to not run this step directly after a **Submit a Form** or **Click a UI Action** step. This is because they can redirect your test to a different page based on the navigation stack configuration on your instance or the script defined in the clicked UI action. Unless you are certain that the UI action will take you to a specific page, you should explicitly use an **Open a New Form** step after **Submit a Form** or **Click a UI Action** to ensure that the test is on the form as expected. Ensure that the test keeps passing consistently when added to a suite.

The **Field Values Validation**, **Set Field Values**, **Field State Validation**, and **UI Action Visibility** steps can appear in any order.

For the **Field Values Validation** step, specify the values that you want to test using the standard conditions builder. You can test several conditions against the same field. This step passes if the overall condition is satisfied and fails if the condition is not satisfied. The **Conditions** field is case-sensitive and requires to have the exact value as on the form. To test the values of individual fields independently of each other, include a separate **Field Values Validation** step for each value that you test.

Note:
The **Field Values Validation** step works only with fields that belong to the record for the open form. For example, with the incident table, this step is not able to validate the **Additional comments**, **Approval history**, **Comments**, or **Work notes** fields because these UI controls are not actual fields on the incident record. These UI controls make it convenient to work with related tables. To validate these cases, use the Server test step, **Record Validation**, instead.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You

Inputs (continued)

Field	Description
	can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Table	Name of the table for the new form.
Conditions	Fields and values to be validated. Includes only fields that are visible on the open form. This field is case-sensitive.

Field State Validation

Validate the state of specified fields. States validated can include mandatory, non-mandatory, read-only, non-read-only, visible, and non-visible.

You can specify a maximum time to wait for the states of the fields to match the conditions in this step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Table	Name of the table for the new form.
Visible	Validates whether the fields on this form are visible. The test fails if the fields are not visible.
Not visible	Validates whether the fields on this form are visible. The test fails if the fields are visible.
Read only	Validates whether the fields on this form are read only. The test fails if the fields are not read only.

Inputs (continued)

Field	Description
Not read only	Validates whether the fields on this form are read only. The test fails if the fields are read only.
Mandatory	Validates whether the fields on this form are mandatory. The test fails if the fields are not mandatory.
Not mandatory	Validates whether the fields on this form are mandatory. The test fails if the fields are mandatory.

UI Action Visibility

Verify if a UI action is visible on the current form. To run this step, your test must have already opened a form using either the **Open a New Form** or **Open an Existing Record** step.

The default visible UI actions vary depending on the user that you're currently impersonating.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step

Inputs (continued)

Field	Description
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Table	Name of the table for the new form.
Visible	Validates whether UI actions on this form are visible. The test fails if the UI actions are not visible.
Not visible	Validates whether UI actions on this form are visible. The test fails if the UI actions are visible.

Declarative Action Visibility

Verify if a declarative action is visible on the current form.

The default visible declarative actions vary depending on the user that you're currently impersonating.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Notes	Notes about the test step
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace

Inputs (continued)

Field	Description
	selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Table	Name of the table for the new form.
Visible	Validates whether declarative actions on this form are visible. The test fails if the declarative actions are not visible.
Not visible	Validates whether declarative actions on this form are visible. The test fails if the declarative actions are visible.

Add Attachments to Form

Add one or more mandatory attachments to the current form. Select the attachments that the test step adds to the form from the Upload Attachments list.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Notes	Notes about the test step
Upload Attachments	Button to add one or more mandatory attachments to the form.

Note:
The Add Attachments to Form test step can't access the newly added UI workspace options.

Click Modal Button

Click a button within a modal in the specified **Form UI**.

Specify your testing by selecting either the standard platform UI or workspace UI from the **Form UI** field. If you select the standard platform UI, this test step selects the button by ID on the specified modal and validates the following conditions:

- UI page was opened in a modal.
- Button is visible and enabled.

If you select an available workspace UI, this test step selects either the Confirm or Cancel action and optionally sets the field values within the modal. This step succeeds only if a modal dialog is open on the form, and if the specified button exists on that modal dialog.

Only modals opened with the following g_modal functions are supported:

- alert
- confirm
- confirmDestroy
- showFields

Note:
Click Modal Button now supports global and scoped application modals.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds for the modal and the button to appear.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step
Form UI	Option to select an available UI.

Inputs (continued)

Field	Description
	Note: Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
UI page	The UI modal dialog to be tested. **Note:** This field appears only if **Standard UI** is selected from **Form UI**.
Button	The ID attribute of the button element. For example, the **OK**button has an ID of `OK_button` in the UI modal dialog. **Note:** This field appears only if **Standard UI** is selected from **Form UI**.
Modal values	The field values to be set in the modal. **Note:** This field appears only if a workspace is chosen as **Form UI**.
Modal action	The action to be selected in the modal. **Note:** This field appears only if a workspace is chosen as **Form UI**.
Assert type	The effect on the tests on clicking the modal button. • **Modal closed and page is reloaded or redirected**: The test passes only if the modal was closed and the page was reloaded or redirected to another page. • **Modal closed and page is not reloaded or redirected**: The test passes only if the modal was closed and the page was not reloaded or redirected to another page. • **Modal not closed**: The test passes only if the modal still exists and was not closed.
Assertion timeout	Number of seconds allowed before the assert fails. If the assert fails, the system repeats the assert until the duration of the Timeout

Inputs (continued)

Field	Description
	is reached. If the validation fails after the Timeout duration has passed, the step fails.

Note:
The Click Modal Button test step can't access the newly added UI workspace options.

Click a UI Action

Click a UI action on the current form.

When this step runs, the system performs the action normally activated by that control. The test step also validates that the current form contains the control and that the control is visible and enabled. To run this step, your test must have already opened a form using either the **Open a New Form** or **Open an Existing Record** step. It is recommended to not run this step directly after a **Submit a Form** or **Click a UI Action** step. This is because they can redirect your test to a different page based on the navigation stack configuration on your instance or the script defined in the clicked UI action. Unless you are certain that the UI action will take you to a specific page, you should explicitly use an **Open a New Form** step after **Submit a Form** or **Click a UI Action** to ensure that the test is on the form as expected. Ensure that the test keeps passing consistently when added to a suite.

In the Xanadu release, this step supports UI actions of type Form context menu.

Note:
Don't write tests that depend on the system displaying a specific page after executing a **Submit a Form** or **Click a UI Action** step. After these test steps, the system returns to the page that was open before the form was opened. The test cannot determine what that page was, so writing a test that expects a particular page can lead to unpredictable results.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of

Inputs (continued)

Field	Description
	the test step. This field appears after the test step is submitted.
Notes	Notes about the test step
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Table	Name of the table for the new form.
UI action	The UI action to click. **Note:** The declarative action and UI action buttons look exactly the same. Depending on the selected test step (either Click a UI Action or Click a Declarative Action), the type of button can be identified.
Assert type	The effect on the tests with clicking of a UI action. • **Form submission canceled in browser**: The step passes only if form submission is canceled. • **Form submitted to server**: The step passes only if the form is submitted.

Outputs

Field	Description
table	The table for the form that contains this UI action.
record	The sys_id of the record on which the action was clicked.

Click a Declarative Action

Click a declarative action on the current form.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Notes	Notes about the test step
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Table	Name of the table for the new form.
Declarative action	Option to select a declarative action from the list. **Note:** The declarative action and UI action buttons look exactly the same. Depending on the selected test step (either Click a UI Action or Click a Declarative Action), the type of button can be identified.
Assert type	The effect on the tests with clicking of a UI action. • **Form submission canceled in browser**: The step passes only if form submission is canceled. • **Form submitted to server**: The step passes only if the form is submitted.

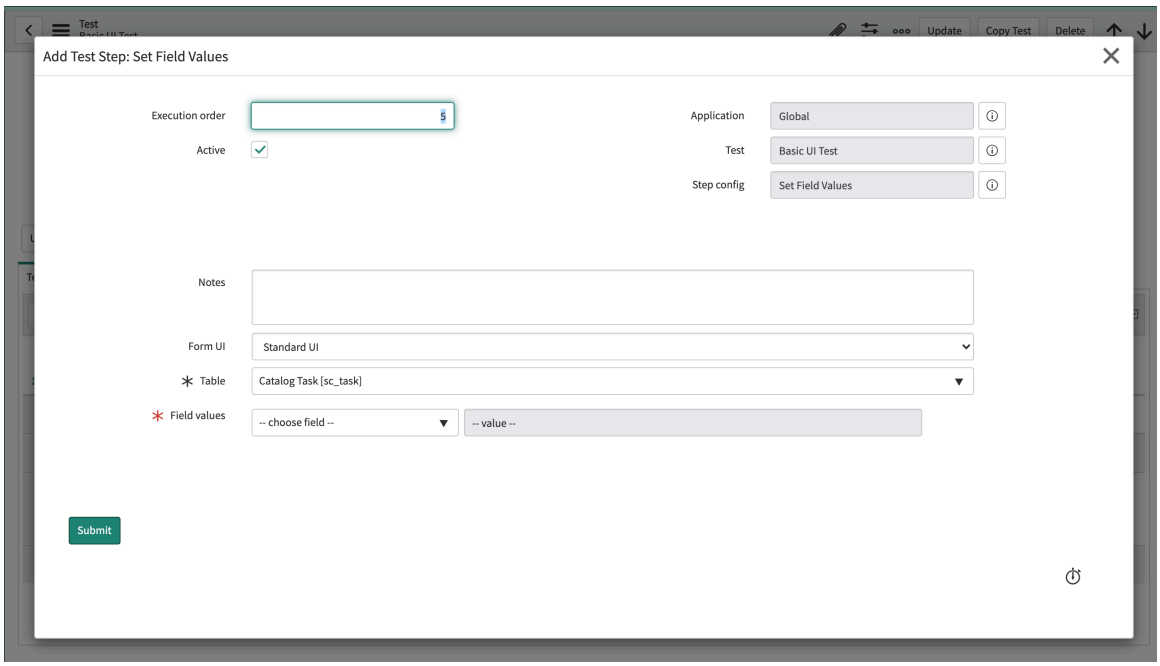
Submit a Form

Submit the current form.

To run this step, your test must have already opened a form using either the **Open a New Form** or **Open an Existing Record** step. It is recommended to not run this step directly after a **Submit a Form** or **Click a UI Action** step. This is because they can redirect your test to a different page based on the navigation stack configuration on your instance or the script defined in the clicked UI action. Unless you are certain that the UI action will take you to a specific page, you should explicitly use an **Open a New Form** step after **Submit a Form** or **Click a UI Action** to ensure that the test is on the form as expected. Ensure that the test keeps passing consistently when added to a suite.

Note:
Don't write tests that depend on the system displaying a specific page after executing a **Submit a Form** or **Click a UI Action** step. After these test steps, the system returns to the page that was open before the form was opened. The test cannot determine what that page was, so writing a test that expects a particular page can lead to unpredictable results.

A modal form appears either on top of another form or a list. To submit a modal form, your test must have already opened it on top of a form or a list.



Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.

Inputs (continued)

Field	Description
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails. **Note:** This field appears when an available workspace is chosen from **Form UI**.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step
Form UI	Option to select an available UI. Apart from the Standard UI, you have other workspace selections available. Depending on the selected workspace, the UI might vary. **Note:** Workspaces are not supported in Internet Explorer (IE). See KB0683275 for more details.
Assert type	The effect on the tests on submitting a form. • **Form submitted to server**: The step passes only if the form is submitted. • **Form submission canceled in browser**: The step passes only if form submission is canceled. • **None**: Make no assertion on submitting the form. The step always passes when using this assert type regardless of the result of submitting the form.

Outputs

Field	Description
table	The table for the submitted record.

Outputs (continued)

Field	Description
record	The sys_id of the submitted record.

Service Catalog category

Validate single catalog item transactions as well as requester and fulfiller flows in Service Catalog.

Activation of the Automated Test Framework for Service Catalog

These test steps require activation of the The Automated Test Framework Service Catalog (com.glide.automated_testing_impl.service_catalog) plugin, which is active by default on new instances. Administrators may need to activate the plugin on instances upgraded from earlier versions.

Support for parametrized tests

Service Catalog step configurations support parametrized tests. For more information on parametrized tests, refer to Parameterized tests.

Variable editor support

After opening a record that supports variable editor (requested item, catalog task, or incident), you can add step configurations to set variable values, or validate variable states or values. Use the step configurations in the following order to support variable editor.

1. Step configurations to order a catalog item or record producer in the **Service Catalog** category.
2. **Open an Existing Record** step configuration in the **Form** category.
3. **Set Variable Values**, **Validate Variable Values**, or **Variable State Validation** step configuration in the **Service Catalog** category.

Note:
Custom variables and custom variable with labels are not supported for Set Variable Values, Validate Variable Values, and Variable State Validation step configurations.

Open a Catalog Item

Open a catalog item.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.

Inputs (continued)

Field	Description
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Catalog Item	Catalog item that you want to open. **Note:** You should have access to this catalog item.

Open a Record Producer

Open a record producer.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Record Producer	Record producer that you want to open. **Note:** You should have access to the record producer.

Set Variable Values

Set variable values for the current catalog item or the record producer.

For a catalog item, use this step after opening a catalog item page using the **Open a Catalog Item** step, and before using the **Order Catalog Item** step. For a record producer, use this step after opening a record producer page using the **Open a Record Producer** step, and before using the **Submit Record Producer** step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Item	Catalog item or record producer for which you want to set variable values.
Variable Values	List of variables and the values that you want to set for them. **Note:** You can set the value for multiple variables.

Set Catalog Item Quantity

Set the quantity for the current catalog item.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.

Inputs (continued)

Field	Description
Item	Catalog item whose quantity you want to set.
Quantity	Quantity you want to set for the catalog item.

Validate Variable Values

Validate variable values on the current catalog item or record producer.

For a catalog item, use this step after opening a catalog item page using the **Open a Catalog Item** step, and before using the **Order Catalog Item** step. For a record producer, use this step after opening a record producer page using the **Open a Record Producer** step, and before using the **Submit Record Producer** step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Item	Catalog item or record producer whose variables should be validated.
Catalog Conditions	Conditions for variable validation. If the conditions are met, the test passes. **Note:** The label of a variable associated with a variable set reflects the variable set name. The format is *`variable_set_name » variable_name`*.

Variable State Validation

Validate the state of variables. Possible variable states are mandatory, not mandatory, read only, not read only, visible, and not visible.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Catalog item	Catalog item for which you want to validate the state. **Note:** You should have access to this catalog item.
Visible	List of the catalog item variables that must be visible for the step to pass.
Not visible	List of the catalog item variables that must be hidden for the step to pass.
Read only	List of the catalog item variables that must be read-only for the step to pass.
Not read only	List of the catalog item variables that must not be read-only for the step to pass.

Inputs (continued)

Field	Description
Mandatory	List of the catalog item variables that must be mandatory for the step to pass.
Not mandatory	List of the catalog item variables that must not be mandatory for the step to pass.

Validate Price and Recurring Price

Validate price and recurring price of a catalog item. Use this step after opening a catalog item page using the **Open a Catalog Item** step, and before using the **Order Catalog Item** step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Price	Price of the catalog item for the step to pass.
Recurring price	Recurring price of the catalog item for the step to pass.
Recurring price frequency	Recurring price frequency of the catalog item for the step to pass.

Add Item to Shopping Cart

Add a catalog item to the shopping cart. Use this step after opening a catalog item page using the Open a Catalog Item step. After this step, you cannot use any other steps on the catalog item.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.

Inputs (continued)

Field	Description
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Assert Type	Criteria for the test to pass. **Successfully added item to Shopping Cart** Test passes only if the catalog item is successfully added to the shopping cart. **Cannot add item to Shopping Cart** Test passes only if the catalog item cannot be added to the shopping cart.

Outputs

Field	Description
cart_item_id	The sys_id of the added catalog item.

Order Catalog Item

Clicks **Order Now** for a catalog item. Use this step after opening a catalog item page using the Open a Catalog Item step.

After this step, you cannot use any other steps on the catalog item. If the two-step checkout is false, a request is generated for the catalog item. If the two-step checkout is true, you are redirected to the cart preview page.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.

Inputs (continued)

Field	Description
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Assert type	Criteria for the test to pass. **Successfully ordered Catalog Item** Test passes only if the catalog item is successfully ordered. **Cannot order Catalog Item** Test passes only if the catalog item cannot be ordered.

Outputs

Field	Description
request_id	The sys_id of the created catalog request.
cart	the sys_id of the cart holding the catalog item.

Submit Record Producer

Submits the current record producer. Use this step after opening the record producer page using the Open a Record producer step. After this step, you cannot use any other steps on the catalog item.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.

Inputs (continued)

Field	Description
Notes	Notes about the test step.
Assert Type	Criteria for the test to pass. **Successfully submitted Record Producer** Test passes only if the record producer is submitted successfully. **Cannot submit Record Producer** Test passes only if the record producer cannot be submitted.

Outputs

Field	Description
record_id	The sys_id of target record of the record producer.

Forms in Service Portal category

Validate the functionality of fields and UI actions in Service Portal form widgets.

Service Portal dependency

Creating automated Service Portal steps requires knowledge of the ServiceNow data model and Service Portal form and data structures.

Open a Form (SP)

Opens a form in a portal.

Use this step for the base system Form page. For custom form pages, use the Open Service Portal Page step from the Custom UI category.

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.

Field	Description
Notes	Notes about the test step.
Portal	Portal in which the defined form opens. Service Portal is the default.
Page	Page to open in the defined portal. The form page is the default.
Table	Table containing the form to open.
sys_id	Sys_id of the record to open. Default is `-1`, which opens a new record.
View	The form view to open. If blank, the system uses the default view.
Query parameters	Additional URL query parameters and values for the form.

Set Field Values (SP)

Sets the values of fields in a form. To use this step, you must have already opened a form using the **Open a Form (SP)** test step.

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Table	The table for the form on which to set field values. The value should be the table in the **Open a Form (SP)** step.
Field values	Fields for which to set values and the values to be set for those fields.

Field Values Validation (SP)

Validates field values on the current form based on defined conditions. To use this step, you must have already opened a form using the **Open a Form (SP)** test step.

Field	Description
Execution order	Integer specifying the order in which the test executes this step.

Field	Description
	As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Table	The table that contains the form on which to validate fields. The value should be the table in the **Open a Form (SP)** step.
Conditions	Conditions used to validate one or more fields on the form. If the condition evaluates to true, the step passes.

Field State Validation (SP)

Validates field states on a form in Service Portal.

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Table	The table for the form on which to validate field states. The value should be the table in the **Open a Form (SP)** step.

Field	Description
Visible	Validates whether the fields on this form are visible. The test fails if the fields are not visible.
Not visible	Validates whether the fields on this form are visible. The test fails if the fields are visible.
Read only	Validates whether the fields on this form are read only. The test fails if the fields are not read only.
Not read only	Validates whether the fields on this form are read only. The test fails if the fields are read only.
Mandatory	Validates whether the fields on this form are mandatory. The test fails if the fields are not mandatory.
Not Mandatory	Validates whether the fields on this form are mandatory. The test fails if the fields are mandatory.

Add Attachments to Form (SP)

Test the functionality of attaching a file to a Service Portal form widget.

To use this step, you must have already opened a form using the **Open a Form (SP)** test step or **Open Service Portal Page steps**. This step can't be used after a **Submit Form** step or **Click a UI Action** step has been used.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Notes	Notes about the test step.
Upload Attachments	Button to attach one or more files to the form.

UI Action Visibility Validation (SP)

Determines whether a UI action on the current Service Portal form is visible. To use this step, you must have already opened a form using the **Open a Form (SP)** test step.

Service Portal only supports server UI actions. The setRedirectURL() method and client UI actions are not supported. UI action visibility can vary depending on the currently logged in or impersonated user.

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Table	The table for the form on which to check UI action visibility. The value should be the table in the **Open a Form (SP)** step.
Visible	Fields from the UI Actions table to be checked for visibility. Includes only form-based UI actions. The test fails if the UI action is not visible on the form for the currently logged in user. **Note:** If the list contains UI actions with the same name, check the form to determine the sys_id of the element. Then filter by sys_id in the UI Action table to select the correct element in the step.
Not visible	Fields from the UI Actions table to be checked for visibility. Includes only form-based UI actions. The test fails if the UI action is visible on the form for the currently logged in user. **Note:** If the list contains UI actions with the same name, check the form to determine the sys_id of the element. Then filter by sys_id in the UI Action table to select the correct element in the step.

Click UI Action (SP)

Selects a UI action on the current Service Portal form and outputs the table and sys_id of the record on which the action was selected.

To use this step, you must have already opened a form using the **Open a Form (SP)** test step. After using this step, you cannot use any other form steps.

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the

Field	Description
	order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Table	The table for the form on which to click a UI action. The value should be the table in the **Open a Form (SP)** step.
UI action	The UI action to click, selected from the UI Actions table. Includes only form-based UI actions.
Assert type	Specifies where to check for form submission after clicking the UI action: • **--None--**: Selects the UI action without validating mandatory or other fields. • **Form submission canceled in browser**: Checks whether the form was canceled in the browser and did not reach the server due to validation or other issues. • **Form submitted to server**: Checks whether the form was submitted to the server.

Outputs

Field	Description
record_id	Sys_id of the record on which the action was clicked.
table	Table with the clicked UI action.

Submit a Form (SP)

Submits the current form in a Service Portal page and outputs the table and sys_id of the submitted record.

To use this step, you must have already opened a form using the **Open a Form (SP)** test step. After using this step, the page closes. You can't use any other steps on the current page.

Field	Description
Execution order	Integer specifying the order in which the test executes this step.

Field	Description
	As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Assert type	Specifies where to check for form submission: • **--None--**: Submits the form without validating mandatory or other fields. • **Form submitted to server**: Checks if the form was submitted to the server. • **Form submission canceled in browser**: Checks whether the form was canceled in the browser and did not reach the server due to validation or other issues.

Outputs

Field	Description
record_id	Sys_id of the submitted record.
table	Table for the submitted record.

List and Related List

Validate the functionality and visibility of records and UI actions in lists and related lists.

Validate Related List Visibility

Validate the visibility of the selected related lists on a form.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step.

Inputs (continued)

Field	Description
	As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
Table	Name of the table of the parent form where related list visibility is validated.
Visible	List of related lists to assert as visible.
Not visible	List of related lists to assert as not visible.

Apply Filter to List

Apply a filter to a list to find the required record.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
List type	Option to select the type of list. • **List** • **Related list**
Assert Type	Specifies the assertion on executing the test step: • **At least one matching record found**: The step succeeds if there is at least one matching record. • **No matching record found**: The step fails if there is any matching record. • **One matching record found**: The step succeeds if there is exactly one matching record.

Inputs (continued)

Field	Description
Table	If the **List type** is **List**, this field is the table of the opened list. If the **List type** is **Related list**, this field states the name of the table of the parent form where the list belongs.
Related List	The related list to apply a filter to. **Note:** This field is available when **Related list** is selected from **List type**.
List filter	Filter conditions to apply to the list.

Outputs

Field	Description
first_record	The first record found in the list after applying the indicated filter.

Validate Record Present in List

Validate the presence of a record in a list. A valid form must be open and the list containing the record must be visible to proceed.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Step config	Name of the step.
Notes	Notes about the test step.
List type	Option to select the type of list. • **List** • **Related list**
Assert type	Specifies the assertion on executing the test step:

Inputs (continued)

Field	Description
	• **Record is present in the list** • **Record is not present in the list**
Table	Name of the parent table where the list belongs.
Related list	The related list which has the record to validate. **Note:** This field is available when **Related list** is selected from **List type**.
Record	The record whose presence is validated.

Open a Record in List

Open a specific record in a list.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
List type	Option to select the type of list. • **List** • **Related list**
Table	Name of the parent table where the list belongs.
Related List	The related list which has the record to open. **Note:** This field is available when **Related list** is selected from **List type**.
Record	The record to be opened.

Validate List UI Action Visibility

Validate that a UI action is visible in a list. If you're impersonating a user, the visibility of a UI action can change depending on the user being impersonated.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
List type	Option to select the type of list. • **List** • **Related list**
Table	Name of the table of the parent form.
Related List	The related list which has the UI actions. **Note:** This field is available when **Related list** is selected from **List type**.
Visible	List of UI actions to assert as visible.
Not visible	List of UI actions to assert as not visible.

Click a List UI Action

Select a list UI action in a list on a form.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
List type	Option to select the type of list.

Inputs (continued)

Field	Description
	• **List** • **Related list**
Table	Name of the table where the UI action belongs.
Related list	The related list which has the UI actions. **Note:** This field is available when **Related list** is selected from **List type**.
List action	The list UI action to be clicked.
Action type	The type of UI action to be clicked.
Apply to	The record on which the UI action is applied.
Assert type	The assertion made by the step when executed. • **None**: Click a UI action and continue without making an assertion. • **Page reloaded or redirected**: Click a UI action, and assert that the page was reloaded or redirected. This assert type doesn't assert form submission.
Record	The record to which the UI action is applied. **Note:** This field appears when you choose Single record in the **Apply to** field.

REST category

Verify the functionality of REST calls.

Send REST Request - Inbound - REST API Explorer

This test step begins with the REST API Explorer. Use the REST API Explorer to create and specify the HTTP method, path, query parameters, request headers, and body of a REST request, and then send the REST request to the current instance.

When you have tested the request, the **Create Automated Test Step** button appears. Click **Create Automated Test Step** to create the test step. This button does not appear until after the request has been sent. You cannot create a test step when the request payload is larger than the maximum request payload size property.

This test step creates the same test record as the **Send REST Request - Inbound** test step. After the test step is created, you cannot go back and use the REST API Explorer to update the test. All changes must be made on the **Send REST Request - Inbound** test step form.

No HTTP response validation is performed as part of this step. The step fails if the response payload size is too big, the request parameters are invalid, or the request could not be sent. Use the assert steps to validate the response.

You cannot use this step to send a request to another instance or third party/remote address.

These inputs are for the REST API Explorer. The fields you see depend upon the API selected. For more information on using REST APIs on your instance, see REST APIs.

Inputs

Field	Description
Namespace	Namespace for the request. Select from the list.
API Name	API to be used. Select from the list.
API Version	API version to be used. Select the version available on your instance from the list.
Path parameters	The part of the path after the API name. Path parameters are generally name-value pairs where the allowable values are in a list.
Query parameters	Name-value pairs of query parameters added to the URI after the path. The REST API Explorer encodes the URI, so it is not necessary to encode query parameters and values.
Request headers	Name-value pairs contained in the request header. The authentication header is set to **Send as me** to use the current user's credentials. To use the test step in production, you may need to change the **Authentication Type** field in the **Send REST - Inbound** test step. Do not encode the name or value.
Request Body	The request content. Some requests do not have a body.

Send REST Request - Inbound

Create a test step to send a REST request to the current instance. Specify the HTTP method, path, query parameters, request headers, and body of a REST request.

No HTTP response validation is performed as part of this step. The step fails if the response payload size is too big, the request parameters are invalid, or the request could not be sent. Use the assert steps to validate the response.

You cannot use this step to send a request to another instance or third party/remote address.

For more information on using REST APIs, see the REST APIs.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step.

Inputs (continued)

Field	Description
	As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Authentication Type	The type of authentication to use. For public APIs, select **None**. To configure basic or mutual authentication, users need the atf_ws_designer role. For more information, see Automated Test Framework roles.
Basic authentication	The basic authentication profile to use when doing the test. You must create or select a basic authentication profile to assign to test steps to avoid authentication issues when running the test. For more information, see Create a basic auth profile using the Automated Test Framework.
Mutual authentication	An X.509 certificate for mutual authentication. You must create or select a client certificate to assign to test steps to avoid authentication issues when running the test. For more information, see Set up Certificate-based authentication.
Method	The HTTP method to be used: • **GET** • **POST** • **PUT**

Inputs (continued)

Field	Description
	• **DELETE** • **PATCH**
Path	The path to be used. This field accepts only the portion of the URI after the instance name. If you use `https://<instance name>`, you get an error.
Query Parameters	Query parameter names and values. Do not encode the parameter names or values.
Headers	Header names and values. Do not encode the header names or values.
Body	The body of the request.

Assert Status Code

Assert that the HTTP response status code has the specified relationship to the specified value. You specify a numeric value of the status code and the relationship.

Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.

Inputs (continued)

Field	Description
Notes	Notes about the test step.
Operation	Comparison to be performed between values: • **is** • **is not** • **less than** • **greater than** • **less than or is** • **greater than or is**
Status code	Status code to be tested against the response code.

Assert Status Code Name

Assert that the HTTP response status code name has the specified relationship to the specified value. You specify a value of the status code name, and the relationship.

Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.

Inputs (continued)

Field	Description
Notes	Notes about the test step.
Operation	Comparison to be performed between values: • **contains** • **does not contain** • **is** • **is not**
Status code name	Status code name to be tested.

Assert Response Time

Assert that the HTTP response time has the specified relationship to the specified value. You specify a value of the response time and the relationship.

Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Operation	Comparison to be performed between **Response time** entered and actual response time:

Inputs (continued)

Field	Description
	• **less than** • **greater than**
Response time (ms)	Time in milliseconds to be compared to the actual response time.

Assert Response Header

Assert the HTTP response header exists, or the header has the specified relationship to the specified value.

Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Header	Header name.
Operation	Comparison to be performed between values: • **contains** • **does not contain** • **is**

Inputs (continued)

Field	Description
	• **is not** • **is not empty**
Value	Element value to be used in the test. Not shown if the **Operation** is **is not empty**.

Assert Response JSON Payload is Valid

Assert that the response payload is in valid JSON format.

Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.

Assert Response XML Payload is Well-Formed

Assert that the response payload is well-formed XML.

Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.

Assert XML Payload Element

Assert the XML response payload element exists, or has the specified relationship to the specified value.

Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.

Inputs (continued)

Field	Description
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Element path	XML path to the element to be evaluated. For example, `/result/short_description` for `{ "result": { "number": "INC0020001", "short_description": "test" } }`
Operation	Comparison to be performed between values: • **contains** • **does not contain** • **is** • **is not** • **is not empty**
Value	Element value to be used in the test. Not shown if the **Operation** is **is not empty**.

Assert JSON Response Payload Element

Assert the JSON response payload element exists, or has the specified relationship to the specified value.

Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value.

Inputs (continued)

Field	Description
	This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Element path	SNC path to the element to be evaluated. For example, `/result/short_description` for `{ "result": { "number": "INC0020001", "short_description": "test" } }` See Importing JSON files for more information.
Operation	Comparison to be performed between values: • **contains** • **does not contain** • **is** • **is not** • **is not empty**
Value	Element value to be used in the test. Not shown if the **Operation** is **is not empty**.

Assert Response Payload

Assert the HTTP response payload has the specified relationship to the specified value. You specify the value and the relationship.

Assert steps must immediately follow a **Send REST Request - Inbound** step. You can have multiple REST assert steps following a **Send REST Request - Inbound** step, but the assert steps

cannot be separated from the **Send REST Request - Inbound** step by steps from other test categories.

Note:
The entire payload is used to look for a match. A large payload can affect performance.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Operation	Comparison to be performed between values: • **contains** • **does not contain** • **is** • **is not** • **is not empty**
Response body	The value of the response body to be used in the test. Must contain the name and value to be compared as it appears in the response payload. Must not contain any curly braces. Not shown if the **Operation** is **is not empty**.

To check the short description in the response payload

```
{"result":{"number":"INC0010040","short_description":"Test ATF
 Incident"}}
```

the **Response body** should contain

```
"short_description":"Test ATF Incident"
```

These formats are incorrect and the step fails.

- `{"short_description":"Test ATF Incident"}`
- `"{"short_description":"Test ATF Incident"}"`
- `short_description: Test ATF Incident`
- `short_description:"Test ATF Incident"`

Email category

Use Automated Test Framework (ATF) to test email notifications, outbound email flows, and inbound email responses.

Validate Outbound Email

Verify that a certain outbound email exists by searching for it in the Email [sys_email] table.

Use this step to test any email-generating script that isn't captured in the Validate Outbound Email Generated by Flow or Validate Outbound Email Generated by Notification test steps.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
Conditions	Conditions used to filter outbound emails. For example, to find an email that was sent to user@example.com, set the condition to **[Recipients] [is] [user@example.com]**.
Table	Table to be queried. By default, this step queries only the Email [sys_email] table.

Note:
During testing, this step may take longer than expected to execute. The step times out after two minutes by default.

Validate Outbound Email Generated by Flow

Verify that a certain outbound email exists by searching for it in the Email [sys_email] table. Use this step to test that a flow is triggered as expected.

Note:
This step searches only for emails that are created by a flow Send Email action. To find an email that was sent after a notification was triggered in a flow, use the Validate Outbound Email Generated by Notification step instead. To find an email that was created in any other flow action, such as a custom scripted action, use the Validate Outbound Email step instead.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
Source Flow	Flow that the outbound email was created from.
Conditions	Conditions used to filter outbound emails. For example, to find an email that was sent to user@example.com, set the condition to **[Recipients] [is] [user@example.com]**.
Table	Table to be queried. By default, this step queries only the Email [sys_email] table.

Note:
During testing, this step may take longer than expected to execute. The step times out after two minutes by default.

Validate Outbound Email Generated by Notification

Verify that a certain outbound email exists by searching for it in the Email [sys_email] table. Use this step to test that an email notification is triggered as expected.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.

Inputs (continued)

Field	Description
Notes	Notes about the test step.
Source Notification	Notification that the outbound email was created from. This field references the Notification [sysevent_email_action] table.
Conditions	Conditions used to filter outbound emails. For example, to find an email that was sent to user@example.com, set the condition to **[Recipients] [is] [user@example.com]**.
Table	Table to be queried. By default, this step queries only the Email [sys_email] table.

Note:
During testing, this step may take longer than expected to execute. The step times out after two minutes by default.

Generate Inbound Email

Create an incoming email record to test an inbound email flow or an inbound email action.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
From	Address of the email sender. You can manually enter an email address or reference an email address from a user record. If you manually enter an email address, the system matches the address to an existing user record during testing.
To	Address of the email recipient. You can manually enter an email address or reference an email address from a user record. If you manually enter an email address, the system matches the address to an existing user record during testing.
Subject	Subject of the email. You can enter text or reference a string output from a previous test step.
Body	Content of the message body. You can enter text or reference a string output from a previous test step.

Output

Field	Description
output_email_record	Record in the Email [sys_email] table. The **Receive type** is **New** and the **Type** is **send - ready**. For more information on what these fields mean, see System email log and mailboxes.

Generate Inbound Reply email

Create a reply email record to test how the system handles a user response to an email notification.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
Target Table	Table of the target record.
Target Record	Record that the reply email updates. Selecting a target record also applies a watermark to the reply email.
From	Address of the email sender. You can manually enter an email address or reference an email address from a user record. If you manually enter an email address, the system matches the address to an existing user record during testing.
To	Address of the email recipient. You can manually enter an email address or reference an email address from a user record. If you manually enter an email address, the system matches the address to an existing user record during testing.
Subject	Subject of the email. You can enter text or reference a string output from a previous test step.
Body	Content of the message body. You can enter text or reference a string output from a previous test step.

Output

Field	Description
output_reply_email_record	Record in the Email [sys_email] table. The **Receive type** is **Reply** and the **Type** is **send - ready**. For more information on what these fields mean, see System email log and mailboxes.

Generate Random String

Generate a string of random alphanumeric characters that you can use as test data for another step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
Length	Number of characters in the generated string. The maximum length is 10,000 characters. If you leave the field blank, the string length is 10 characters by default.

Output

Field	Description
random_string	String of random alphanumeric characters.

Server category

Perform server-side operations. For example, query and update a record, impersonate a user, or run a server-side script.

Create a User

Create a user with specified roles and groups for the test. The user record gets rolled back after the test completes.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
First name	First name of the user.
Last name	Last name of the user.
Roles	Assigned roles of the user.
Groups	Assigned groups of the user.
Impersonate this user	Option to impersonate the new user.

Outputs

Field	Description
user	The user ID of the user impersonated.

Impersonate

Impersonate the specified user for the test.

Impersonate specifies a user for executing subsequent steps in this test. It works for both server-side and browser-side steps and stays in effect until changed with another **Impersonate** step or until the test ends. The impersonation automatically ends when the test is over.

Note:

- Do not impersonate a user with the test author role. Doing so can lead to conflicts that interfere with executing the test.
- Tests which involve impersonated users which no longer exist fail.

  Tip:
  It is recommended to create a new user to avoid data dependencies. See Create a User, for more information.

- Do not rely on user IDs being consistent across different instances. The system dynamically assigns users IDs, so the ID for a particular user often differs from one instance to the next.
- When exporting and importing automated tests, keep in mind that update sets do not update the user field.
- Tests can impersonate users with the snc_external role, which allows testing users who do not have login access. See Explicit Roles for requirements of the snc_external role.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
User	The user ID for the test to impersonate.

Outputs

Field	Description
user	The user ID of the user impersonated.

Search for a Catalog Item

Searches for a catalog item or record producer in the specified catalog and category. You can perform this step both in Platform and Service Portal.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.

Inputs (continued)

Field	Description
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Search in Portal only	Selected if the search is restricted to the Service Portal only. Otherwise, not selected.
Search term	Term used to search for a catalog item.
Catalog	Catalog in which to search for the catalog item.
Category	Category in which to search for the catalog item.
Assert item	Catalog items that should be available in the search results.
Assert Type	Specifies how searching the catalog item affects the test: • **Assert Item present in search result**: Test passes only if the assert item is present in the search result. • **Assert Item not present in search result**: Test passes only if the assert item is not present in the search result.

Outputs

Field	Description
catalog_item_id	The sys_id of the first catalog item found that the user can view.

Record Query

Query the database to verify that a record exists matching the conditions set in this step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.

Inputs (continued)

Field	Description
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Enforce security	Selected to enforce ACLs. Otherwise, not selected.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Assert type	Specifies how querying the record affects the test: • **There is at least one record matching the query**: The test fails if there are no records matching the query. • **No records match the query**: The test fails if any records match the query.
Table	The table to be queried.
Conditions	Conditions used to run the query.

Outputs

Field	Description
table	The table queried.
first_record	The first record resulting from the query.

Note:
If you don't update your record query test step, the original record query test step still functions the same way as before, irrespective of it being termed as **(Deprecated)**.

Record Insert

Inserts a record into a table with the field values you specify.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.

Inputs (continued)

Field	Description
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Assert type	Specifies how inserting the record affects the test: • **Record successfully inserted**: Test fails if the record was not successfully inserted. • **Record was not inserted**: Test fails if the record was successfully inserted.
Enforce security	Selected to enforce ACLs and the read-only role. Otherwise, not selected.
Table	The table into which the record should be inserted.
Conditions	Specific field values to be set when the test runs this step.

Outputs

Field	Description
table	The table to which the new record belongs.
record_id	The sys_id of the new record.

Record Update

Changes field values on a record on the server.

Note:
To ensure that the changes were applied, follow this step with a **Record Validation** step.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.

Inputs (continued)

Field	Description
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Assert type	Specifies how updating the record affects the test: • **Record successfully updated**: Test fails if the record was not successfully updated. • **Record was not updated**: Test fails if the record was successfully updated.
Enforce security	Selected to enforce ACLs and the read-only role. Otherwise, not selected.
Table	The table containing the record to be updated.
Record	ID of the record to be updated.
Field values	Fields for which you want to set values and the values you want to set for those fields. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.

Note: Record Update step succeeds even if a field on the record is blocked by ACL. Use the Record Validation step after Record Update to check whether a particular field was modified, or use the Form steps to evaluate ACL conditions for individual fields on a record.

Record Delete

Deletes a specified record in a table.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step.

Inputs (continued)

Field	Description
	As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Assert type	Specifies how updating the record affects the test: • **Record successfully deleted**: Test fails if the record was not successfully deleted. • **Record was not deleted**: Test fails if the record was successfully deleted.
Enforce security	Selected to enforce ACLs and the read-only role. Otherwise, not selected.
Table	The table containing the record to be deleted.
Record	ID of the record to be deleted.

Record Validation

Validates that a record meets the specified conditions on the server side.

For the **Record Validation** step, specify the values you want to test using the standard conditions builder. You can apply several conditions to the same field.

This step passes if the overall condition is satisfied and fails if it is not. If you need to test the values of individual fields independently of each other, include a separate **Record Validation** step for each value to be tested.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.

Inputs (continued)

Field	Description
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step. **Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.
Assert type	Specifies how validating the record affects the test: • **Record successfully validated**: The test fails if the record does not match the conditions. • **Record not found**: Test fails if the record is found.
Table	The table that contains the field to be validated.
Record	The record that contains the field to be validated.
Field values	Specific fields to be validated when the test runs this step.

Run Server Side Script

Executes a script on the server.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Jasmine version	The version of the Jasmine testing framework to use for running the server-side script. Any new scripts you create use Jasmine version 3.1. Your existing scripts

Inputs (continued)

Field	Description
	can continue to use Jasmine version 1.3, or you can upgrade them to Jasmine version 3.1.
Test script	The javascript for the server to execute. Supports the Jasmine testing framework. **Note:** steps(SYS_ID) can be defined as a function to retrieve Output variable data from a step that executed earlier in the test. The Run Server Side Script test step now supports parameters as step inputs.

Outputs

Field	Description
record_id	The sys_id output by the server-side script.
table	The table output by the server-side script.

Note:
If the script creates data, the system rolls back that data after all steps in the test finish. The output for the Run Server Side Script test step now indicates the line and the line number for the cause of failure.

- Test script

```
//  Test step 1 - add data
var now_GR = new GlideRecord('sc_task');
//  this sample step's Step config has Output variables named
 table and record_id
outputs.table = 'sc_task';
outputs.record_id = gr.insert();
//  Test step 2 - access added data and validate
//  check that the record exists (or that business logic
 changed it)
var now_GR = new GlideRecord("sc_task");
gr.get(steps(PREVIOUS_STEP_SYS_ID).record_id);
assertEqual({name: "task gr exists", shouldbe: true, value:
 gr.isValidRecord()});
stepResult.setOutputMessage: Log a message to step results
 after step executes.
                            Can only be called once or will
 overwrite previous message
```

- Create a suite of test scripts

```
var now_GR = new GlideRecord('sc_task');
gr.setValue('short_description', 'verify task can be
 inserted');
var grSysId = gr.insert();
var justCreatedGR = new GlideRecord('sc_task');
if (justCreatedGR.get(grSysId)) {
stepResult.setOutputMessage("Successfully inserted task
 record");
```

```
return true; // pass the step
} else {
stepResult.setOutputMessage("Failed to insert task record");
return false; // fail the step
}
```

- Jasmine test

```
describe('my suite of script tests', function() {
it('should meet expectations', function() {
expect(true).not.toBe(false);
});
});
//  make sure to uncomment jasmine.getEnv().execute(); outside
 the function body
assertEqual: A function used to compare that assertion.shouldbe
 == assertion.value;
in case of failure it throws an Error and logs that the
 assertion by name has failed
```

Note:
`describe` is only supported in Global scope. Use `describe` to create a suite of test scripts and `it` to define test expectations.

- Logs message to test step output

```
var testAssertion = {
name: "my test assertion",
shouldbe: "expected value"
value: "actual value",
};
assertEqual(testAssertion); // throws Error, logs message to
 test step output
```

- See Step Execution Scripts for Run Server Side script example.

Replay Request Item

Get the item and requester from an existing request item, add the item to a new cart for that user, and place an order.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.

Inputs (continued)

Field	Description
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Original Request Item	The request item to be replayed.

Outputs

Field	Description
table	The table to which the replayed request item belongs.
request	The replayed request item.

Related topics

Automated Test Framework use case: test a Service Catalog request

Log

Logs a message and stores it as a step result.

The log message can contain variables and other information pertaining to the test. The message is stored as a step result.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
Log	The message to be logged. To include the value of an output variable from a previous step, click the input value icon () and follow the procedure to Pass values from one automated test step to another.

Add Attachments to Existing Record

Add one or more mandatory attachments to the specified record. Use **Upload Attachments** to select from the attachments the test step adds to the record.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them. You can change this default order by editing the **Execution order** values.
Application	Application scope in which the system runs this step.
Active	Option to activate this test step for use.
Test	Name of the test that you're adding the step to.
Step config	Name of the step.
Notes	Notes about the test step.
Table	Valid table selection from the list.
Record	Pre-existing record either from before the test or inserted as a part of the test.
Upload Attachments	Button to add one or more mandatory attachments to the record.

Checkout Shopping Cart

Submits the cart and generates a request. You can perform this step both in the ServiceNow AI Platform and Service Portal.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Timeout	Number of seconds allowed before the step fails. If the validation fails, the system repeats the step until it reaches the duration of the timeout. If the validation fails after the timeout duration has passed, the step fails.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.

Inputs (continued)

Field	Description
Assert Type	Criteria for the test to pass. **Empty cart** Test fails if the cart is not empty. **Successfully Checkout cart** Test fails if the cart is not successfully checked out.
Requested For	User for whom the request is generated.
Delivery Address	Delivery address for the request.
Special Instructions	Special instructions for the request.

Outputs

Field	Description
request_id	The sys_id of the submitted request.

Custom Scripted StepConfig

Provides an example of scripts for a custom step configuration.

This example checks if the user name provided starts with the letter A. This step is useful primarily to users with the [atf_test_admin] role. Users with the [atf_test_admin] role can view the example scripts by opening the record for this step in Step configurations.

Inputs

Field	Description
Execution order	Integer specifying the order in which the test executes this step. As you create steps, the system automatically assigns each step an incremental value. This value causes the test to execute steps in the order that you created them in. You can change this default order by editing the **Execution order** values.
Active	Option to activate this test step for use.
Application	Application scope in which the system runs this step.
Test	Read-only name of the test that you're adding the step to.
Step config	Read-only name of the step.
Description	Description of the test step. This field value is automatically set based on the field values of the test step. This field appears after the test step is submitted.
Notes	Notes about the test step.
User	The user whose name the system checks to see if it starts with the letter **A**. To include the value of an output variable from a previous step, click the input value

Inputs (continued)

Field	Description
	icon ( ) and follow the procedure to Pass values from one automated test step to another.

Outputs

Field	Description
value	The name of the user.

Related topics

Create custom step configuration

Step execution scripts

Step description generation script

Automated Test Framework reference

Reference information for the Automated Test Framework.

Watch https://players.brightcove.net/6274575390001/nUx4EKfUz_default/index.html?videoId=6286204365001 for some helpful tips to use ATF features.

Tables excluded from rollback after running an automated test

The Automated Test Framework tracks data created by running tests and rolls back changes after testing. The system excludes certain tables from being tracked during testing.

The system excludes certain tables from being tracked or rolled back:

- The History [sys_history_line] table
- The ECC Queue table [ecc_queue].
- The Email [sys_email] Email Log [sys_email_log] tables
- The Report Executions [report_executions] and ReportStats [report_stats] tables.
- The Execution Tracker [sys_execution_tracker] tables
- The Progress Worker [sys_progress_worker] table
- The Schema Change [sys_schema_change]
- The Upgrade History [sys_upgrade_history] and Upgrade History Log [sys_upgrade_history_log] tables
- The Mutex [sys_mutex] table
- The Plugin Log [sys_plugin_log] table
- The Status [sys_status] table
- The Number Counter [sys_number_counter] table
- The AMB Channel Presence [sys_amb_channel_presence] table
- Any table that extends an excluded table.
- Any table starting with the following prefixes are also excluded:

- syslog
- sys_amb_message
- sys_cluster
- cmdb_metric
- ts_
- v_
- sys_delete_recovery

If your test run changes (inserts/updates/deletes) any record on these excluded tables, the system does not roll back the change after testing.

Tests

The Tests module opens the Test table. From here, you can add, edit, and run tests. By opening an individual test record, you can view and edit the steps comprising that test.

Related topics

Building and running automated test suites

Test record form

In the Test record form, you view and edit values of fields for the test record.

Fields

Field / UI Element	Description
Name	Test name.
Description	Test description.
Active	If this test is active, **true**. Otherwise, **false**.
Application	Application scope in which the system runs this test or test suite.
Test steps related list	The steps that this test executes.
Test results related list	The results from individual executions of this test.
Fail on server error	The errors detected while performing UI steps on the server that occur and fail the test.

UI Actions

Name	Description
Update	Click to update the test record.
Run test	Click to run test.
Copy test	Click to copy test.
Delete	Click to delete this test.

Suites

The Suites module opens the Test Suites table. You can create, edit, and run test suites from this table.

Related topics

Building and running automated test suites

Test suite form

The Test Suite form contains information about one test suite.

Fields

Field / Element	Description
Name	The name of the test suite.
Active	To make this test suite active, check this field.
Description	(Optional) Enter a description to identify the purpose of this test suite.
Application	Application scope in which the system runs this test or test suite.
Parent Suite	To make this test suite a child of another test suite, enter the name of the parent test suite here.
Test Suite Tests (related list)	The tests included in this test suite.
Child Test Suites (related list)	Any test suites that are children of this test suite.
Test Suite Results (related list)	Results from executing this test suite.
Test Suite Schedules (related list)	Any test suite schedules that include this test suite.

Test results

Each time you run a test, the automated test framework creates a record of the test results. Use the Test Results module to view details about the results of individual tests and individual steps within a test. If the test generated screenshots, they appear as attachments on the test results record.

Each Test Results record displays detailed results information about one test execution, and contains links to detailed step results and test logs. Step results records display information about one step in a test result, while test log (test results item) records contain detailed console logging and test execution information.

By default, the system deletes test and test suite results data 30 days after creation. You can modify this default retention policy as needed in the Table Cleanup module.

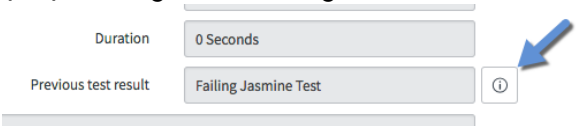
Related topics

Modify data retention policy for ATF test results

Table Cleanup module

Test results record

A Test Results record contains detailed results information about one test execution. Client Error Details and Failure Details sections appear when a test fails due to error conditions, and include detailed error information you can use for troubleshooting purposes.

Field	Description
Test	Name of the test.
Status	Result of the Test execution: **Success** Test passed. **Failure** Test failed. **Success with warning(s)** Test run passed but encountered client error(s) that were allowed as warnings in the Allowed Client Error table. **Waiting** Test waiting to start. **Running** Test is running. **Skipped** Occurs if an earlier test in the suite failed and **Abort on failure** is set to **true**. **Error** An error occurred in the test framework. For example, the test runner halted, or the server encountered an unintended scenario. The error message appears in the Summary field of the steps results record for the step that threw the error. **Cancelled** User canceled the Test.
Retain indefinitely	Check box to keep this record even after the specified data retention period has passed. For more information, see Table Cleanup.
Start time	Date and time of day at which this test started executing.
End time	Date and time of day at which this test finished executing.
Duration	Elapsed time it took to execute this test.
Previous test result	Results of the previous test execution, indicating if the test passed or failed. Only appears if this test result is for a re-run of a failed test and you are logged on with the admin, atf_test_admin, or atf_test_designer role. Click the information icon (ⓘ) to navigate to the original test result. 

Note:
If this test created screenshots, they appear as attachments in **Manage attachments**.

Client Error Details

This section only displays when a test fails due to a client error.

Field	Description
Failing step	Reference to the first test step result that failed during this test. Click the information icon ( ) to view detailed step results and error information in the Step Results form.
Summary	The following text indicates that a client error occurred at this step, and includes the detailed error message: `This step failed because the client error 'DETAILED ERROR MESSAGE' was detected on the page being tested. See failing Test Logs. To ignore these errors in the next test run, use 'Add all client errors to warning/ ignored list' links.` Test designers and developers should always investigate client errors to determine if there are issues with your business process. For more details, see Identify and resolve client errors.
First failing client error	Reference to the first client error that failed during the test. Click the information icon ( ) to view detailed test results and error information in the Test Logs form. **Note:** For details on how to allow client errors as ignored or warning entries, refer to Allowed client errors
Failing step screenshot	Screenshot of the step that failed. Click to download the screenshot.

Failure Details

This section only displays when a test fails due to a cause other than a client error.

Field	Description
Failing step	Reference to the first step result that failed during this test. Click to download the screenshot.
Summary	Output of the step that failed.
Failing screenshot	Screenshot of the step that failed.

Additional information

This section is visible for all test results.

Field	Description
Output	Output generated for the test step. • Test result (often `Passed` or `Failed`). • For failed tests, includes the same error message detail displayed in the Summary field for the first test step that failed. See Client Error Details above.
Browsers involved	User agent strings returned by browsers that ran the test.

Related lists

Related list	Description
Step Results	Record for each step result and log entry for this test.
Test Log	Record for each test log related to this test results record.
Test Transactions	Record for each transaction (from the system transaction log [syslog_transaction] related to this record. To view the step results associated with a transaction, click the appropriate link in the **Step Results** column. To view the transaction logs associated with a transaction, click the appropriate link in the **Transaction** column. **Note:** The system may not be able to log some transactions with short durations.
Warnings	List of test logs containing client errors with a warning status. The Warnings related list only appears on test results with warnings.

Related topics

- View test results and automated test results
- Test logs record
- Step results record
- Allowed client errors
- Allow client errors from step results
- Allow client errors from the test logs

Step results record

The Step Results record contains information about one step in a test result. You access specific step results from the Step Results related list in the Test Results record.

Field	Description
Status	Result of the step: **Success** Test step passed.

Field	Description
	Failure Test step failed. **Success with warning(s)** Test step passed but encountered client error(s) that were allowed as warnings in the Allowed Client Error table. **Skipped** Occurs if an earlier test in the suite failed and **Abort on failure** is set to **true**. **Error** An error occurred in the test framework. For example, the test runner halted or the server encountered an unintended scenario. **Cancelled** User canceled the test or suite.
Type	Type of test result item: Step Result
Summary	Summarized version of the Output field. If an error occurred in the test step, the following text indicates that a client error occurred at this step, and includes a detailed exception/error message. `This step failed because the client error 'DETAILED ERROR MESSAGE' was detected on the page being tested. See failing Test Logs. To ignore these errors in the next test run, use 'Add all client errors to warning/ignored list' links.` Test designers and developers should always investigate client errors to determine if there are issues with your business process. For more details, see Identify and resolve client errors. For a Send REST Request - Inbound step, the URL for the request endpoint, and a response code. For example: `The HTTP request has been sent to the endpoint 'https://demonightlyus.service-now.com/api/now/table/kb_knowledge'`
Output	Generated output for the test step. • For a step result, the outputs from the step, including any explanation why a step failed or was skipped. • For a Send REST Request - Inbound step, the REST request and response including the response body. The output field is truncated at 4096 characters. ○ The REST request and response headers are filtered to prevent sensitive information from being added to the log. A filtered header text is replaced with the text "Header redacted for security." ○ See Filter REST request and response headers for information on how to add a list of REST request and response headers to be filtered. For additional console logging and test execution information for a test step, see the Test Logs record.

Field	Description
Step	Name of the step executed.
Test Result	Reference to the test result associated with this step result. Click the information icon ( ) to view detailed test results information.
Description	Description of the specific actions performed (for example, Open Create Incident Report Producer, Validate that the form matches the following condition) for this test step. For a client log, blank.
Start Time	Date and time of day at which this step or log entry started.
End Time	Date and time of day at which this step or log entry ended.
Duration	Elapsed time it took to execute this test.
Step Transactions (related list)	Record for each transaction (from the system transaction log [syslog_transaction] related to this step result.

Related topics

- Test results record
- Test logs record
- Allowed client errors
- Allow client errors from step results

Test logs record

The Test Results Item (test log) record contains console logging and test execution information.

During test execution, any information reported to the environment is recorded in the Test Log (sys_atf_test_result_item) table. This information can include browser console logging, results, and error messages recorded by step environments. You access specific test logs from the Test Log related list in the Test Results form.

Field	Description
Status	The result of the test log: **Success** Test step run passed. **Failure** Test step run failed. **Success with warning(s)** Test step passed but encountered client error(s) that were allowed as warnings in the Allowed Client Error table. **Ignored** Client error that has been allowed with a report level of Ignored. **Warning** Client error that has been allowed with a report level of Warning. **Waiting** Test or suite waiting to start.

Field	Description
	Running Test or suite is running. **Skipped** Occurs if an earlier test in the suite failed and **Abort on failure** is set to **true**. **Error** An error occurred in the test framework. For example, the test runner halted or the server encountered an unintended scenario. The error message appears in the Summary field of the Steps Results record for the step that threw the error. **Cancelled** User canceled the test or suite.
Type	Type of test log: • Step Result • Client Log • Client Error **Note:** If a Client Error, you can optionally add it as an ignored or warning entry in the Allowed Client Errors. Doing so prevents the allowed client errors from affecting ATF test executions when they recur in future test runs. For more details, refer to Allowed client errors.
Output	Output generated for the test log. • For a step result, the console logging and test execution outputs from the step, including any explanation why a step failed or was skipped. • For a client log, the log entry text. • For a client error, the actual client error is displayed.
Step	Name of the step executed. This field may be blank for a client log.
Test Result	Reference to the test result associated with this step result. Click to view detailed test result information.
Description	For a step result, the actions performed in this test step. For a client log, blank. For a client, this message displays: `This client error occurred on the page in Browser Type Browser Version` (for example, Chrome 62.0.3202.62).
Start time	Time at which this test step started executing.
End time	Time at which this test step finished executing.
Duration	Time duration it took to execute this test step.
Recorded at	Time at which this step or log entry was recorded.

Field	Description
Allow listed client error	Reference to the Allowed Client Error record (if any). Click ⓘ to view the Allowed Client Error record for this client error.

Related topics

- Test results record
- Step results record
- Allow client errors from the test logs
- Allowed client errors

Suite results

The Suites Results module opens the Suites Results table.

You can drill down to the Test results record to view details about the results of individual test suites, the individual tests within those test suites, any child test suites, and so on.

Related topics

- Building and running automated test suites

Test suite results record

The Test Suite Results record displays information about the results of one execution of one test suite.

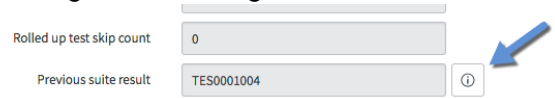
Fields

Field / Element	Description
Start Time	(Appears in list of records, but not on default record form). The time this test suite started.
Test Suite	The test suite that was run.
Number	Unique ID for this test suite results record.
Base Test Suite Result	If this test suite is a child in a hierarchy of test suites, the base test suite result is the unique ID of the result record for the suite at the top-most level of the hierarchy. For more information, see Example: Base test suite result.
Parent	If this test suite has a parent, this is the test result record for the parent suite. For more information, see Example: Parent test suite results.
Status	Result of the test or test suite execution: • Canceled – user canceled the test or test suite run. • Error – an error occurred in the test framework. For example, the test runner halted or the server encountered an unintended scenario. The error message appears in the summary field of the Step Results form for the test step that contains the error. • Failure – test, test suite run, or test step failed due to an error. • Ignored – test step passed but contains client-side JavaScript errors entered into the Allowed Browser Error table with report level of Ignored. • Running – test or test suite is running.

Fields (continued)

Field / Element	Description
	• Skipped – occurs if an earlier test in the suite failed and **Abort on failure** is set to **true**. • Success – test, test suite, or test step run passed. • Success with warning(s) – test or test suite run passed but contains a test step with client-side JavaScript errors entered into the Allowed Browser Error table with report level of Warning. • Waiting – test or suite run is waited to start. • Warning – test or suite contains client-side JavaScript errors entered into the Allowed Browser Error table with report level of Warning.
Run time	The duration it took to execute this test suite.
Retain indefinitely	Check box to keep this record even after the specified data retention period has passed. For more information, see Table Cleanup.
Rolled up test success count	How many tests were successful. The tests counted as part of the roll up are all tests included in this suite, plus all others included in suites that are descendents of this one. For more information, see Rolled up counts for test suites results.
Rolled up test failure count	How many tests failed. The tests counted as part of the roll up are all tests included in this suite, plus all others included in suites that are descendents of this one. For more information, see Rolled up counts for test suites results.
Rolled up test error count	How many tests resulted in an error. The tests counted as part of the roll up are all tests included in this suite, plus all others included in suites that are descendents of this one. For more information, see Rolled up counts for test suites results.
Rolled up test skip account	How many tests were skipped. The tests counted as part of the roll up are all tests included in this suite, plus all others included in suites that are descendents of this one. For more information, see Rolled up counts for test suites results.
Test Results (related list)	Results of the individual tests included in this test suite.
Child Test Suites Results (related list)	The results of any test suites that are children of this test suite.
All Test Suite Results (related list)	Results from this test suite and all descendent test suites.
Failed Tests in Suite	Results from any failed tests included in this test suite.

Fields (continued)

Field / Element	Description
(related list)	
Previous suite result	Only appears if this suite result is for a re-run of failed tests and you are logged on with the atf_test_admin, atf_test_designer, or admin role. Click the information icon to navigate to the "original" suite result record. 

Related topics

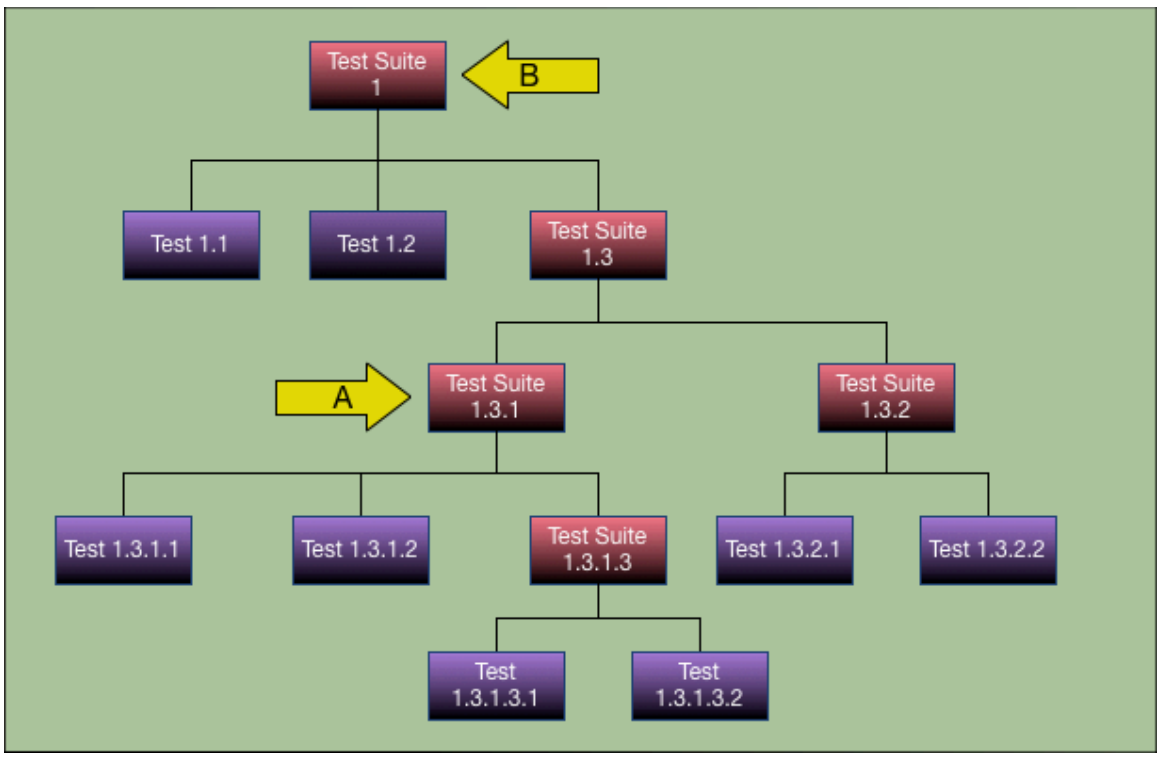
Test results

Test suite results examples

Examples of relationship terms and how aggregated results roll up for test suites.

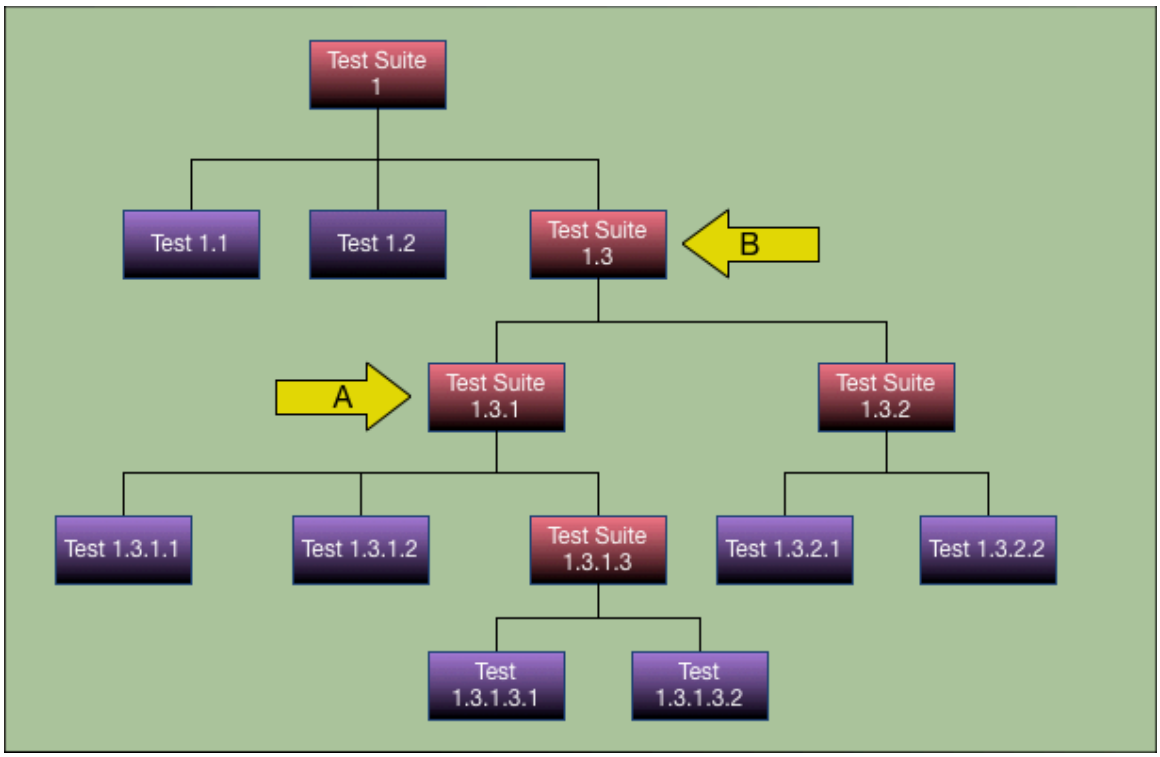
Example: Base test suite result

The base test suite is the top-level test suite in the hierarchy. For example, if you are viewing the test suite results record for (A) Test Suite 1.3.1, the **base test suite result** field links to the test suite results record for (B) Test Suite 1.



Example: Parent test suite results

The parent test suite is the test suite immediately above the one you are currently viewing. For example, ff you are viewing the test suite results record for (A) Test Suite 1.3.1, the **parent** field links to the test suite results record for (B) Test Suite 1.3.

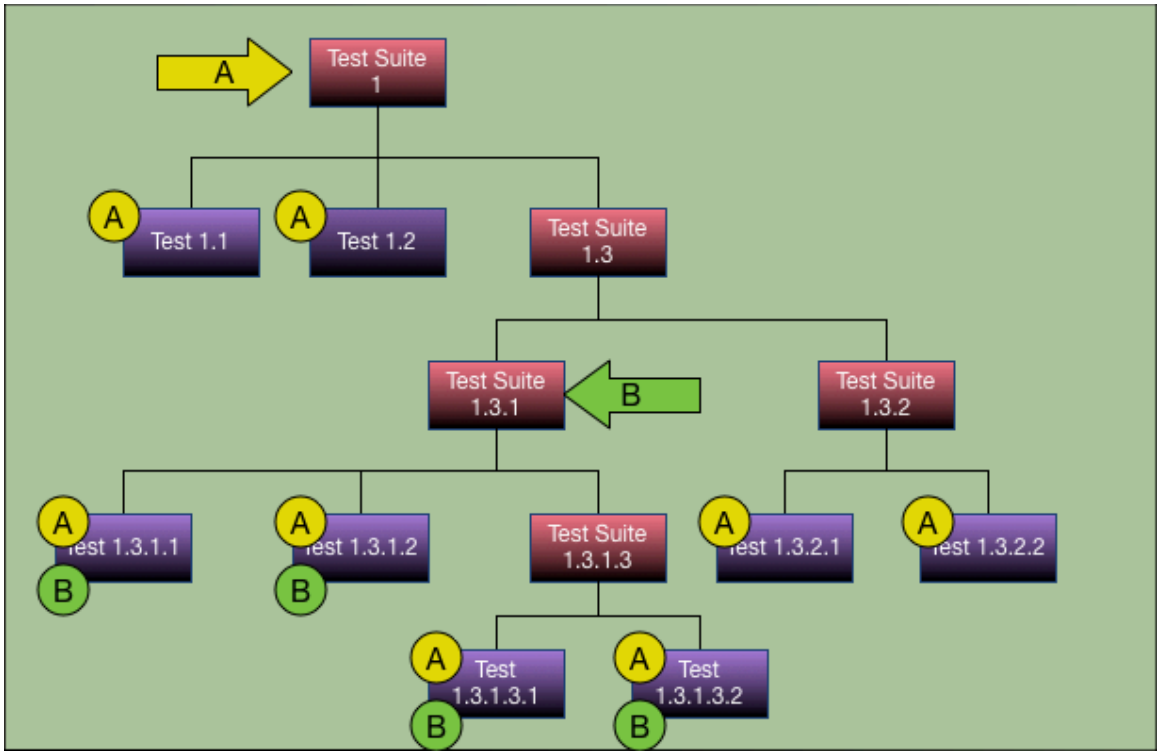


Rolled up counts for test suites results

You can view rolled up counts for a test suite for four metrics: test successes, failures, errors, and skips. Each of these sum data from all tests in the test suite plus all tests in the test suite's descendents.

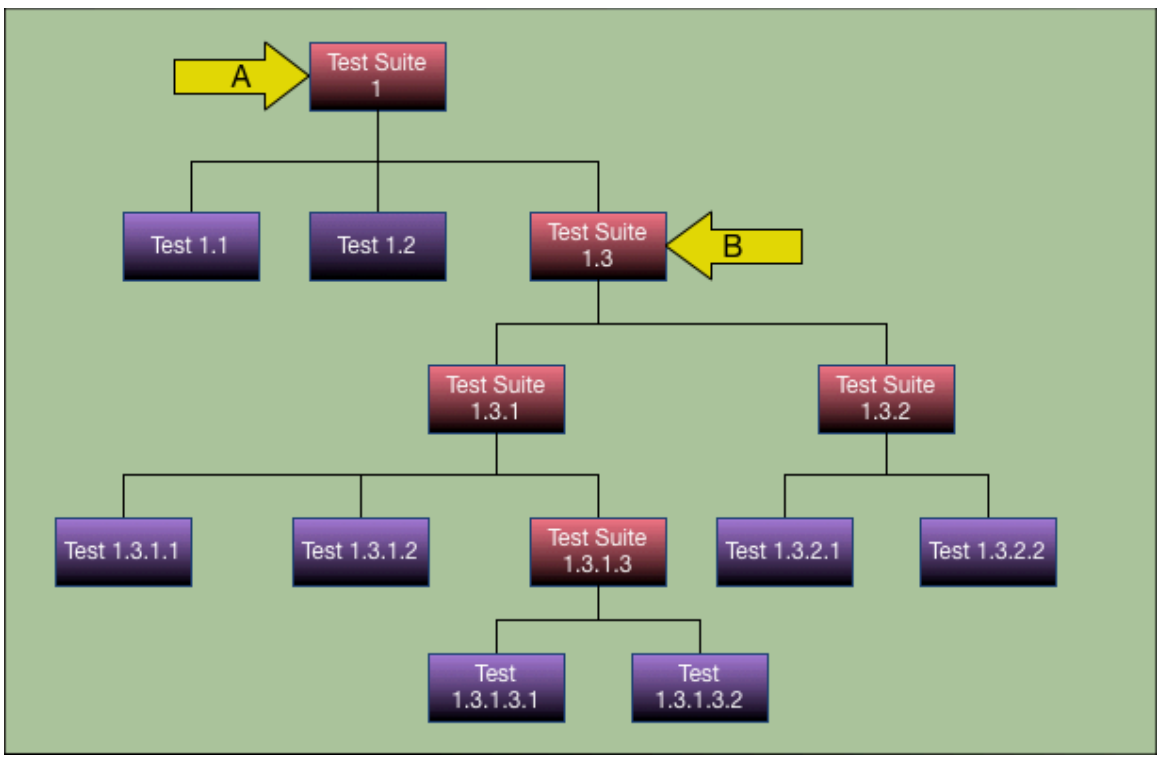
For example, if you are viewing the test suite results record for (A) Test Suite 1.3.1, the **Rolled up test success count** field shows the total number of successes counting results from all the tests represented by boxes labeled A.

If you are viewing the test suite results record for (B) Test Suite 1.3.1, the **Rolled up test success count** field shows the total number of successes counting results from all the tests represented by boxes labeled B.



Child Test Suites Results

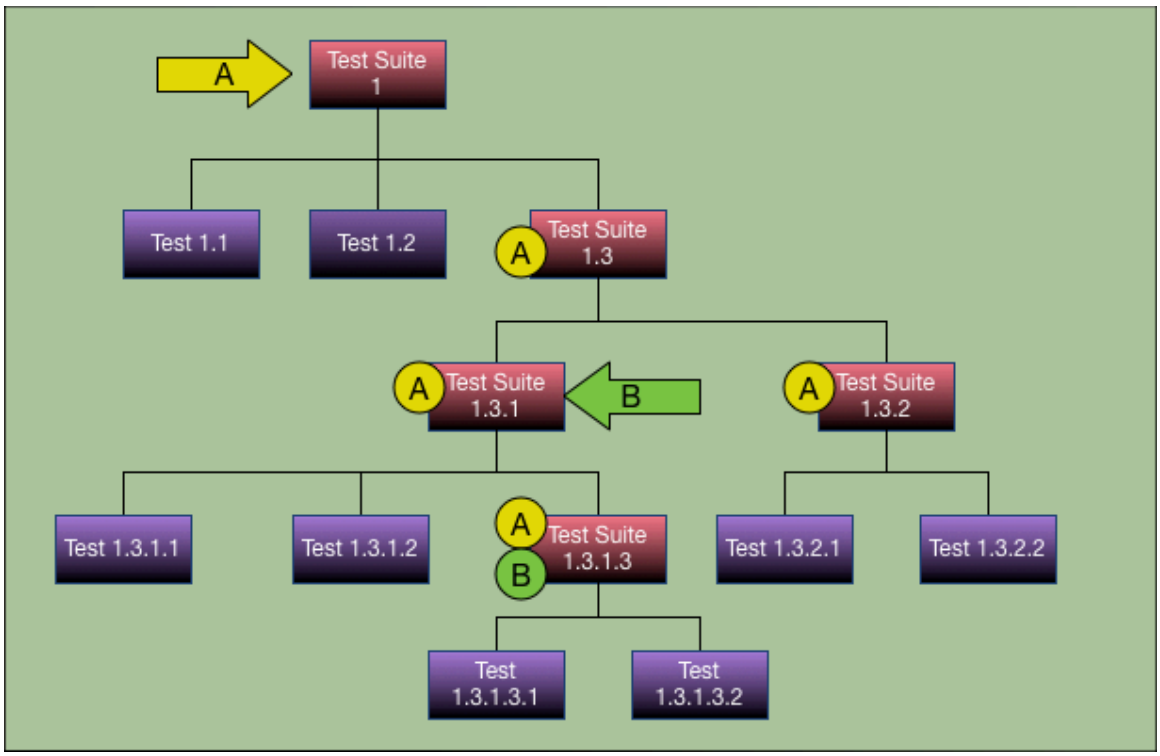
If you are viewing the test suite results record for (A) Test Suite 1, the **Child Test Suites Results** related list shows result records for Test Suite 1.3 (B).



All Test Suites Results

If you are viewing the test suite results record for (A) Test Suite 1, the **All Suites Results** related list shows result records for all Test Suites represented by boxes labeled A

If you are viewing the test suite results record for (B) Test Suite 1.3, the **All Suites Results** related list shows result records for all Test Suite 1.3.1 only (represented by the box labeled B)



Suite schedules

Open the Suites Schedules table. You can drill down to see details about the results of individual schedules or create a new schedule.

Related topics

Building and running automated test suites

Suite schedule record

The Suite Results record displays information about one test suite schedule.

Fields

Field / Element	Description
Name	A name for this test suite schedule.
Description	A description for this test suite schedule.
Active	If this schedule is active **true**. Otherwise, **false**.
Application	The application scope for this test suite schedule.
Run	The frequency with which the system runs test suites belonging to this schedule.
Run as tz	The timezone used for the **Time** field.
Time	The time of day at which the system runs test suites belonging to this schedule.

Fields (continued)

Field / Element	Description
Day	If **Run** is set to Weekly, the day of the week on which the system runs test suites belonging to this schedule. If **Run** is set to monthly, the day of the month.
Repeat interval	If Run is set to **Periodically,**the number of days and hours that constitute the repeat interval for running suites in this schedule.
Conditional	Check to enable a script to define conditions under which the system should run the test suites in this schedule.
Condition	If **Conditional** is checked, the script to execute for determining the conditions under which the system should run the test suites in this schedule.
Scheduled Suite Runs (related list)	The test suites the system should run on this schedule.

Related topics

Schedule an automated test suite

Scheduled suite run record

A Scheduled Suite Run record associates a Suite Schedule record with a Test Suite.

Fields

Field / Element	Description
Schedule	The schedule to use.
Test Suite	The test suite to run.
Application	The application scope for this scheduled suite run.
Watch list	Users the system notifies when this scheduled suite run completes.
Browser name	If the test suite has UI components, the browser that must be used. If no client test runner is available with this browser, the system does not run the suite.
Browser version starts with	If the test suite has UI components, the browser version for the client test runner must start with this string. If no client test runner is available with this browser version, the system does not run the suite.
OS name	If the test suite has UI components, the client test runner must run under this OS. If no client test runner is available with this OS, the system does not run the suite.
OS version starts with	If the test suite has UI components, the OS version for the client test runner must start with this string. If no client test runner is available with this OS version, the system does not run the suite.
Test Suite Results (related list)	All test suite results from this scheduled run.

To determine the browser name and version of a browser you want to use, start a scheduled test runner with that browser, then inspect that runner's record in the Active Scheduled Test Runners Module.

Related topics

Schedule an automated test suite

Run

Start a client test runner and view information about test runners and test runs.

For information on how to work with test runners, see Working with client test runners.

Client test runner

The Client Test Runner opens a browser window or tab for running manually-started client automated tests.

You can toggle a client test runner to act as either a manual or scheduled client test runner. For more information about scheduling test suites, see Schedule an automated test suite.

Fields on the Client Runner Test window

Field / UI Element	Description
Form preferences icon ()	Click to display the form preferences panel.
Form preferences panel: Screenshots mode	Choose among: • Enable for all steps • Enable for failed steps • Disable for all steps For additional information, see Set the system property to control when the Automated Test Framework captures screenshots
Form preferences panel: Run scheduled tests only	Click to toggle between: • On (green): Use this client test runner to run only scheduled tests and suites. • Off (gray): Use this client test runner to run only manually-started tests and suites.

Scheduled client test runner

The Scheduled Client Test Run opens a browser window for running scheduled client automated tests.

For information about scheduling automated tests, see Schedule an automated test suite and Working with scheduled test suites. You can toggle a client test runner to act as either a manual or scheduled client test runner.

Fields on the Client Runner Test window

Field / UI Element	Description
Form preferences icon ( )	Click to display the form preferences panel.
Form preferences panel: Screenshots mode	Choose among: • Enable for all steps • Enable for failed steps • Disable for all steps For additional information, see Set the system property to control when the Automated Test Framework captures screenshots
Form preferences panel: Run scheduled tests only	Click to toggle between: • On (green): Use this client test runner to run only scheduled tests and suites. • Off (gray): Use this client test runner to run only manually-started tests and suites.

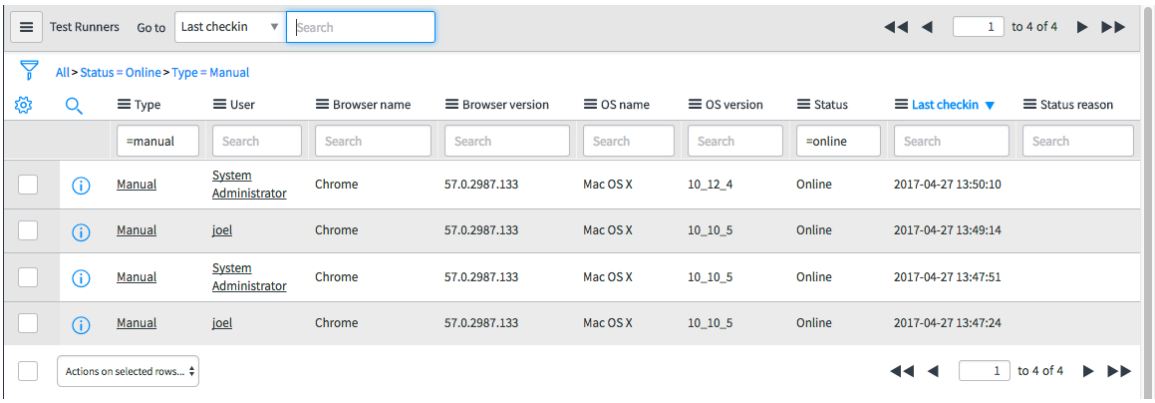
Active manual test runners

View the client test runners table filtered to show only those runners available to run manually-started tests.

When you start a manual client test runner, the system registers that runner in the Test Runners table as active.

The data in this table is transient. While the runner is active, it reports in to the system at a specified interval. If the runner does not report in at the expected time, the system marks the runner as inactive. After a period of time the system deletes the runner. You can modify these intervals on the Automated Test Framework properties page.

Example Active Manual Client Runner Test table



Fields on the Client Runner Test table

Field / UI Element	Description
Type	Whether this test runner is for manual or scheduled tests. In the Active Manual Test Runners module, Type is always **Manual**.
User	The user logged into the browser session.
Browser name	The browser name.
Browser version	The browser version.
OS name	The name of the operating system running the browser.
OS version	The version of the operating system running the browser.
Status	Whether this runner is currently online or offline. In the Active Manual Test Runners module, Status is always **Online**.
Last checkin	The time/date this runner most recently reported in to the system.
Status reason	If the Status is Offline, the reason why. In the Active Manual Test Runners module, Status reason is always empty.

Active scheduled test runners

View the client test runners table filtered to show only those runners available to run tests to be started by a schedule.

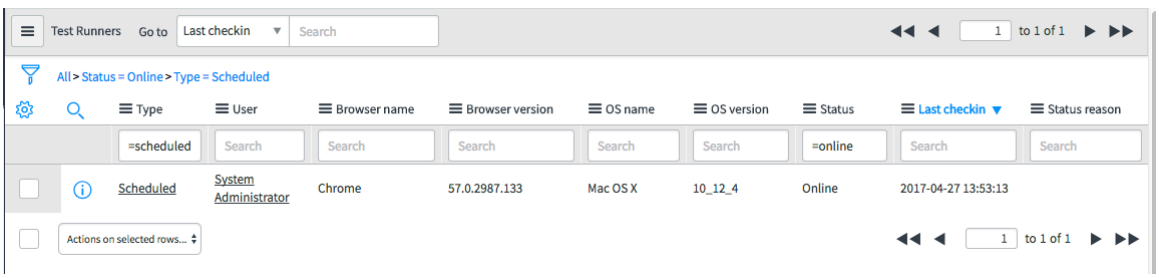
When you start a scheduled client test runner, the system registers that runner in the Active Scheduled Test Runners table.

The Active Scheduled Test Runner module is useful when you create a scheduled suite run. For scheduled suite runs, you can specify the browser to use. To determine the name and version of a browser you want to use, start a scheduled test runner with that browser, then inspect that runner's record in the Active Scheduled Test Runners module.

The data in this table is transient. While the runner is active, it reports in to the system at a specified interval. If the runner does not report in at the expected time, the system marks the runner as inactive. After a period of time the system deletes the runner.

You can modify these intervals on the Automated Test Framework properties page.

Example Scheduled Client Runner Test table



Fields on the Client Runner Test table

Field / UI Element	Description
Type	Whether this test runner is for manual or scheduled tests. In the Active Scheduled Test Runners module, Type is always **Scheduled**.
User	The user logged into the browser session.
Browser name	The browser name.
Browser version	The browser version.
OS name	The name of the operating system running the browser.
OS version	The version of the operating system running the browser.
Status	Whether this runner is currently online or offline. In the Active Manual Test Runners module, Status is always Online.
Last checkin	The time/date this runner most recently reported in to the system.
Status reason	If the Status is Offline, the reason why. In the Active Scheduled Test Runners module, Status reason is always empty.

Waiting/running test runs

The Waiting/Running Test Run module opens a list of records showing the tests waiting to be run.

Note:
To prevent conflicts, the system allows only one test to run at a given time. This is true even if you have multiple client test runner windows open. If you submit tests to run when another test is already running, the system holds the new tests in a queue to run later. If a test remains in the queue for more than ten minutes, the system cancels the test.

You can cancel execution of a waiting automated test.

Waiting/running suite runs

The Waiting/Running Suite Runs module opens a list of records showing the test suites waiting to be run.

Allowed client error records

Review the list of existing Allowed Client Error [sys_atf_whitelist] records to see which client errors produce warnings and which are ignored. Modify existing Allowed Client Error records as needed or create new ones.

Allowed Client Error form

Field	Description
Report level	Report action to take when the client error is encountered. Options include:

Allowed Client Error form (continued)

Field	Description
	• **Warning – Step & Test will report Success with Warning(s)**: Test steps containing the allowed client error report a status of `Success with warning(s)`. The error message appears in the test result output, and is recorded in the test logs with the status `Warning`. • **Ignored – Step & Test will report Success**: Test steps containing the allowed client error report a status of `Success`. The error is recorded in the test logs with an `Ignored` status.
Active	Check box to enable or disable allowing of a client error.
Error message	Message of the client error you want to allow.
Description	Description of the error you want to allow. If this client error was allowed from a test result, step result, or a test log, the test log description is copied into this field.

Related topics

Allowed client errors

Reported client errors

The Reported Client Error module lists test logs across all tests that are client errors and have failed. You can review individual test log records, and allowed client errors as needed to prevent them from adversely impacting future test executions.

Field	Description
Start time	Time at which this test step started executing.
Step	Name of the step executed.
Status	The result of the test log. Failure: Test step run failed.
Output	Output generated for the test log. For a client error, the actual client error is displayed.
Duration	Time duration it took to execute this test step.
Type	Type of test log. Client Error **Note:** You can optionally add client errors as an ignored or warning entry in the Allowed Client Errors. Doing so prevents the allowed client errors from affecting ATF test executions when they recur in future test runs. For more details, see Allowed client errors.
Recorded at	Time at which this step or log entry was recorded.

Related topics

Allowed client errors

Manually allow client errors

Identify and resolve client errors

Automated Test Framework roles

Automated Test Framework is installed with these roles.

ATF test administrator [atf_test_admin]

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

Create or edit Automated Test Framework properties. Has permission to:

- View the tests page
- Create/edit/delete tests
- Create/edit/delete test steps
- View the step config page
- View the test runner page
- View the test suite results, test results and result items pages
- Execute user tests
- View, create, edit, delete and execute test suites
- Create/edit step config records
- Create/edit Automated Test Framework properties

Contains Roles

List of roles contained within the role.

- impersonator
- personalize_dictionary

Groups

List of groups this role is assigned to by default.

None

Elevated

Whether the role is an elevated role. Elevated roles aren't assigned to users or groups, and must be used by elevation. For details, see Elevate to a privileged role.

No.

Special considerations

None.

ATF test designer [atf_test_designer]

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

View Automated Test Framework properties only (can't create or edit properties). Has permission to:

- View the tests page
- Create/edit/delete tests
- Create/edit/delete test steps
- View the step config page
- View the test runner page
- View the test suite results, test results and result items pages
- Execute user tests
- View, create, edit, delete and execute test suites
- View Automated Test Framework properties

Contains Roles

List of roles contained within the role.

impersonator

Groups

List of groups this role is assigned to by default.

None

Subscription

Whether the role is a chargeable user role that requires allocation of users with this role to subscriptions.

Elevated

Whether the role is an elevated role. Elevated roles aren’t assigned to users or groups, and must be used by elevation. For details, see Elevate to a privileged role.

No.

Special considerations

None.

ATF workspace designer [atf_ws_designer]

To learn more about managing per-user subscriptions, see Managing per-user subscriptions in Subscription Management and contact your account representative.

View or set the basic or mutual authentication needed for REST endpoints that require authentication.

Contains Roles

List of roles contained within the role.

- atf_test_designer
- web_service_admin

Groups

List of groups this role is assigned to by default.

None

Subscription

Whether the role is a chargeable user role that requires allocation of users with this role to subscriptions.

No.

Elevated

Whether the role is an elevated role. Elevated roles aren't assigned to users or groups, and must be used by elevation. For details, see Elevate to a privileged role.

No.

Special considerations

None.

Administration

The Administration module contains forms for configuring and managing the automated test framework.

Properties

On the Properties form, you can set parameters that control how the system executes automated tests and test suites.

Test/Test Suite Properties

Field / Element	Description
Enable test/test suite execution *sn_atf.runner.enabled*	If checked, enables running tests and test suites on this instance. This setting is unchecked by default to prevent users from unintentionally running tests on production instances.
Enable scheduled test suite execution sn_atf.schedule.enabled	If checked, enables scheduling test suites on this instance.

Test Debugging Properties

Field / Element	Description
Enable additional debugging functionality *sn_atf.debug*	Enables additional debugging functionality, including adding a debugging tab on the client Test Runner UI page and saving UI Test Result JSON to the test result record.

Screenshot Properties

Field / Element	Description
Enable or disable screenshot capture during test execution. *sn_atf.screenshots.mode* **Note:** For additional information, see Set the system property to control when the Automated Test Framework captures screenshots	• To capture screenshots for all steps, select **Enable for all steps**. • To capture screenshots only for failed steps, select **Enable for all failed steps**. • To capture no screenshots, select **Disable for all steps**.
Enable the GlideScreenshot feature *sn_atf.screenshots.use_glide_screenshot*	Improves the fidelity of screenshots on Workspaces and other interfaces. **Note:** IE and Safari don't support this property. If this property is disabled or your current browser doesn't support it, html2canvas is used to take screenshots. If you are upgrading to a new release version, this property is set to false by default.
Enable capturing the full page when taking a screenshot *sn_atf.screenshots.capture_full_page*	Enables dynamic resizing of the test frame and capturing the full page while taking a screenshot.

Screenshot Properties (continued)

Field / Element	Description
	Note: By default, the property is set to false because it can significantly slow test execution time If the property is disabled, only the portion of the page that is visible in the test frame is captured.
Screenshot timeout *sn_atf.atf_test_runner.screenshot_timeout*	Skips a screenshot capture attempt in the Client Test Runner if it exceeds this value in seconds. Users should review performance settings and browser caches on affected client systems before increasing this value.
Number of pixels for screenshot height *sn_atf.atf_test_runner.testframe.min_height*	Numeric value representing the number of pixels for the screenshot height. Default value: 600
Number of pixels for screenshot width *sn_atf.atf_test_runner.testframe.min_width*	Numeric value representing the number of pixels for the screenshot width. Default value: 800

Custom UI Page Data Capture Properties

Field / Element	Description
Enable tests with Custom UI steps to capture page data each time they are run. *sn_atf.page_data_capture.enabled*	When true, Custom UI test steps retrieve page data every time the test runs. Set this property to true when developing custom UI pages to always run tests on the most recent page version. When false, Custom UI test steps do not retrieve page data unless the test designer manually selects **Retrieve Components** during test design. Set this property to false when UI development is complete to enable faster test runs.

Test Runner Properties

Field / Element	Description
Test runner timeout *sn_atf.runner.heartbeat.timeout*	If there is no heartbeat from the test runner within this period of time in seconds, the status changes from online to offline. The value of this property should be between 120 and 1800. Default value: 120
Test runner heartbeat interval *sn_atf.runner.heartbeat.interval*	Time interval in seconds for sending a heartbeat from the test runner to the server.
Offline test runner retention interval *sn_atf.runner.offline_retention.timeout*	If an offline test runner does not communicate with the system for this period of days, the system deletes that test runner.

Test Suite Report Properties

Field / Element	Description
Test suite report properties *sn_atf.schedule.reports.suite.aging_threshold*	The number of test suite results to display in the test suite aging report.

Email Properties

Field / Element	Description
Boolean value for results displayed in scheduled suite result emails. *sn_atf.schedule.suite_result_email.only_show_failed_results*	When true, the scheduled suite test result emails only show results that failed. When false, displays all results.
Maximum number of test results to be displayed in scheduled suite result emails. *sn_atf.schedule.suite_result_email.max_test_history*	Maximum number of test results to be displayed in scheduled suite result emails.

Email Properties (continued)

Field / Element	Description
Maximum depth when printing suite results in suite result emails. *sn_atf.schedule.suite_result_email.max_depth*	Maximum depth when printing suite results in suite result emails.
Color to indicate an ATF test failed in scheduled suite result emails. *sn_atf.schedule.suite_result_email.fail_color*	Hexadecimal code for color to indicate an ATF test failed in scheduled suite result emails.
Color to indicate an ATF test errored in scheduled suite result emails. *sn_atf.schedule.suite_result_email.error_color*	Hexadecimal code for color to indicate an ATF test errored in scheduled suite result emails.
Color to indicate an ATF test passed in scheduled suite result emails. *sn_atf.schedule.suite_result_email.pass_color*	Hexadecimal code for color to indicate an ATF test passed in scheduled suite result emails.
Color to indicate an ATF test was skipped in scheduled suite result emails. *sn_atf.schedule.suite_result_email.skip_color*	Hexadecimal code for color to indicate an ATF test was skipped in scheduled suite result emails.
Color to indicate an ATF test was canceled in scheduled suite result emails. *sn_atf.schedule.suite_result_email.cancel_color*	Hexadecimal code for color to indicate an ATF test was canceled in scheduled

Email Properties (continued)

Field / Element	Description
	suite result emails.

Step configurations

Step configuration records define how each type of step behaves.

Automated Test Framework Step Config record

The step config record controls how a test step of this type behaves.

Fields

Field / Element	Description
Name	The name for this test step configuration.
Active	**True** if this step configuration is active. Otherwise, **false**.
Step Environment	The step environment in which a step with this configuration can run.
Category	The category assigned to a step with this configuration.
Application	The application scope in which the system runs steps with this configuration.
Batch order constraint	Constrains where a step with this configuration can appear in a test: • **None**: A step based on this configuration can appear at any point in a test. • **Start Batch Execution**: If a test includes this a step based on this configuration, it must be the first step in the test. • **Run in the middle of an execution**: If a test includes a step based on this configuration, it cannot be the first or last step. • **Stop Execution**: If a test includes a step based on this configuration, it must be the last step in the test.
Class type	For custom step configurations, this field is always **Script**.
Order	An integer specifying where steps with this configuration appear in the step list on the Add Test Step dialog. For more information, see the example using the Order field.
Template reminder	The instructions that appear when a step with this configuration is included in a test as part of a template. For more information, see theexample of using the Template reminder field.
HTML description	The text that appears when the cursor highlights a step with this configuration on the Create New Step dialog. For more information, see the example using the HTML description field.
Description generation script	Generates the text that describes a step when a test includes it. For more information, see the example using the description generation script.

Fields (continued)

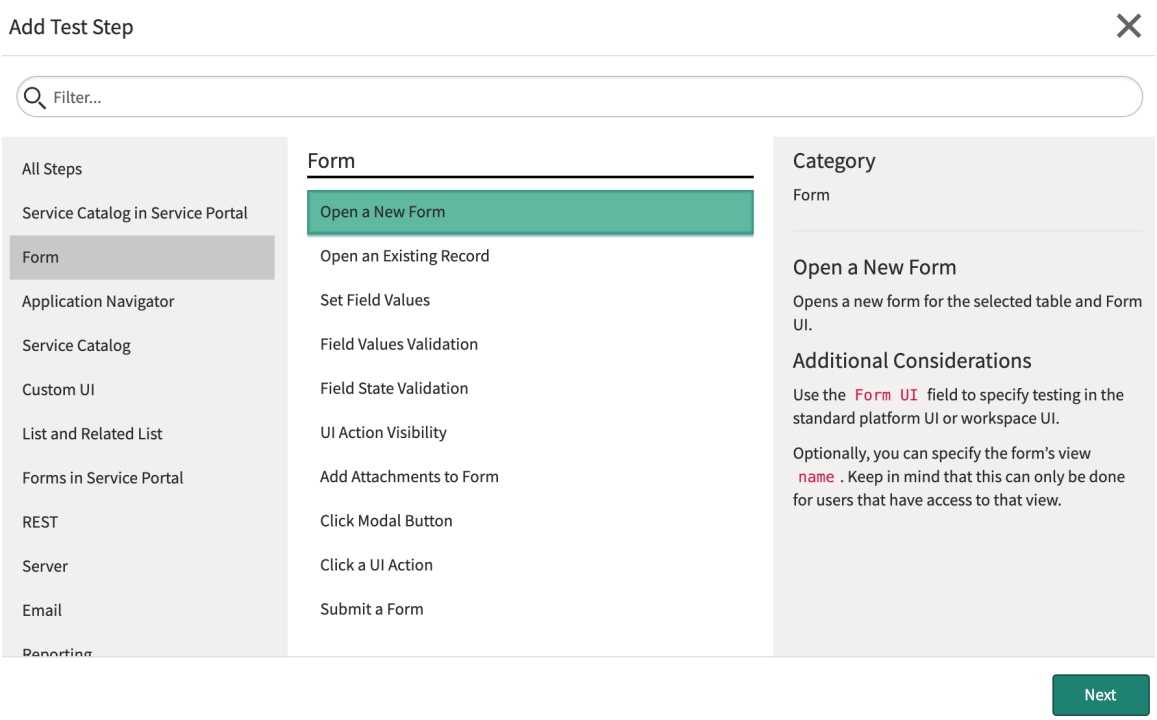
Field / Element	Description
Step execution script	Script (javascript) that runs when a step with this config runs.
Input Variables related list	The variables that act as inputs for a step with this config.
Output variables related list	The variables that act as outputs for a step with this config.

Examples of step config field values

Examples of where the system displays values assigned to some of the step config fields.

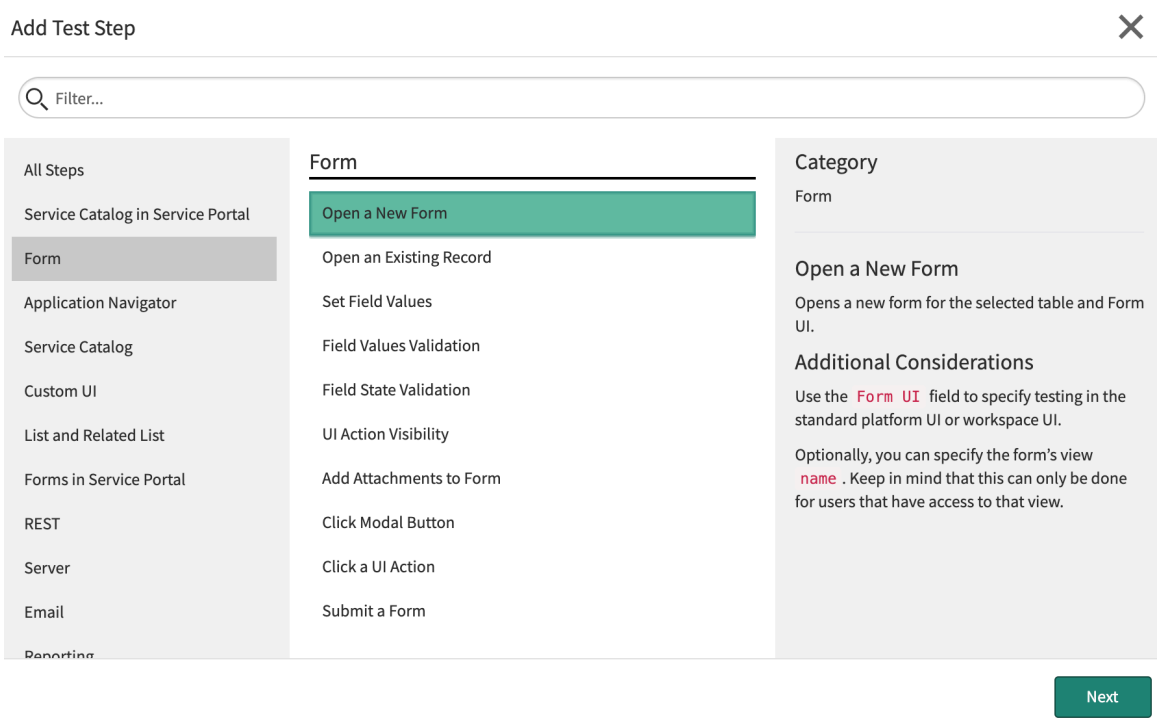
Order field example

The steps in the middle column are sorted according to the values of the **Order** field.



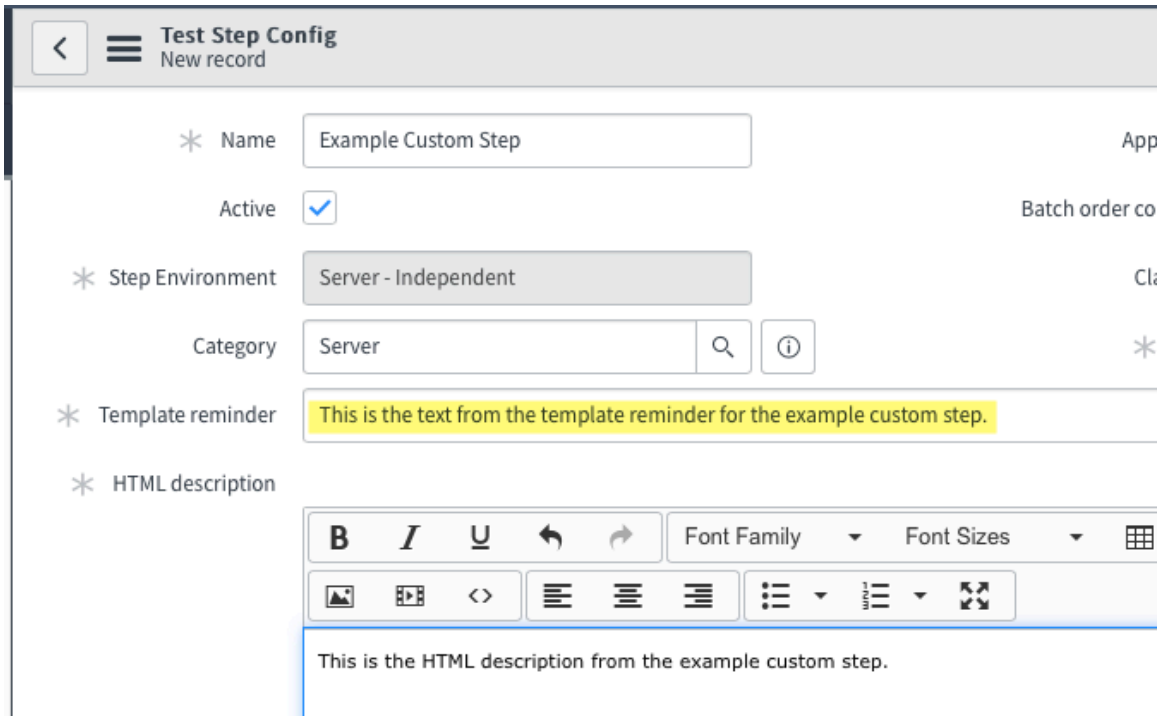
Category field example

The labels in the left column show the categories for available steps. You can filter the list in the middle column, by selecting a category in the left column.

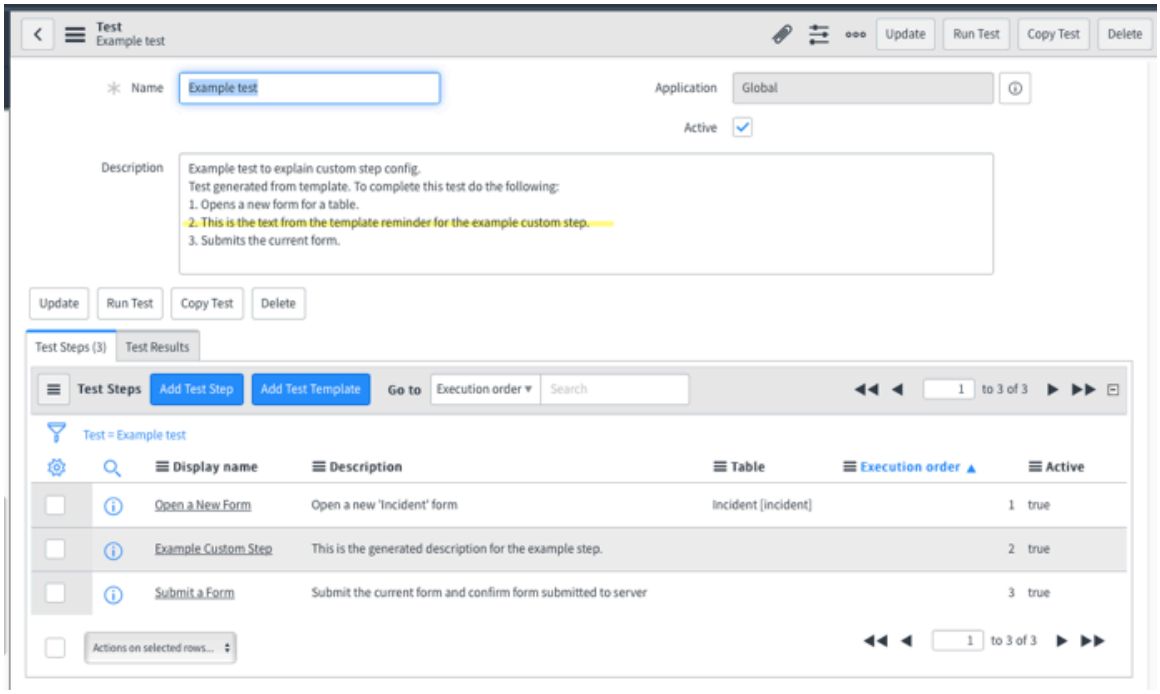


Template reminder example

Here is a portion of the new record form for a test step config for Example Custom Step.

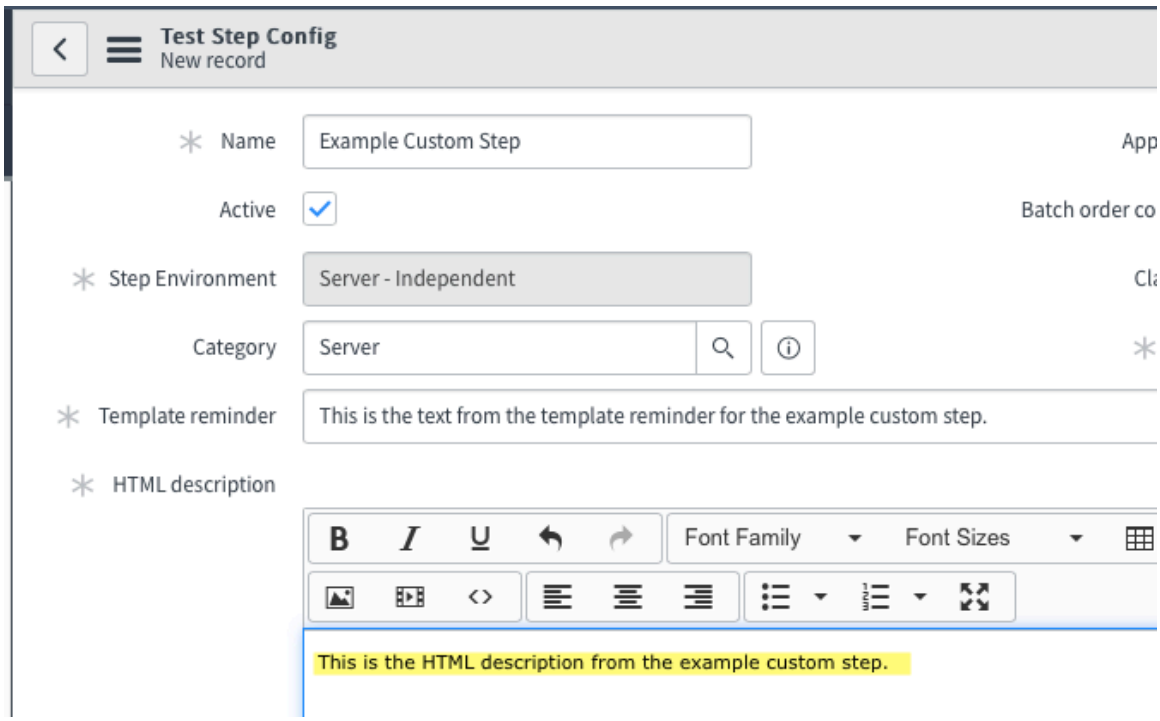


When you add a template to a test, the system generates a set of instructions for completing the template steps and saves them in the test **Description** field. The text in the step config's **Template reminder field** appears as the instructions for the corresponding step.

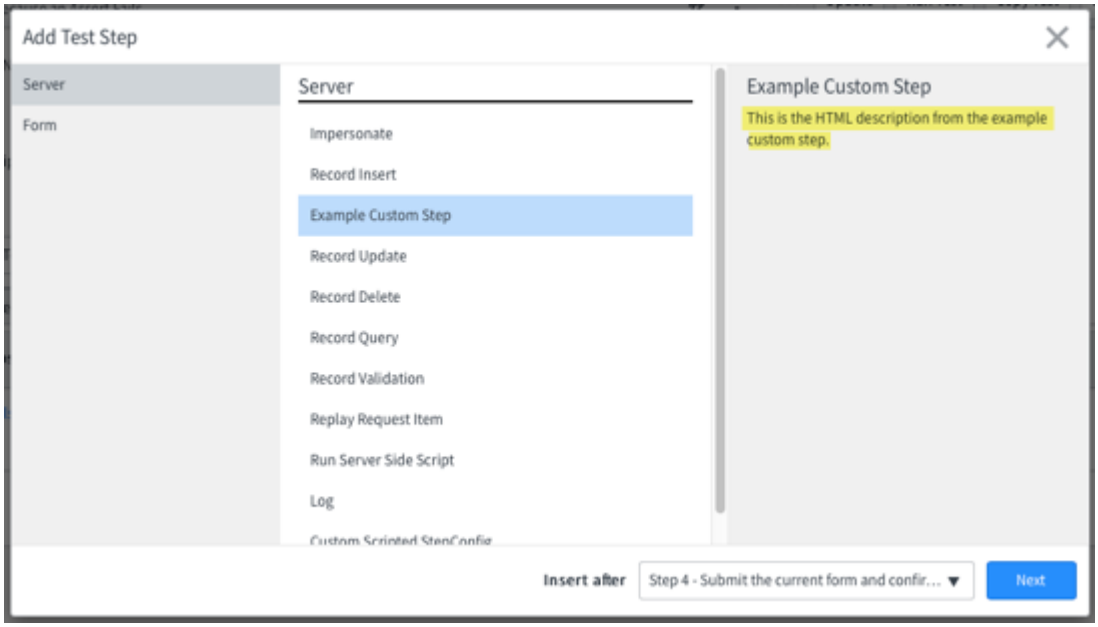


HTML description example

Here is a portion of the new record form for a custom step named Example Custom Step.

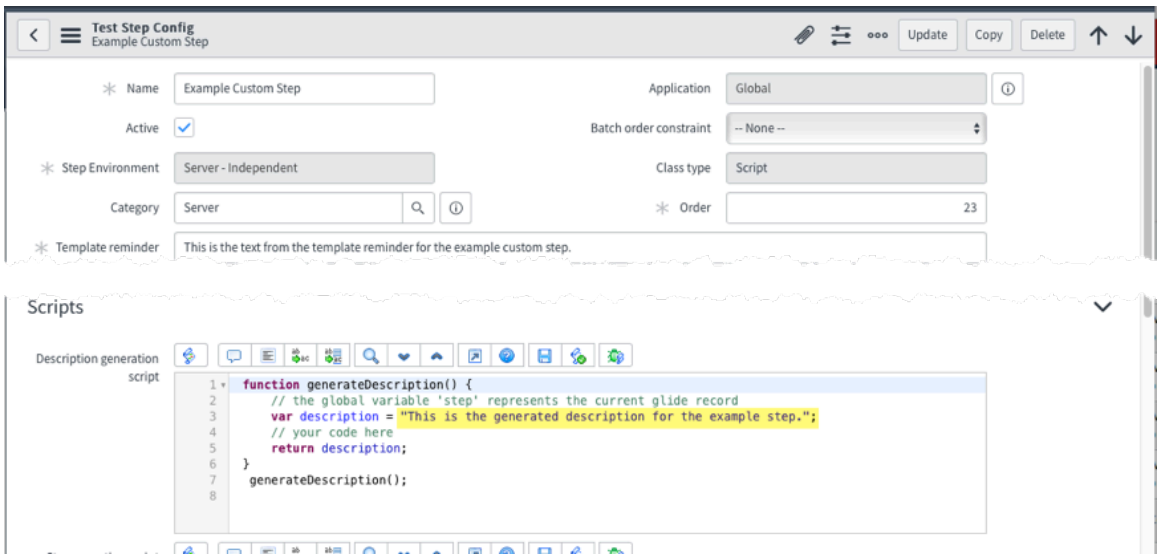


Here is how the step appears in the Add Test Step dialog.

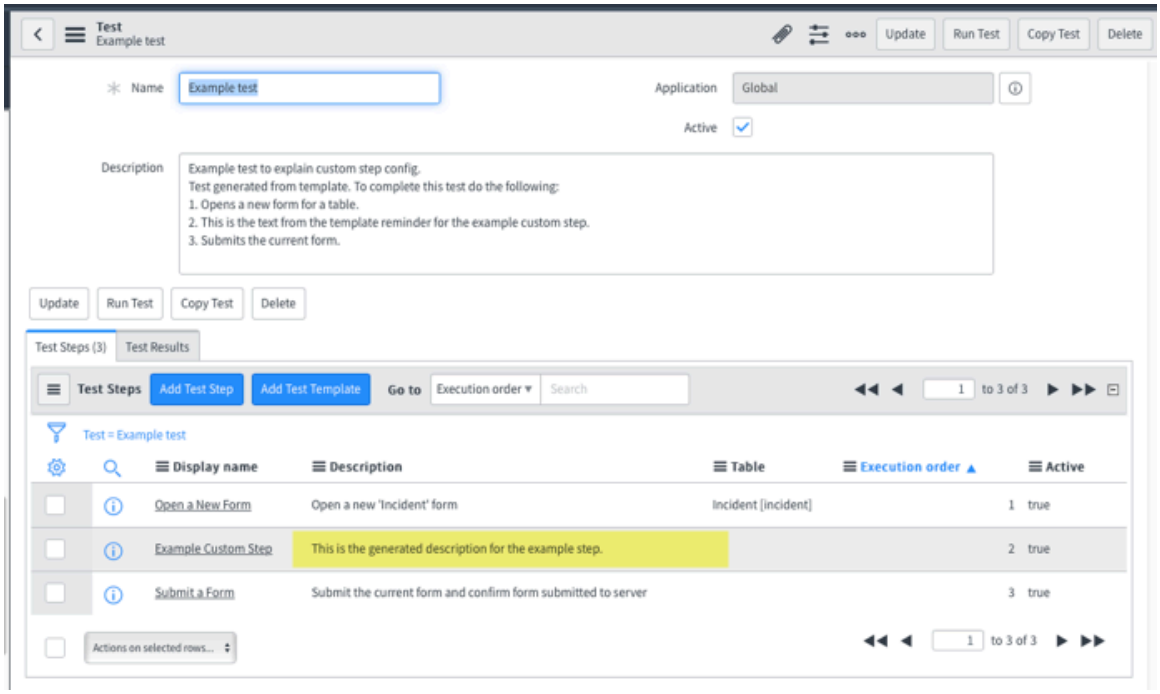


Description generation script example

Here is a portion of the new record form for a custom step named Example Custom Step.



When a step of this type is included in a test, the generated description appears in the Test Steps related list.



Step execution scripts

In a step configuration record, the step execution script field determines what a step with this configuration does when it runs.

Step inputs

The input variables to a step are determined by the inputs related list in the step configuration record. The *`inputs`* parameter to `executeScript()` gives the script access to these variables. For example, if the inputs related list contains two records, *`var1`* and *`var2`*, the script can reference *`var1`* with the expression `inputs.var1` and can reference *`var2`* with `inputs.var2`.

Step outputs

The output variables to a step are determined by the outputs related list in the step configuration record. The *`outputs`* parameter to `executeScript()` gives the script access to these variables. For example, if the outputs related list contains two records, *`out1`* and *`out2`*, the script can reference *`out1`* with the expression `outputs.out1` and can reference *`out2`* with `outputs.out2`.

Step result

The *`stepResult`* parameter provides access to an API that controls whether the step passes or fails. It also determines the message the step writes to the log.

The method `stepResult.setSuccess()` causes the step to succeed. The method `stepResult.setFailed()` causes the step to fail.

The method `stepResult.setOutputMessage()` sets the message to write to the log when the step succeeds or fails. It takes one parameter: the string to write to the log. If the script calls `stepResult.setOutputMessage()` more than once, the most recent value set overwrites any previous value.

Example: Record Query step execution script

```
   (function executeStep(inputs, outputs, stepResult) {
        if (gs.nil(inputs.table)) {
           stepResult.setOutputMessage(gs.getMessage("The '{0}'
input variable was not specified",
               'table'));
        stepResult.setFailed();
         return;
   }
   var query = new GlideRecord(inputs.table);
   query.addEncodedQuery(inputs.field_values);
   query.query();
   if (!query.next()) {
        stepResult.setOutputMessage(gs.getMessage("No records
matching query:\n{0}",
            inputs.field_values));
         stepResult.setFailed();
   } else {
        stepResult.setSuccess();
        outputs.table = inputs.table;
        outputs.first_record = query.getUniqueValue();
        stepResult.setOutputMessage(gs.getMessage("Found {0}
{1} records matching query:\n{2}",

[query.getRowCount(),
                                                inputs.table,

inputs.field_values]));
   }
   }(inputs, outputs, stepResult));
```

Example: Custom scripted step configs

```
   ((function executeStep(inputs, outputs, stepResult, timeout)
{
// Waits up to the timeout for some asynchronous logic to
finish
// This script checks for completion once a second for up to 60
seconds
var counter = 1;
// Try for up to 60 seconds
while (counter <= timeout) {
 // If the asynchronous logic is finished, return "true" to
pass the step
 // isMyAsyncLogicFinished() can be replaced with any
asynchronous event that needs to be tested
 if (isMyAsyncLogicFinished()) {
  stepResult.setOutputMessage("Success!");
  stepResult.setSuccess();
  return;
 }
 // Wait one second, and log the total number of seconds waited
```

```
 gs.info("Waited " + counter + " seconds for asynchronous logic
to finish");
 sn_atf.AutomatedTestingFramework.waitOneSecond();
 counter++;
}
// If this point is reached, the retry loop ran out of tries;
return false to fail the step
stepResult.setOutputMessage("FAILURE: Timed out after waiting
for " + timeout + " seconds");
stepResult.setFailed();
}(inputs, outputs, stepResult, timeout));
```

Note:
The above example can also be used for Run Server Side script by replacing `stepResult.setSuccess()"` and `stepResult.setFailed()` with `return true` and `return false`.

Step description generation script

In a step configuration record, the step description generation script field determines the step description that the system generates when a step of this type is added to a test.

For an example showing how where the description generated by this script appears, see Description generation script example.

Step

The *step* parameter to `generateDescription()` gives the script access to the step object, which it turn gives access to the input variables as defined in the step configuration record. (Input variables are defined in a related list.)

For example, if – in a step config record – the inputs related list contains two records: *var1* and *var2*, the script can reference *var1* with the expression `step.inputs.var1` and can reference *var2* with `step.inputs.var2`.

Example: Record Query description generation script

```
   (function generateDescription(step) {
   var td = GlideTableDescriptor.get(step.inputs.table);
   if (!td) {
        gs.log("Invalid table name in test step: " +
step.inputs.table);
        return gs.getMessage("Set field values");
   }
   var descriptionGenerator = new
ATFStepDescriptionGenerator();
   var description = gs.getMessage("There should be at least
one record in '{0}' matching " +
                   "a query of\n{1}",
                   [step.inputs.table.getDisplayValue(),

descriptionGenerator.getConditionDescription(step.inputs.table,
step.inputs.field_values)]);

   return description;
```

```
})(step);
```

Add output variables to scripted steps

Execute the following steps to add additional outputs in Run Server Side Script and Custom Scripted StepConfig test steps.

Adding outputs in Run Server Side Script test step

Modify the test scripts of Run Server Side Script test step to create additional outputs of your choice.

Before you begin

Role required: admin or atf_test_admin

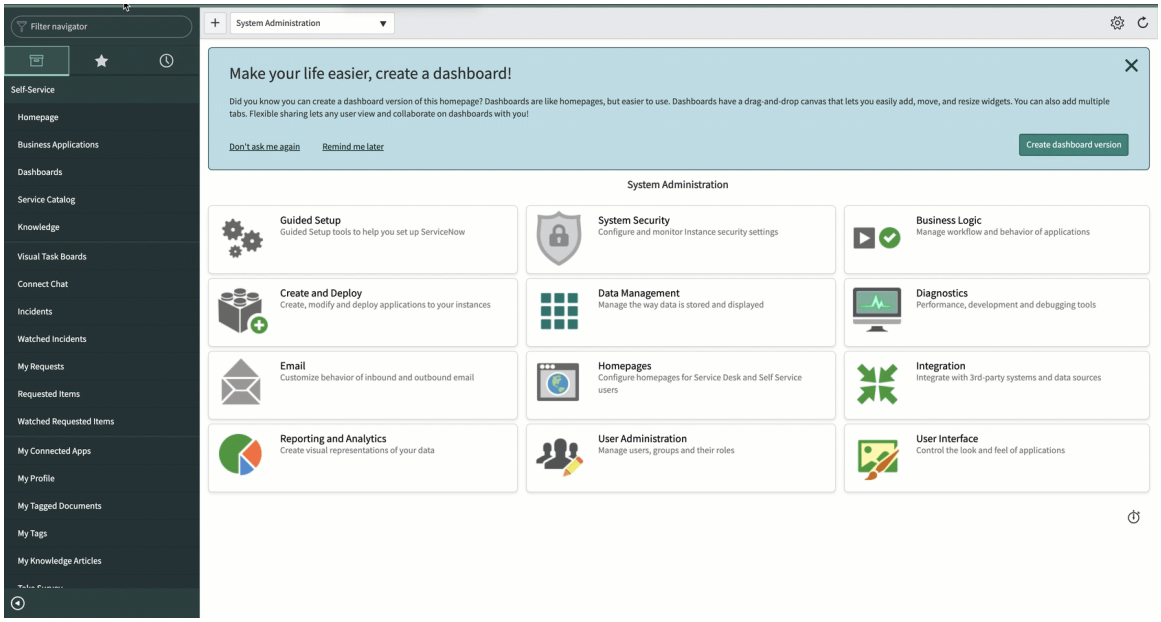
Procedure

1. Navigate to **All** > **Automated Test Framework (ATF)** > **Administration** > **Step Configurations**.
2. Search and select Run Server Side Script.
   The read-only Test Step Config form shows up.

   Note:
   Although the form is read-only, new output variables can be created.

3. Scroll down to the Output Variables related list.
   The list of default output variables shows up.
4. Click **New** to create a new output variable.

Field	Input Value
Type	Type of output variable
Application	Scope of the user
Label	User-facing name of the output variable
Column name	Variable name used in the script
Max length	Length of the data type. **Note:** This field doesn't show up for all data types.



The new output variable gets added to the Output Variable related list.

5. Navigate to **Automated Test Framework (ATF)** > **Tests**.
6. Select the test where you want to implement the new output variable.
7. Click **Add Test Step**.
8. Search and select Run Server Side Script test step.
9. Follow the instructions to change outputs mentioned in the **Test script**.

```
outputs.<column_name> = "<desired value>";
```

10. Click **Submit**.
    The newly created output variables are now ready to be used in any subsequent steps of the test.

Adding outputs in Custom Scripted StepConfig test step

Copy the Custom Scripted StepConfig test step and customize the copied version by adding additional outputs.

Before you begin

Role required: admin or atf_test_admin

Procedure

1. Navigate to **All** > **Automated Test Framework (ATF)** > **Administration** > **Step Configurations**.
2. Search and select Custom Scripted StepConfig.
   The read-only Test Step Config form shows up.

> **Note:**
> Since the form is read-only, you must copy the test step to customize it to add more output variables.

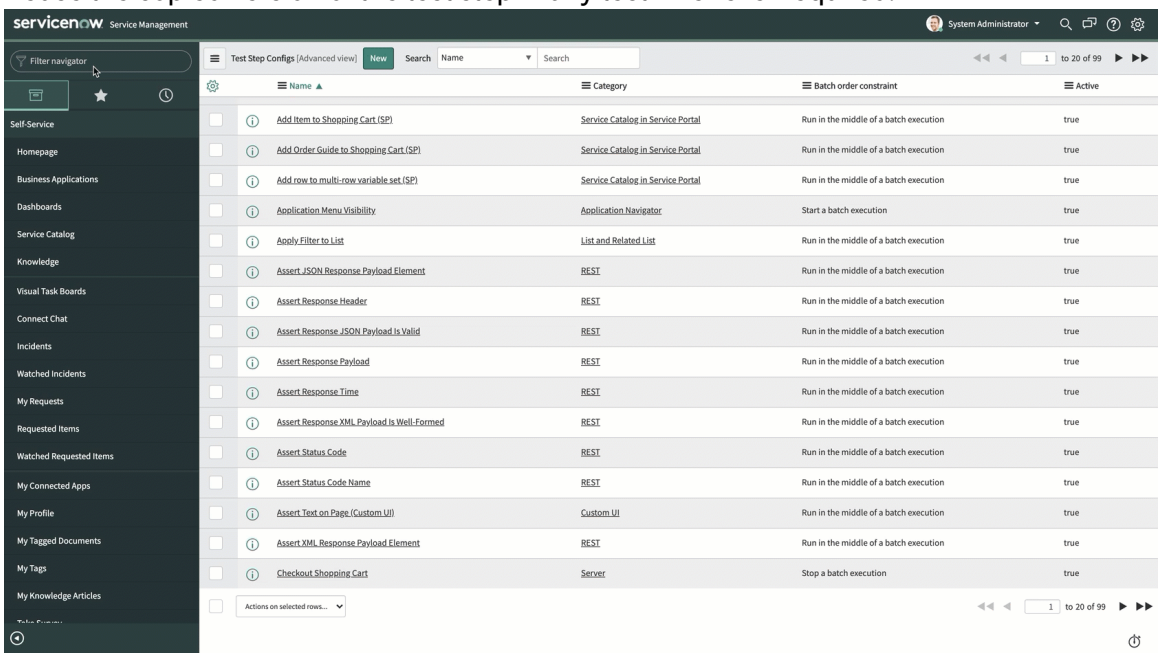
3. Select **Copy** to have a copied version of the test step.

> **Note:**
> The copied version is no longer read-only.

4. Add additional output variables in the copied version by implementing the following steps:
   a. Select **New** under Output Variables related list.
   b. Modify **Step execution script** to add more output variables.
   See Adding outputs in Run Server Side Script test step to add more output variables.

> **Note:**
> You can use these steps to customize the test step only in the copied version.

5. Reuse the copied version of the test step in any test whenever required.



Test templates

The Test Templates module opens a list of available templates. From this module, you create, view, and edit test templates.

To view or edit a test template, click the row for the template you want. For more information, see Edit automated test steps template.

For information how to add a new template, see Create an automated test steps template. For information on how to use a template when creating a test, see Add a predefined list of steps (template) to an automated test.

Automated Test Template record

The Test Template record contains information about one test template.

Fields

Field / Element	Description
Name	The name of this test template.
Application	Application scope in which the system runs this test or test suite.
Test Template	The tests to include in this template.
Description	(Optional) Enter a description to identify the purpose of this test template.

Step configuration categories

The Step Configuration Categories module opens a list of records specifying the step categories on the Add Step dialog. From this module, you can add, delete, and edit these categories.

Categories are used for filtering the step list in the Add Step dialog. For more information, see Category field example.

Test step config category form

On the Test Step Config Category form, you specify a retention policy for a set of records on a given test results table.

Fields

Field / Element	Description
Name	The name for this step config category.
Step Environment	The step environment for this step config category.
Display name	The category name that appears in the middle column of the Add Test Step dialog when this category is selected. See an example in Step Config Category Display name.
Application scope	The application scope to which this policy applies.

Table cleanup

The Table Cleanup module opens a list of records specifying the retention policies for test result and test suite result tables and the records within them. From this module, you can view and modify these policies.

By default, the system deletes system records related to test results and test suite results 30 days after creation. To modify the retention policies for a table and its records, click the table for which you want to modify policies. For more information, see Modify data retention policy for ATF test results.

Note:
Table cleanup policies are platform-specific policies. See Table cleaner for more information.

Autoflush form

On the Auto Flush form, you specify a retention policy for a set of records on a given test results table.

Fields

Field / Element	Description
Tablename	The table containing the records to which the policy applies.
Matchfield	The field for which the system monitors duration.
Age in seconds	The amount of time (in seconds) the system wait before deleting the records.

Fields (continued)

Field / Element	Description
Active	**True** if this policy is active. Otherwise, **false**.
Application scope	The application scope to which this policy applies.
Cascade delete	If checked, the system deletes all matching records plus any records referring to them. If unchecked, the system deletes matching records, but not records referring to them.
Conditions	The filter conditions defining the records in this table to which the policy applies.

Note:
Table cleanup policies are platform-specific policies. See Table cleaner for more information.

Step environments

A test step environment specifies where the step executes (for example, server versus browser). In this release, custom step configs can use only the Server-Independent environment.

In this release, you cannot add custom test step environments.

Deploying applications

Deliver your application to users by publishing it to a live production environment.

Deploying applications on the Next Experience

Managing application deployments using Pipelines and Deployments

Use Pipelines and Deployments to review and publish applications built in App Engine Studio.

Manage deployments using AEMC

Use App Engine Management Center (AEMC) dashboard to view multiple pipelines and manage applications at all stages of deployment.

Managing app deployments using Pipelines and Deployments

Use Pipelines and Deployments to review and publish applications built in App Engine Studio.

Managing deployments using pipelines in AEMC

Use App Engine Management Center (AEMC) dashboard to view multiple pipelines and manage applications at all stages of deployment.

Deploying applications in the Core UI

ServiceNow application repository

The ServiceNow Application Repository is a central location for all scoped applications that are published by all ServiceNow customers. After you develop and test a custom application, you can make the application available to company instances by publishing it to this repository.

System update sets

An update set is a group of configuration changes that can be moved from one instance to another. This feature enables administrators to group a series of changes into a named set and then move them as a unit to other systems for testing or deployment.

ServiceNow application repository

The ServiceNow Application Repository is a central location for all scoped applications that are published by all ServiceNow customers. After you develop and test a custom application, you can make the application available to company instances by publishing it to this repository.

System update sets

An update set is a group of configuration changes that can be moved from one instance to another. This feature enables administrators to group a series of changes into a named set and then move them as a unit to other systems for testing or deployment.

System update sets

An update set is a group of configuration changes that can be moved from one instance to another. This feature allows administrators to group a series of changes into a named set and then move them as a unit to other systems for testing or deployment.

Note:
When an update set is backed out, there is an OOB business rule that will delete sys_upgrade_state record on customer update deletion. This is an expected behavior.

An update set is an XML file that contains:

- A set of record details that uniquely identify the update set.
- A list of configuration changes.
- A state that determines whether another instance can retrieve and apply configuration changes.

Update sets track changes to applications and system platform features. This allows developers to create new functionality on a non-production instance and promote the changes to another instance.

Warning:
Update sets allow moving changes between instances that may be running different family release versions and different features. You can always load an update set created on an older family release on an instance running a newer family release. Loading an update set created on a newer family release on an instance running an older family release requires additional testing to determine compatibility. Updates from newer family releases may not produce the same functionality when moved to older family releases. In extreme cases, newer family release updates may cause outages or data loss on an older family release instance. Where possible, avoid moving updates from newer family releases to older family releases. Similar constraints apply to moving updates between instances running different versions of ServiceNow Store apps.

System properties

Administrators can exclude system properties from update sets by making them private. Keeping system properties private prevents settings in one instance from overwriting values in another instance. For example, you may not want a system property in a production instance to use a particular value from a development instance. See Add a system property.

Applications

Application developers have additional options with update sets such as:

- Create an update set for a specific version of an application.
- Specify which application tables to track in update sets.

Update set tables

Each update set is stored in the Update Set `[sys_update_set]` table, and the customizations that are associated with the update set, which are entries in the Customer Update `[sys_update_xml]` table, appear as a related list on the update set record.

When a tracked object is customized, a corresponding record is added or updated in the Customer Update `[sys_update_xml]` table and is associated with the user current update set. The associated application file properties are tracked and transferred along with the customized object in a single update record. A corresponding record is also added to the Versions `[sys_update_version]` table.

The Customer Update table contains one record per customized object, per update set. The Versions table contains one record per change to a customized object.

- Administrators can compare two versions and revert to a specific version of an object.
- Administrators can suppress versions for specific tables.
- Administrators can specify fields on tracked tables that you can change without skipping updates to the rest of the record (exclude the field from the update).

Note:
Do not directly modify Customer Updates `[sys_update_xml]` records.

Customizations tracked by update sets

Update sets can track customizations to application tables, fields, and records.

Update sets track customizations under these conditions:

- Where the table has an *update_synch* dictionary attribute.
- Where there is a special handler to track changes to multiple tables.
- Where the administrator has not excluded a field from updates.

In general, update sets capture configuration information but not task or process data. For example, update sets track service catalog item definitions and related configuration data like variables and variable choices. However, if you test the service catalog by placing orders, update sets do not track order requests, items, and catalog tasks.

Update sets have a limited capacity to transfer data as application files. For larger data transfers, export data and import it with an import set or web service.

Do not combine the usage of both Update Sets and the Application Repository for scoped app development. This will result in numerous issues, including skipped changes, commit errors, and more. Once you have installed an application from the Application Repository, you must continue to develop and publish to the Application Repository for all future development. If you decide to develop an application using update sets, you must continue to use that method exclusively.

update_synch attribute

To see the list of tables where customizations are tracked, navigate to **System Definition** > **Dictionary** and filter on *attributes CONTAINS update_synch*.

> **⚠ Warning:**
> Do not add the `update_synch` attribute to a dictionary record. When improperly used, this attribute can cause major performance issues or cause the instance to become unavailable. Because of this, the `update_synch` attribute is not accessible to customers.

A default rule blocks the use of the `update_synch` attribute on a table for which it is not predefined to avoid the following issues:

- Some core tables require special update handling because they represent information on multiple tables. When the `update_synch` attribute is added to these tables, duplicate update records are created, causing major conflicts that are difficult to troubleshoot and repair.
- Using the update_synch attribute to migrate data records between instances can cause performance issues, because it is not intended for this purpose. To migrate data, use an instance-to-instance import.

  See Import sets.

Special handlers

Some changes require special handlers because they represent information on multiple tables. These changes are packaged into one update set entry so that all records are properly updated when the customization is committed. The following changes are tracked with special handlers:

- Workflows
- Form sections
- Lists
- Related lists
- Choice lists
- System dictionary entries
- Field labels

Warning:
The form sections, lists, related lists, choice lists, and field labels special handlers delete and reinsert records. This might cause unexpected results and data loss if there are fields referencing the tables.

Choice lists

Update sets store both new and updated choice options as separate records in the Update Version `[sys_update_version]` and Customer Update `[sys_update_xml]` tables. For example, you create a new Activity `[u_activity]` table that extends the Task table. You then add a new choice option to the Task state field that is only visible for your extended table (for example, *My State*).

When you publish these changes as an update set, the update only contains update and version records for the choice you added to the u_activity table. The choice options in the task table are unaffected.

Warning:
Do not use large choice lists in update sets. Doing so leads to excessively long update set commits.

Dictionary changes

Usually, using update sets prevent you from applying dictionary changes that result in data loss. Blocked dictionary changes include:

- Removing tables
- Changing a column data type

Update sets do not track the removal of tables from the system dictionary. Instead, customers must manually remove tables from the target instance. While update sets track data type changes, the target instance skips any change that results in data loss and instead adds a log message about the action. Customers can use the log to manually make data type changes on the target instance.

Note:
Update set previews do not check for type mismatch problems since the target instance skips changes resulting in data loss. Also, using update sets to delete a column from a table can cause data loss in certain circumstances. If there is data in the column on the target instance, that data is deleted, as well as the column itself, when the update set is committed. A warning message appears if you attempt to commit an update set that deletes a column. The message states that there are one or more delete updates that cause the data to be deleted, and it specifies what delete updates there are.

Homepages and content pages

Homepages and content pages are not added to update sets by default. Add pages to the current update set by unloading them.

Important:

The functionality found in homepages, arranging information from your instance to tell a story about your data, is found in dashboards on new instances. On upgraded instances with Next Experience enabled, users can view existing homepages if they have a direct URL, but they can't create or edit them. Responsive dashboards and Analytics Center dashboards take over homepage functionality.

Use the Homepage deprecation help tool to convert the homepages on your instance to responsive dashboards.

For more information, see:

- Dashboards in the Analytics Center.
- Working with responsive dashboards.

Application changes

The system creates a separate update set for each application that only contains changes associated with the application. This ensures that access settings for each application are properly evaluated and applied when committing update set changes.

Default update set

Only one update set can be the default set for any application scope.

To set an update set to be the default set, you set the **Default set** field to true. When you set *Default set = true*, the following actions occur:

- The update set becomes the default update set for its scope.
- The system sets *Default set = false* for all other update sets with the same scope. This ensures that there is only one default update set for each scope.

Global default set

Use the global default update set to make changes to an instance without adding the changes to any user-created update sets. The global default update set is the set where *Default set = true* and application scope is *global*. The global default set (regardless of the *Name* of the set) provides system functionality and should not be changed, deleted, or moved between systems. Use this update set to make changes to an instance without adding the changes to any user-created update sets.

Auto-generated default set

At all times, to ensure that no updates to an instance are lost, the system ensures that there is a default set for the user's current scope. If the system finds that a default update set does not exist (or is marked **Ignored** or **Completed**) for the current scope, then the system auto-generates an update set and sets **Default set = true**.

These are some common cases where the system auto-generates a default update set.

- The very first time that an admin logs in, the system sets the system's global default update set as the administrator's update set. In addition, the application picker sets the administrator's application scope to global.

  If a global default update set does not exist (or is marked **Ignored** or **Completed**), the system creates a new update set for the global application scope and performs the following actions:

- The system sets *Default set* = *true* for the new set.
- The system sets the name of the new set to start with the name of the former default set and appends the next numeral (in the sequence SetName, SetName 1, SetName 2, …, SetName n).
- The system sets the newly created set as the administrator's update set.

- When a user marks the default set for a scope as **Ignored** or **Completed** (not a recommended practice), the system immediately auto-generates a new default set for the scope.
- The system auto-generates a new default update set for a scope when all the following conditions occur:
  - You change application scope.
  - Your preferred update set is **Complete** or **Ignored**.
  - There is no In-Progress default update set for the new scope.

Get started with update sets

Because update sets make changes to an instance, review this information to avoid errors and performance issues. Learn how to plan the update process and avoid common mistakes.

When to use update sets

Deployment option	Good for	Future considerations
Update Sets	Storing changes to a base system or installed application. Storing and applying a particular version of an application. Producing a file for export.	You can manually create update sets to store a particular application version. Use update sets to deploy patches or changes to installed applications. **Note:** Don't use update sets to install applications. Instead, use the application repository or the ServiceNow Store to install applications.
Application Repository	Installing and updating applications on all company instances. Automatically managing application update sets. Restricting access to applications to the same company. Deploying completed applications to end users.	Consider uploading an application to the ServiceNow Store to share it with other users. Allows installation of and update to the latest application version only. Use update sets to store prior application versions. **Note:** If used with team development, publish applications only from a parent instance.

Plan the update process

Before working with update sets, create a standard process for moving customizations from instance to instance using this check list:

1. Check that both instances are on the same version. Customizations may not work if they rely on code that has changed between versions.
2. Determine the changes to make in a single update set. Complete your update sets as you finish small to medium-sized tasks. As update sets get larger, it becomes harder to review them, takes longer to identify changes within them, increases the risk of conflicts with other update sets, and takes more time to preview and commit them. This is especially true if the update sets contain schema changes or revisions to large workflows or if the set has to be backed out.
3. Confirm that all base system records have matching sys_id fields. Some base system records are created on an instance after provisioning and don't match between different instances, leading to problems with update sets. The best way to avoid this issue is to:
   - Provision production and non-production instances.
   - Clone the production instance onto the non-production instance.
4. Identify a common path for update sets to move from instance to instance and maintain that model. Never migrate the same update set from multiple sources. Move update sets from dev to test and then from test to production.
5. Plan for when to commit the update sets to production. Avoid committing an update set to a production instance during business hours. The instance may perform more slowly while the update set applies. Rest assured, this slower performance is temporary.
6. Make sure update set names are clear. Create a naming convention to coordinate changes from multiple developers and to reference when committing the changes to another instance.
   - If update sets are being generated as fixes for problems, consider including the problem ticket in the name (for example, **PR10005 - Duplicate Email Issues Fix**).
   - If you need more than one update set to address a problem, include a sequence number in the naming convention. Doing so helps ensure that update sets are applied in the order that they were created (for example, **PR10005 - Duplicate Email Issues Fix** and **PR10005.2 - Duplicate Email Issues Fix**).
7. Understand the following about update sets:
   - What records are generated.
   - Which customizations are tracked.
   - Which dictionary changes are valid.
   - Which customizations can be backed out (reversed) after applied.
8. Before making any customizations, double-check that the correct update set is selected.

Working with update sets

Review this information to avoid errors and performance issues.

- Don't delete update sets. If an update set is deleted, any updated records may be overwritten in the next update.
- Don't include the **system_id** field from the ldap_server_config record in an update set. An update set from a working configuration points to the wrong system_id node for the target instance and doesn't work.
- Don't back out the Default update set. This action damages the system.
- Never change the **Update Set** field value (update_set) in a Customer Update record (sys_update_xml). If a customization is made in the wrong update set, take the following action:

1. Switch to the desired update set.
2. Modify the object (record) that was originally changed. You can make a trivial change, such as adding a field.
3. Save the record.
4. Back out the change just performed, and then save the record again.

   This action ensures that the latest version of the object is included in the desired update set and prevents duplicate updates for the same object in a single update set.

- Don't mark an update set as **Complete** until it's ready to migrate. After an update set is complete, don't change it back to **In progress**. Instead, create another update set for the rest of the changes, and make sure to commit them together in the order that they were created. Naming conventions may help in this case (for example, Performance Enhancements and Performance Enhancements 2).
- Don't manually merge updates into an update set. Use the Merge Update Sets module. This tool compares duplicate files between update sets and selects the newest version.
- If a committed update set has a problem in the test instance, build the fix in another update set in the development instance. Commit this set to the test instance, and then make sure both sets are migrated to the production instance and committed in the order they were made.
- Always preview an update set before committing it.
- Set completed update set on the production instance to **Ignore**. This state ensures the update set isn't reapplied when cloning the instance.
- Keep a to-do list of manual changes and data loads that must be completed after an update set is applied.
- Don't make too many changes at one time. Verify that the correct changes have been made incrementally.
- You can't change a single update to update across multiple domains (that is, global and TOP domains). This function isn't supported in the ServiceNow AI Platform.

To create an update set see Create and select an update set as the current set.

Update set administration

Manage how update sets store, retrieve, preview, and apply configuration changes between instances.

Administrators have the following options with update sets.

- Create an update set to store local changes.
- Select the current update set to store local changes.
- Commit an update set to prepare it for distribution.
- Report on the contents of update sets.
- Compare update sets to determine what differences they contains.
- Merge separate update sets into a single update set.
- Create an external file from an update set.
- Retrieve update sets from remote instances.
- Apply retrieved update sets.

- Back out changes applied from an update set.
- Set system properties related to update sets

Grant access to the update set picker

The update set picker allows users to choose an update set for making and tracking customizations. By default, only administrators can use the update set picker. You can grant access to additional users.

Automatically preview retrieved update sets

By default, the system automatically previews update sets as soon as it has retrieved them. To turn off this behavior, set the system property **glide.update_set.auto_preview** to **false**: in the navigator filter, type **sys_properties.list** then navigate to the **glide.update_set.auto_preview** property and edit the **value** field.

Track an application table

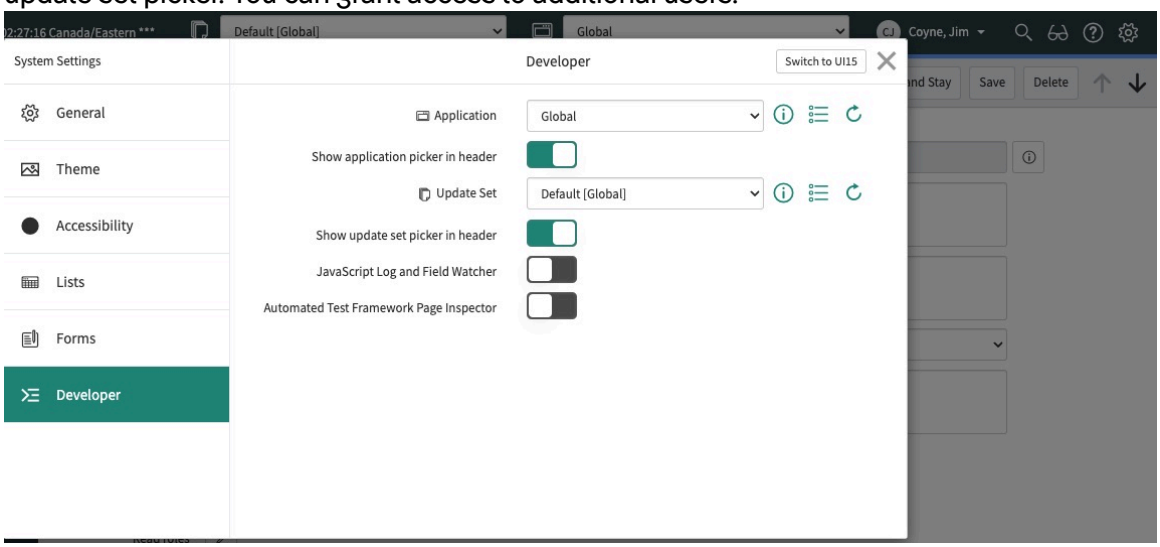
Application developers can track application changes in an update set to save or distribute a particular version of an application. During table creation, set **Extends Table** to Application File [sys_metadata].

Grant access to the update set picker

Enable a non-administrative user to use the update set picker.

Before you begin
Role required: admin

About this task
The update set picker appears on the Settings panel. The picker allows users to choose an update set for making and tracking customizations. By default, only administrators can use the update set picker. You can grant access to additional users.



Procedure

1. Grant the user role read access to the Update Set table [sys_update_set].
2. Enable users to see the update set picker on the Settings panel.

a. Add a system property *glide.ui.update_set_picker.role* to the System Properties table.

b. Set the value of *glide.ui.update_set_picker.role* to the role for which you want to give access.

Overwrite customizations during an upgrade

Specify which customized objects you want to replace during the next upgrade. By default, the upgrade process skips changes to customized objects.

About this task

The systems tracks the configuration changes you make such as modifying the dictionary record for a table or field. Each configuration change you make has a corresponding record in the Customer Update `[sys_update_xml]` table where the **Replace on upgrade** field is set to false. You may want to overwrite your customizations with the next software version. For example, you may change a script to implement a temporary workaround for a problem that is fixed in the next version. You would want to overwrite your workaround when upgrading to the next version to ensure that you receive any future enhancements to the script.

Procedure

1. Open the customized object (for example, the *ArrayUtil* script include).
2. Right-click the header and select **Show Latest Update**.
3. Configure the form to add the **Replace on upgrade** field, if necessary.
4. Select the **Replace on upgrade** check box and click **Update**.
   The customized object will be replaced on the next upgrade.

Creating, testing, and moving customizations

Use these procedures to create, test, and move customizations from a development system to a production system.

Three-step import process

A common process for developing customizations with update sets involves moving changes from development, to test, to production instances.

1. Create an update set on the development instance.
2. Make customizations and changes on the development instance.
3. Mark the update set as Complete.
4. Log in to the test instance and retrieve the completed update set from the development instance.
5. Commit the update set on the test instance, and test customizations thoroughly.
6. If the update set has problems in the test instance, repeat steps 1 - 5 to develop the fix on the development instance with another update set.
7. Identify a common path for update sets to move from instance to instance and maintain that model. Never migrate the same update set from multiple sources. Move update sets from dev to test and then from test to production.
8. Commit the update set on production. If the update set required a fix, commit both update sets in the order they were made.

Two-step import process

If your development environment consists of only two instances, you can combine your development and testing instances into a single staging instance.

1. Create an update set on the staging instance.
2. Make customizations and changes on the staging instance.
3. Mark the update set as Complete.
4. Test customizations thoroughly on the staging instance.
5. If the update set has problems, repeat steps 1 - 4 to develop the fix on the staging instance with another update set.
6. Log in to the production instance and retrieve the completed update set from the staging instance. If the update set required a fix, retrieve both update sets.
7. Commit the update set on production. If the update set required a fix, commit both update sets in the order they were made.

Update set use

These procedures help you manage your customizations and resolve potential collisions before you move them to another instance.

Before using update sets, review Get started with update sets to learn when to use and when not to use update sets, and how to plan the update process and avoid common mistakes. Then, create an update set and use it to make changes on a development instance. You can report on updates, merge update sets, and compare update sets to ensure the desired changes are ready to move.

When the update set is completed, you can transfer the update set to another instance according to your test process. See Update set transfers for details.

Create and select an update set as the current set

Create an update set to store customization changes and select it as the current set.

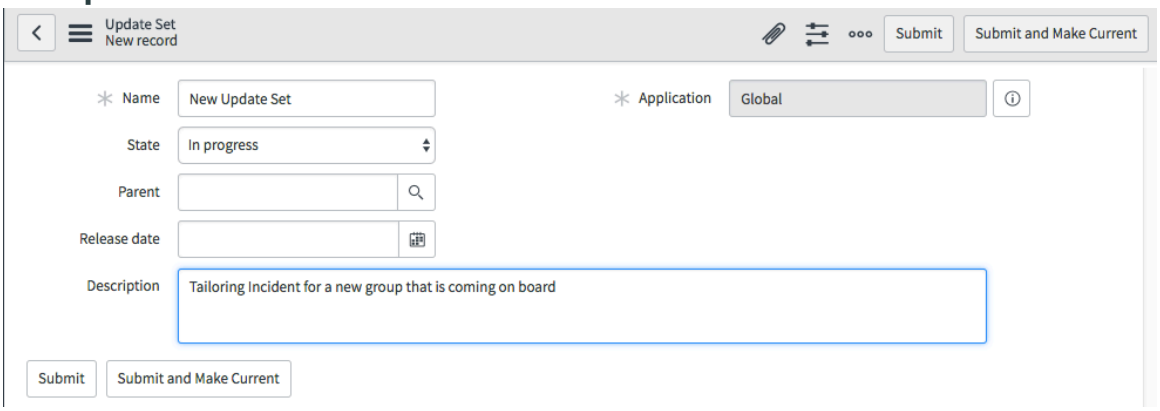
Before you begin
Role required: admin.

About this task
You can select an update set as the current set when you create it, or you can select it later from the Settings panel.

Procedure

1. Navigate to **All** > **System Update Sets** > **Local Update Sets** and click **New**.
2. Complete the form from the fields in the table.
3. Click **Submit** to create the update set.
4. If the picker is enabled and the update set is in the *In progress* state, click **Submit and Make Current** to select the new update set as the target for configuration changes.

New update set



Update Set Fields

Field	Description
Name	Enter a unique name for the update set. You can use naming conventions to organize update sets. For example, add the problem number to the name of the update that fixes it, identify the application scope, or use sequence numbers to keep track of the order in which update sets need to be committed.
State	Select **In progress** for a new update set. Selecting an *In progress* update set tracks customizations in the update set. The update set picker only displays *In progress* update sets. Select **Completed** only when you are certain that the update set is complete. After it is marked *Completed*, do not set it back to *In progress*. Instead, create a new update set with further customizations, and make sure to commit the update sets in the order that they were marked *Completed*. Use *Completed* update sets to transfer changes from one instance to another. Select **Ignore** when you are no longer working on an update set but do not want it to be transferred to another instance. You should always set *Completed* update sets on the production instance to *Ignore*. This state ensures the update set is not committed again when cloning the instance. Update set picker only displays *In progress* update sets. See Select the current update set in Unified Navigation. Select **Completed** only when you are certain that the update set is complete. After it is marked *Completed*, do not set it back to *In progress*. Instead, create a new update set with further customizations, and make sure to commit the update sets in the order that they were marked *Completed*. Use *Completed* update sets to transfer changes from one instance to another. Select **Ignore** when you are no longer working on an update set but do not want it to be transferred to another instance. You should always set *Completed* update sets on the production instance to *Ignore*. This state ensures the update set is not committed again when cloning the instance.
Release date	Enter the date on which you plan to release the update set.
Application	Populates the application scope that is currently selected in the Application picker. All changes in the update set apply only to the current scope.
Description	Enter a description of the update set.

Select the current update set in Unified Navigation

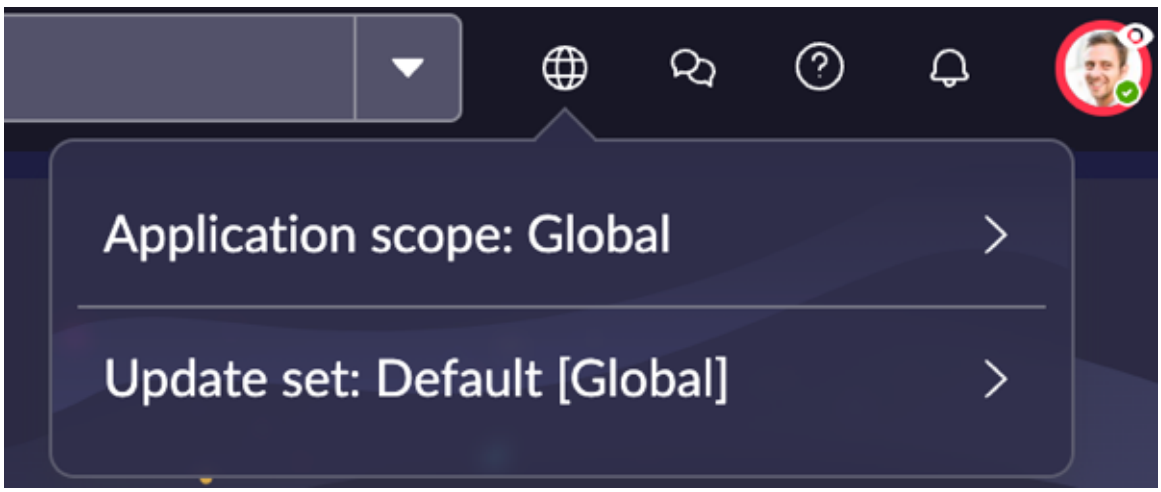
You can change the current update set at any time, using the update set picker in Unified Navigation.

Before you begin
Role required: admin.

About this task
When you change your application scope, the system automatically switches the current update set to be the application's default update set. However, you can choose a different update set.

Procedure

1. Select the Application scope icon ( ) in Unified Navigation.



2. Select the update set you want to make current from the list available in the **Update set** field, using **Filter** if needed.

The update set you selected is confirmed with a check mark icon.

3. **Optional:** If the update set you want isn't in the list, you can create an update set by selecting the plus icon (+).

What to do next

The update set picker is part of the Next Experience picker. For more information, see Exploring Next Experience pickers.

For more information about application scoping, see Application scope.

View customizations and compare with current version

View the customizations that make up an update set and compare the update to the current version.

Before you begin

Role required: admin

About this task

The Customer Update *[sys_update_xml]* table contains one record per customized object.

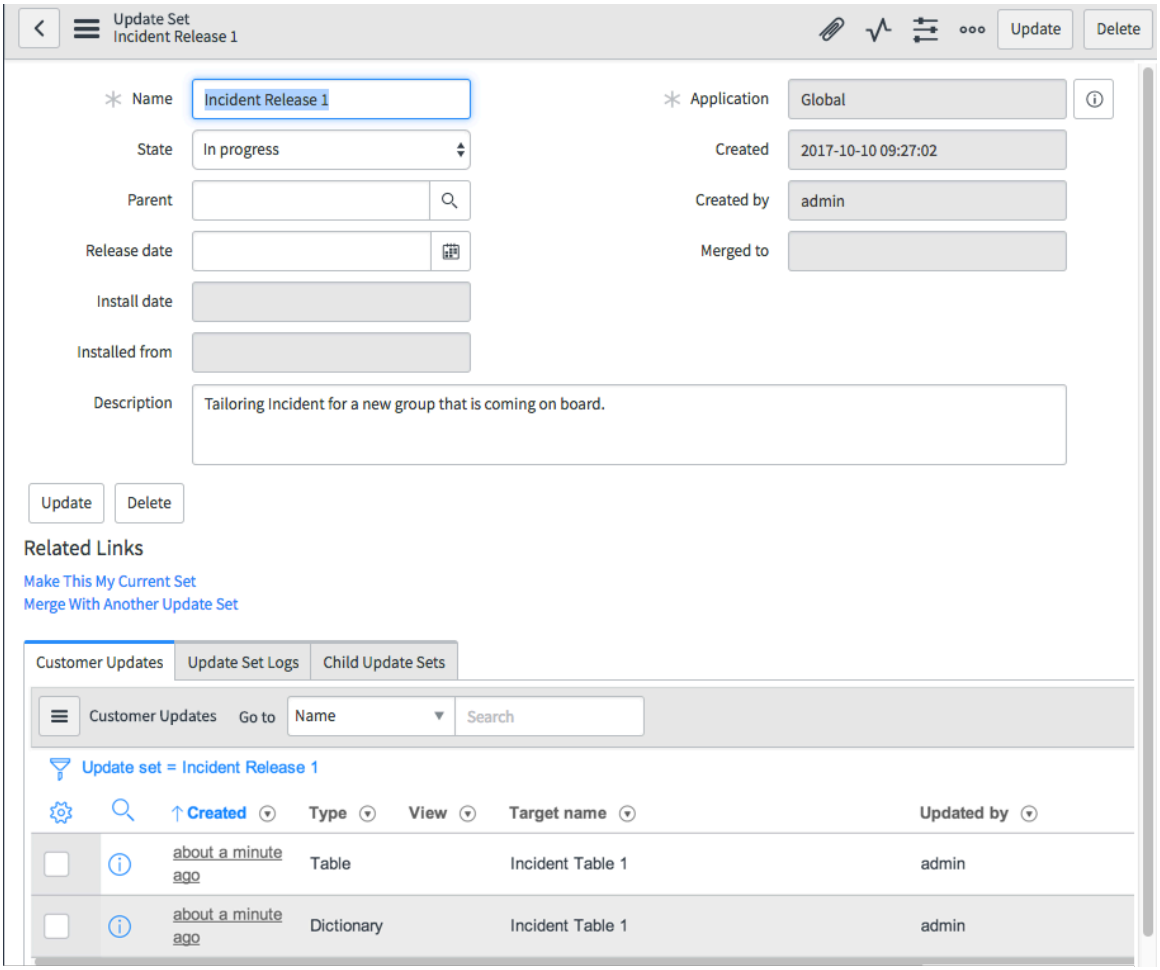
The customer update record specifies:

- The update set containing the customized object.
- The type of action applied to the customized object.
  - INSERT
  - INSERT_OR_UPDATE
  - UPDATE
  - DELETE
- The type of object customized.
- The target object of the update.
- The sys_id of the customized object (if it is a change to a particular record).
- The user who customized the object.
- The date and time the object was customized.

Procedure

1. Navigate to **All** > **System Update Sets** > **Local Update Sets**.
2. Click the update set name.
3. View the *Customer Updates* related list.

You can compare any update to the current version. Right-click the update record and select **Compare to Current**.



Related topics

- Compare two versions of an article
- Resolve conflicts for an individual record

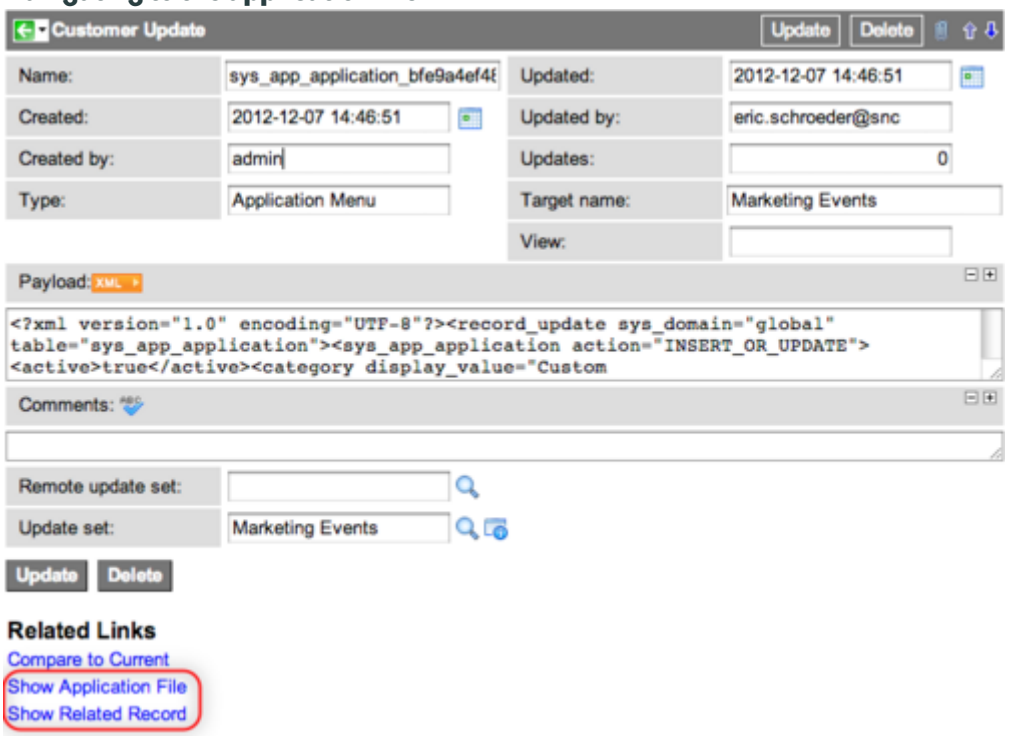
Navigation between records

You can navigate between a customer update record and the customized object or the application file for the object.

Navigate from an update record to:

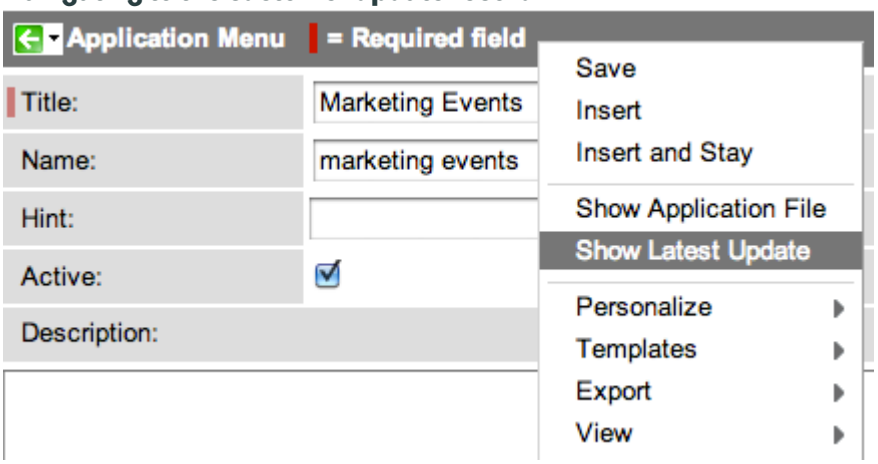
- The customized object, such as the application menu record: Click the **Show Related Record** related link.
- The application file record for the object: Click the **Show Application File** related link.

Navigating to the application file



To navigate from a customized object or an application file to the current customer update record: Click the form header and selecting **Show latest Update**.

Navigating to the customer update record



View a report on customizations and configuration changes

The base system provides reports for changes to the Incident table and changes by the current user.

Procedure

1. Navigate to **All** > **Reports** > **View / Run** and locate the *Customer Update* section.
2. Run any of the available reports or create a new report.

The following reports are available:

- *Application Changes (Incident)*: Displays all changes made to the Incident table. Select a different table and run the report again to view all changes to another application.
- *My Changes*: Displays all changes created or updated by the current user, grouped by table name.

Merge update sets

You can merge multiple update sets into a single update set. This capability is supported for backward compatibility with earlier releases of the ServiceNow® platform. The newer batch update sets feature accomplishes the same outcome with a more predictable and robust solution.

Before you begin

Note:
You cannot "unmerge" update sets once they have been merged. To learn more, see Update set batching.

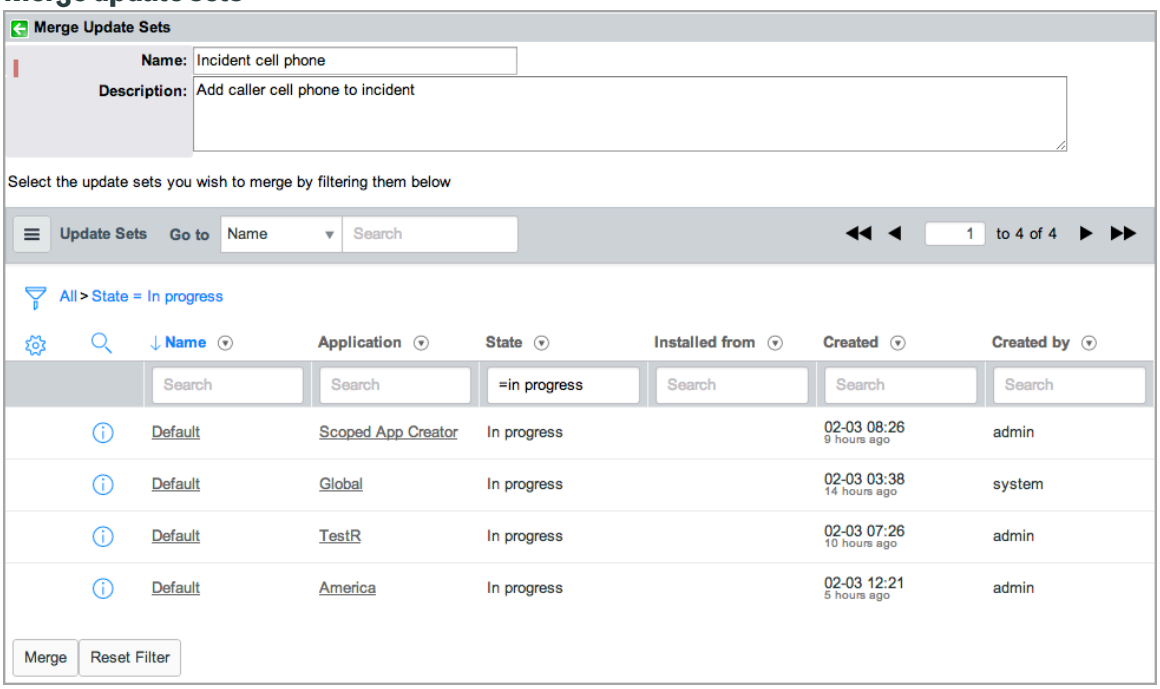
Procedure

1. Navigate to **All** > **System Update Sets** > **Merge Update Sets**.
   By default, the list is filtered to only show update sets that are *In progress*.

   Alternatively, navigate to **System Update Sets** > **Merge Completed Sets**. By default, the list is filtered to only show update sets that are in the *Complete* state. For example, you may want to use this filter after pushing changes or transferring update set from a development to a test instance.

2. Filter the list to show only the update set that you want to merge.
   You can only merge update sets that belong to the same application.

Merge update sets



3. Enter a **Name** for the new update set.
   Updates are moved to this new update set during the merge process.

4. **Optional:** Enter a **Description** for the update set.
5. Click **Merge**.
6. In the confirmation dialog box, click **OK**.
   - The new update set is created.
   - The most recent change for each object is moved from the original sets to the new set. Only changes that are not merged into the new set remain in the original sets. A message indicates how many updates were moved and how many were skipped. For example, if both update sets modify the Incident form, only the most recent change is moved to the new update set. The other modification remains in its original update set to provide a record of the changes that were not moved.

   Note:
   The system determines which record is the most recent by comparing the **Updated** field for the records, NOT the **sys_updated_on** value in the payload.

7. Verify that the correct changes were moved to the new set by scrolling down to the *Merged Update Sets* related list and opening the old update set records.
8. Delete or empty the original update sets to avoid committing an older change by mistake. The system does not remove updates that were not merged into the new set. Do not move updates "left behind" in old sets into the new set.

Compare local update sets

Administrators can preview local and remote (retrieved) update sets and compare the sets with one another to resolve conflicting changes.

About this task

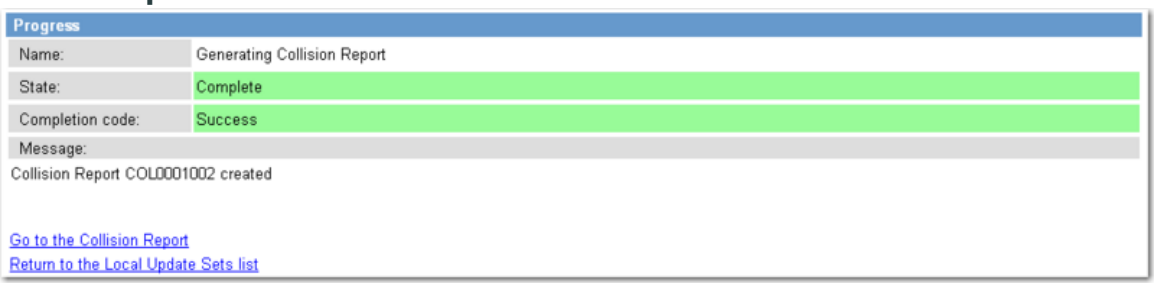
Compare local update sets to identify collisions and ensure that the proper changes are being committed. Resolve all conflicts before moving an update set between instances.

Procedure

1. Navigate to **All** > **System Update Sets** > **Local Update Sets**.
2. Select the check boxes beside the update sets to compare.
3. In the *Action* choice list, select **Compare Update Sets**.

   The progress screen appears as ServiceNow generates the collision report.

Collision report

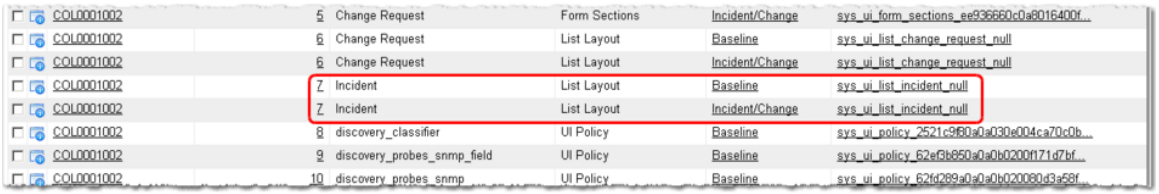


4. Click **Go to the Collision Report** when the report is complete.

   The Update Set Collisions list appears, showing all the changes in the selected sets.

5. Inspect the list for collisions by locating duplicate *Collision Numbers* that show the same change in separate update sets.

Update set collisions

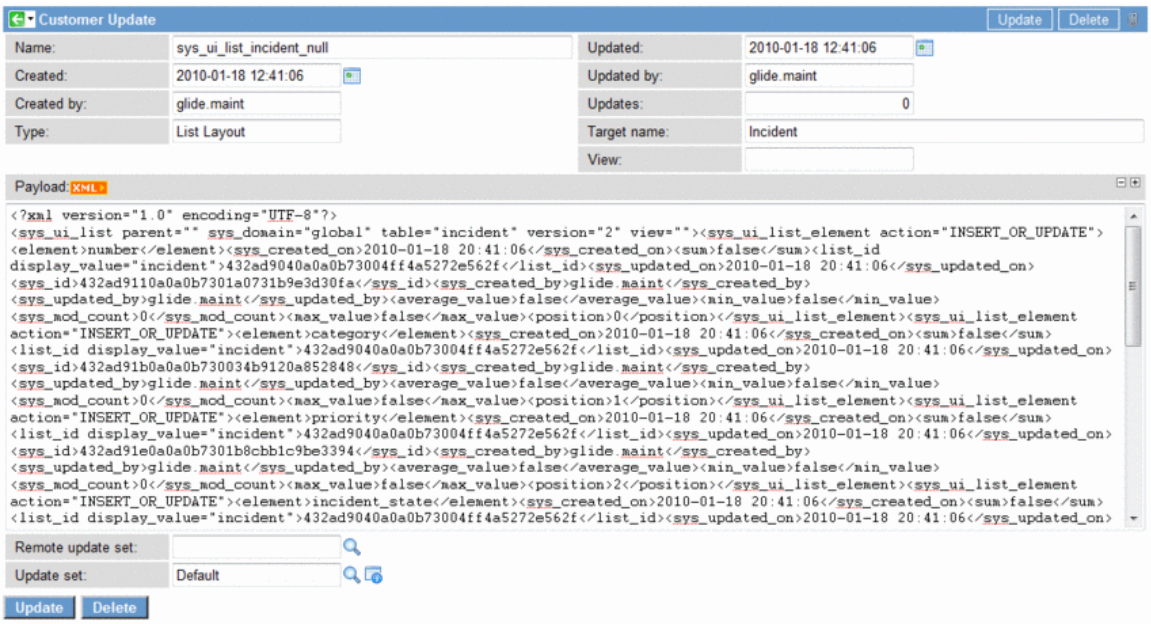


COL0001002	5	Change Request	Form Sections	Incident/Change	sys_ui_form_sections_ee936660c0a8016400f...
COL0001002	6	Change Request	List Layout	Baseline	sys_ui_list_change_request_null
COL0001002	6	Change Request	List Layout	Incident/Change	sys_ui_list_change_request_null
COL0001002	7	Incident	List Layout	Baseline	sys_ui_list_incident_null
COL0001002	7	Incident	List Layout	Incident/Change	sys_ui_list_incident_null
COL0001002	8	discovery_classifier	UI Policy	Baseline	sys_ui_policy_2521c9f80a0a030e004ca70c0b...
COL0001002	9	discovery_probes_snmp_field	UI Policy	Baseline	sys_ui_policy_62ef3b850a0a0b0200f171d7bf...
COL0001002	10	discovery_probes_snmp	UI Policy	Baseline	sys_ui_policy_62fd289a0a0a0b020080d3a58f...

6. Resolve the collision by deleting the unwanted update record from one of the update sets.
   a. Click the link in the *Sys update* column for the unwanted update (`sys_ui_list_incident_null` in the example).
   b. Click **Delete**.

> **Note:**
> You must open the update record to delete the record. You cannot delete the update by selecting the check box for the entry in the Update Set Collisions list and using the *Delete* action. When you delete the update record, the customization is not backed out of the instance. Only the record of the customization is deleted.

Customer updates



Customer Update — Update | Delete

Name:	sys_ui_list_incident_null	Updated:	2010-01-18 12:41:06
Created:	2010-01-18 12:41:06	Updated by:	glide.maint
Created by:	glide.maint	Updates:	0
Type:	List Layout	Target name:	Incident
		View:	

Payload: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<sys_ui_list parent="" sys_domain="global" table="incident" version="2" view=""><sys_ui_list_element action="INSERT_OR_UPDATE">
<element>number</element><sys_created_on>2010-01-18 20:41:06</sys_created_on><sum>false</sum><list_id
display_value="incident">432ad9040a0a0b73004ff4a5272e562f</list_id><sys_updated_on>2010-01-18 20:41:06</sys_updated_on>
<sys_id>432ad9110a0a0b7301a0731b9e3d30fa</sys_id><sys_created_by>glide.maint</sys_created_by>
<sys_updated_by>glide.maint</sys_updated_by><average_value>false</average_value><min_value>false</min_value>
<sys_mod_count>0</sys_mod_count><max_value>false</max_value><position>0</position></sys_ui_list_element><sys_ui_list_element
action="INSERT_OR_UPDATE"><element>category</element><sys_created_on>2010-01-18 20:41:06</sys_created_on><sum>false</sum>
<list_id display_value="incident">432ad9040a0a0b73004ff4a5272e562f</list_id><sys_updated_on>2010-01-18 20:41:06</sys_updated_on>
<sys_id>432ad91b0a0a0b730034b9120a852848</sys_id><sys_created_by>glide.maint</sys_created_by>
<sys_updated_by>glide.maint</sys_updated_by><average_value>false</average_value><min_value>false</min_value>
<sys_mod_count>0</sys_mod_count><max_value>false</max_value><position>1</position></sys_ui_list_element><sys_ui_list_element
action="INSERT_OR_UPDATE"><element>priority</element><sys_created_on>2010-01-18 20:41:06</sys_created_on><sum>false</sum>
<list_id display_value="incident">432ad9040a0a0b73004ff4a5272e562f</list_id><sys_updated_on>2010-01-18 20:41:06</sys_updated_on>
<sys_id>432ad91e0a0a0b7301b8cbb1c9be3394</sys_id><sys_created_by>glide.maint</sys_created_by>
<sys_updated_by>glide.maint</sys_updated_by><average_value>false</average_value><min_value>false</min_value>
<sys_mod_count>0</sys_mod_count><max_value>false</max_value><position>2</position></sys_ui_list_element><sys_ui_list_element
action="INSERT_OR_UPDATE"><element>incident_state</element><sys_created_on>2010-01-18 20:41:06</sys_created_on><sum>false</sum>
<list_id display_value="incident">432ad9040a0a0b73004ff4a5272e562f</list_id><sys_updated_on>2010-01-18 20:41:06</sys_updated_on>
```

Remote update set:
Update set: Default
Update | Delete

7. Run the comparison again to make sure all collisions have been resolved.

Update set collision resolution

A collision is an update that has a newer local update.

The platform detects collisions by comparing the values in the *Name* and *Updated* fields of the customer update record from each update set. If the name matches but there are different update date values, then there is a collision.

When a customer update is moved from one instance to another, it may be re-written to match the target instance. The re-write can involve changing the update name of the customer update and one or more `sys_ids` within the update. The re-writes are done when the record or the

reference field is for a table that uses a coalesce strategy. This ensures that the customer update will be applied to the correct record. For example, if the `sys_dictionary` record for `tablename.fieldname` has `sys_id` `123456789` on instance A and `sys_id` `987654321` on instance B, when a customer update that refers to that record is retrieved from instance A and recorded in the sys_update_xml table on instance B, references to `123456789` are updated to read `987654321`.

Coalesce strategies

Update sets can detect collisions between identical records that you independently create on separate instances. To detect such collisions, the record must have a coalesce strategy based on coalescing columns. Because collision detection depends on uniqueness of tables, the tables must be unique when the coalescing columns are combined. Records that are not listed here will not collide if the same record is created separately on different instances.

Type	Coalescing Columns
atf_input_variable	name, element
atf_output_variable	name, element
dp_data_pattern	source_sys_id
dynamic_attribute	namespaced_name
dynamic_category	namespaced_name
dynamic_category_member	category, attribute
dynamic_choice_override	choice, category, attribute
dynamic_namespace	name
sys_analytics_bucket	sys_scope, bucket_document_id, bucket_table_name
sys_attachment	(uses custom matching logic)
sys_choice_set	name, element
sys_collection	collection, name, join_field
sys_db_object	name
sys_df_data_dictionary	name, element
sys_dictionary	name, element
sys_documentation	name, element, language
sys_index	logical_table_name, col_name_string
sys_module	path
sys_notification_category	name
sys_package	source
sys_package_dependency_m2m	dependency, sys_package
sys_properties	name
sys_report_chart_color	name, element, value
sys_scope_script_access	source_scope, target_scope, script_name
sys_scope_table_access	source_scope, target_scope, table_name

Type	Coalescing Columns
sys_script_validator	internal_type, ui_type
sys_translated	name, element, value, language
sys_translated_text	tablename, fieldname, documentkey, language
sys_ui_form	name, view, sys_domain
sys_ui_list	name, view, sys_domain, element, relationship, parent
sys_ui_message	key, language, code
sys_ui_related_list	name, view, related_list, sys_domain
sys_ui_section	name, view, caption, sys_domain
sys_ui_view	name
sys_user_group	name
sys_user_role	name
sys_wizard	name
ua_table_licensing_config	name

How customer update record names affect collisions

To understand coalescing, it helps to understand how records that do not coalesce work. For most record types, when a customer update is moved to a new instance, the system does not detect collisions for the following reason:

- When you create a record, it receives a unique sys_id. For most record types, the sys_id becomes part of the customer update record name. For example: `sysevent_email_template_9e1998c078b71100a92ecacd80df1d39`.
- Creating an identical record in the same table on another instance produces a customer update record name with a different sys_id. For example: `sysevent_email_template_10b958c8653311005840134572f8e020`

As a result, even though the records might be otherwise identical, the records have different names so the system does not detect the collision.

Coalescing records, in contrast, use the following approach to naming records and determining collisions: The following customer update record types use some or all of their coalescing columns instead of the sys_id in their names.

- sys_dictionary
- sys_documentation
- sys_choice_set
- sys_ui_list
- sys_ui_related_list

The resulting identical record name in each instance helps the system to identify collisions even if the records have different sys_ids.

When a customer update is moved from one instance to another, it may be re-written to match the target instance. The re-write can involve changing the update name of the customer update and one or more sys_ids within the update. The re-writes are done when the record or the reference field is for a table that uses a coalesce strategy. This ensures that the customer

update will be applied to the correct record. For example, if the sys_dictionary record for tablename.fieldname has sys_id "123456789" on instance A and sys_id "987654321" on instance B, when a customer update that refers to that record is retrieved from instance A and recorded in the sys_update_xml table on instance B, references to "123456789" are updated to read "987654321".

Preventing duplicate records

- Transfer data with update sets rather than recreating it on separate instances to ensure the records have the same sys_id.
- Export and import records as XML files to ensure the records have the same sys_id. See Export and import XML files.
- Enable a unique index for the table from the system dictionary. See Table administration.

Note:
The default records included in the baseline system will always have the same Sys ID because the instance imports the records as XML files during instance provisioning.

Mark an update set complete

When you have completed the customizations and compared local update sets to resolve conflicts, mark the update set as *Complete*.

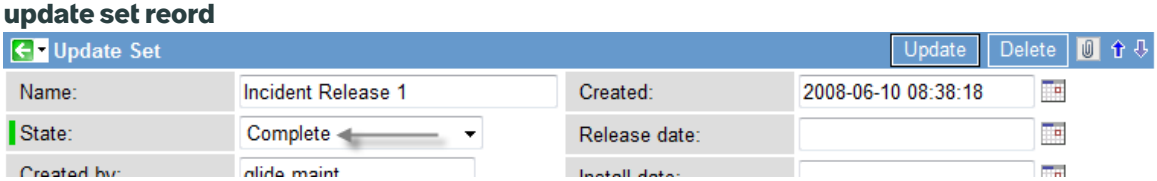
About this task

Mark an update set as *Complete* only when it is ready to migrate. Once an update set is complete, do not change it back to *In progress*. Instead, create another update set for the rest of the changes, and be sure to commit them together in the order that they were created. Naming conventions may help in this case (for example, Performance Enhancements and Performance Enhancements 2).

Procedure

1. Open the update set record.
2. Change the *State* of the update set from *In progress* to *Complete*.
   - The update set is available for other instances to retrieve.
   - No additional customizations are tracked in the update set.

update set reord



Save an update set as a local XML file

Administrators can export an update set as an XML file to save a specific version of an application or set of changes.

Before you begin

Role required: admin

About this task

Typically you create an XML file of an update set when one of the following conditions apply:

- The two instances do not have network connectivity so you cannot retrieve update sets from the remote instance nor create a data source to pull, or import, data directly from the source instance.
- You do not want to provide administrator credentials to the source instance (for example, you do not want to share an administrator password with people outside your company) so you cannot retrieve update sets nor create a data source.
- You want to back up important customizations locally.

The ability to export and import customizations as an XML file is provided by the following UI Actions:

- **Export to XML** on the Update Set [sys_update_set] table.
- **Export to XML** on the Retrieved Update Set [sys_remote_update_set] table.
- **Import Update Set from XML** on the Retrieved Update Set [sys_remote_update_set] table.

The **Export to XML** UI Action on Update Set [sys_update_set] table calls a processor called `UnloadRetrievedUpdateSet`, which transforms a local update set into a retrieved update set, exports the retrieved update set with its related list, and then deletes the temporary update set, if necessary.

Both **Export to XML** UI actions depend on the script include `ExportWithRelatedLists`, which exports a record and manually defined related lists to a single XML file.

Procedure

1. Navigate to **System Update Sets** and click either **Local Update Sets** or **Retrieved Update Sets**.

   Note:
   Retrieved Update Sets must be in a loaded state to be exported.

2. Select an update set that is in the **Complete** state.
3. On the Update Set form, click the **Export to XML** Related Link.
4. Save the XML file.
   An XML file is created. When the file is uploaded to another instance, it appears as a retrieved update set regardless of whether it is local or retrieved on the instance where it is created.

Load customizations from a single XML file

Administrators can load an update set XML file to apply a specific version of an application or set of changes.

Before you begin

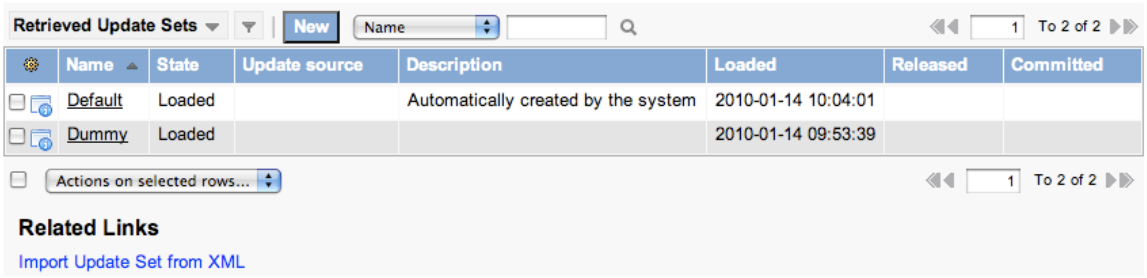
Roles required: admin

Procedure

1. Elevate privileges to the security_admin role.
2. Navigate to **System Update Sets** > **Retrieved Update Sets**.
3. Click the link **Import Update Set from XML**.
4. Click **Choose File** and select an XML file.
5. Click **Upload**.
   The customization is now available as a retrieved update set with state *Loaded*.

6. Follow standard procedure to commit the update set.

Retrieved update sets



Update set transfers

When an update set is completed, you can transfer it between instances to move customizations from development, through testing, and into production.

Warning:
Update sets allow moving changes between instances that may be running different family release versions and different features. You can always load an update set created on an older family release on an instance running a newer family release. Loading an update set created on a newer family release on an instance running an older family release requires additional testing to determine compatibility. Updates from newer family releases may not produce the same functionality when moved to older family releases. In extreme cases, newer family release updates may cause outages or data loss on an older family release instance. Where possible, avoid moving updates from newer family releases to older family releases. Similar constraints apply to moving updates between instances running different versions of ServiceNow Store apps.

Transferring with IP access control

If IP address access control is enabled on the source instance or the source instance resides in a different datacenter than the target instance, complete these tasks before transferring an update set:

1. Contact Customer Service and Support to find out the IP addresses of all application nodes supporting your instance.
2. On the source instance, navigate to System SecurityIP Address Access Control.Add the IP address from step one as an exception.

Transferring with basic authentication

If the source instance has basic authentication turned on for SOAP requests, you must use valid credentials to retrieve update sets.

Transferring with an XML file

You can unload an update set as an XML file and then transfer it to another instance. See Save an update set as a local XML file for details.

Retrieve an update set

Retrieve completed update sets from another instance.

About this task

Warning:
Update sets allow moving changes between instances that may be running different family release versions and different features. You can always load an update set created on an older family release on an instance running a newer family release. Loading an update set created on a newer family release on an instance running an older family release requires additional testing to determine compatibility. Updates from newer family releases may not produce the same functionality when moved to older family releases. In extreme cases, newer family release updates may cause outages or data loss on an older family release instance. Where possible, avoid moving updates from newer family releases to older family releases. Similar constraints apply to moving updates between instances running different versions of ServiceNow Store apps.

Procedure

1. If IP address access control is enabled on the source instance, set up the target instance as an exception.
2. On the target instance, navigate to **System Update Sets** > **Update Sources** and click **New**.
3. Specify the connection settings as described in the table.

Field	Description
Name	Enter a unique name for the instance.
Type	Specify whether the remote instance is a development, test, or UAT instance.
Active	Specify whether the local instance can transfer update sets to the remote instance. You can transfer update sets only to active remote instances.
URL	Specify the URL of the remote instance using the appropriate transfer protocol. Each remote instance record should have a unique URL. Creating duplicate records with the same URL can cause errors. The remote instance must be on the same release family as the local instance. **Note:** You cannot change the URL after the system verifies the connection. Use the *Active* field to deactivate unwanted remote instances.
Username	Enter the user on the remote instance who authorizes transferring update sets to this the instance. This user account must have the admin user role on the remote instance.
Password	Enter the password of the authorizing user.
Short description	[Optional] Enter any other relevant information about the remote instance.

4. Click **Test Connection**.
   - If the connection is successful, a confirmation message appears.
   - If the connection fails, a warning message identifies the cause of the failure.
5. If the connection fails, modify the settings to establish connectivity.

- You must establish connectivity before you can save the connection settings.
- You may want to modify the source instance (for example, change the password).

6. Right-click the form header and select **Save**.
7. Under **Related Links**, click **Retrieve Completed Update Sets**.
   - Any update sets marked as **Completed** are transferred from the source instance to the target instance. Update sets that already exist on the target instance are skipped.
   - The confirmation page provides detailed messages about how many update sets were transferred and how many were skipped.
   - To view retrieved update set, navigate to **System Update Sets** > **Retrieved Update Sets**.

What to do next

If the system property **glide.update_set.auto_preview** is set to **true**, the system automatically starts the preview process after the update set is retrieved. If this property is **false**, you must start the process manually. For more information on the preview process, see Preview a remote update set.

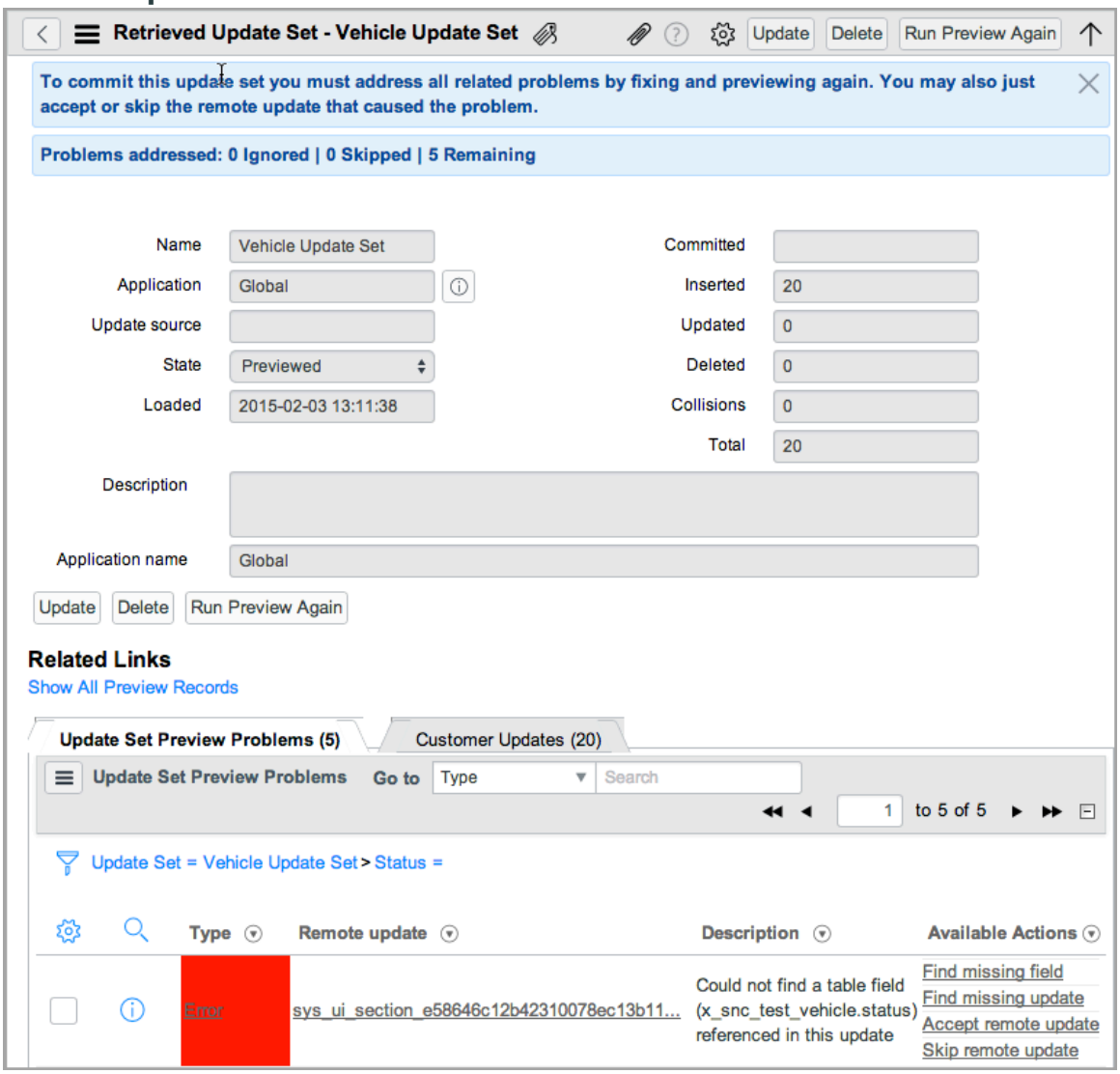
Preview a remote update set

Previewing compares an update set retrieved from a remote instance to updates on the local instance to detect potential problems. You must preview an update set and address all problems before you can commit the update set.

Procedure

1. If the system property **glide.update_set.auto_preview** is set to **true**, the system automatically starts the preview process after the update set is retrieved.
   If this property is **false**, the preview process must be started manually.
   a. Navigate to **System Update Sets** > **Retrieved Update Sets**.
   b. Click **Preview Update Set**.
   For large update sets, the preview process may require a significant amount of time. If necessary, you can cancel the preview process by clicking the **Cancel** button on the progress dialog box.
   The Update Set Preview page shows results and lists problems. Read the information and click **Close**.
2. On the Retrieved Update Set form, make a selection.

Retrieved Update Set form



3. **Optional:** Preview the records.
   a. Open the update set record and click **Show All Preview Records** to make sure the correct updates are being committed.
   b. Open the update set record and click **Run Preview Again** to run the comparisons again.
   c. Review the **Update Set Preview Problems** related list to ensure that the correct updates are being committed.

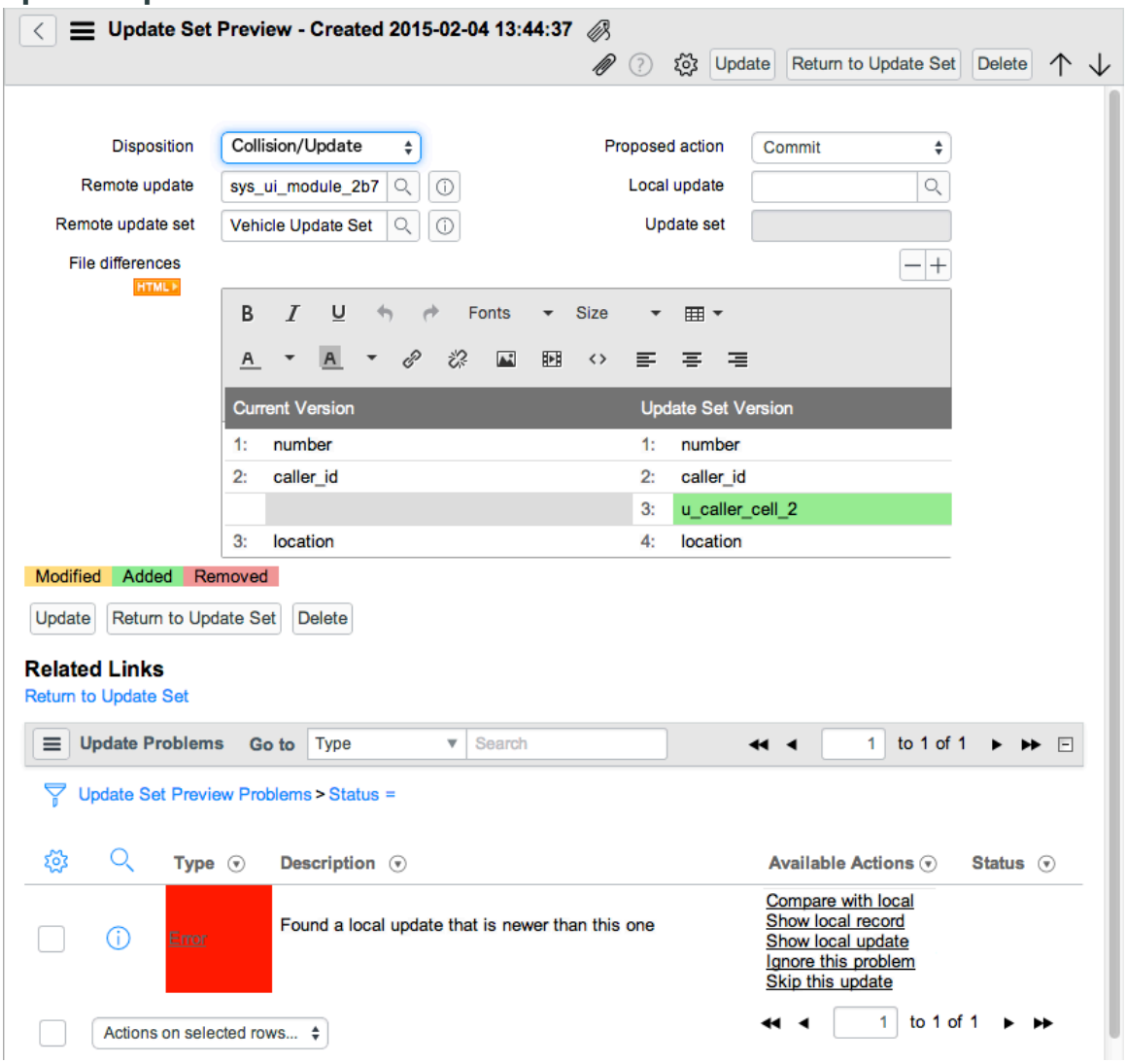
Review a preview record for an update set

The process of previewing an update set creates a preview record for each update. You can review the preview records to make sure that the correct updates are being committed.

Procedure

1. Open the Update Set record and preview the update set.
2. Click the **Show All Preview Records** related link.
3. Click the **Disposition** to open a preview record and then review the information (see table).
4. Fill in the fields on the form, as appropriate.

Update set preview



Field	Description
Disposition	Indicates when a collision is detected: ○ **Collision/Update**, **Collision/Insert**, or **Collision/Delete:** the change is older than a change to the same object on the local instance. ○ **Update**, **Insert**, or **Delete:** the change does not conflict with a change on the local instance.
File differences	Compares the most recent version of the object on the local instance with the version in the update set. Differences are marked with a color key. Deletions are highlighted in red, additions in green, and modifications in yellow.
Proposed action	Indicates how to handle the change when the update set is committed. ○ **Commit:** Accept the change in the remote update. The default proposed action for every preview record is Commit, even if a newer update exists on the instance. ○ **Skip:** Reject the change.

5. If necessary, resolve any problems listed in the **Update Problems** related list.

6. In the Proposed action field, select the action to take when committing the update set.
7. Click **Update** to save the action.
8. Repeat the process for every preview record.

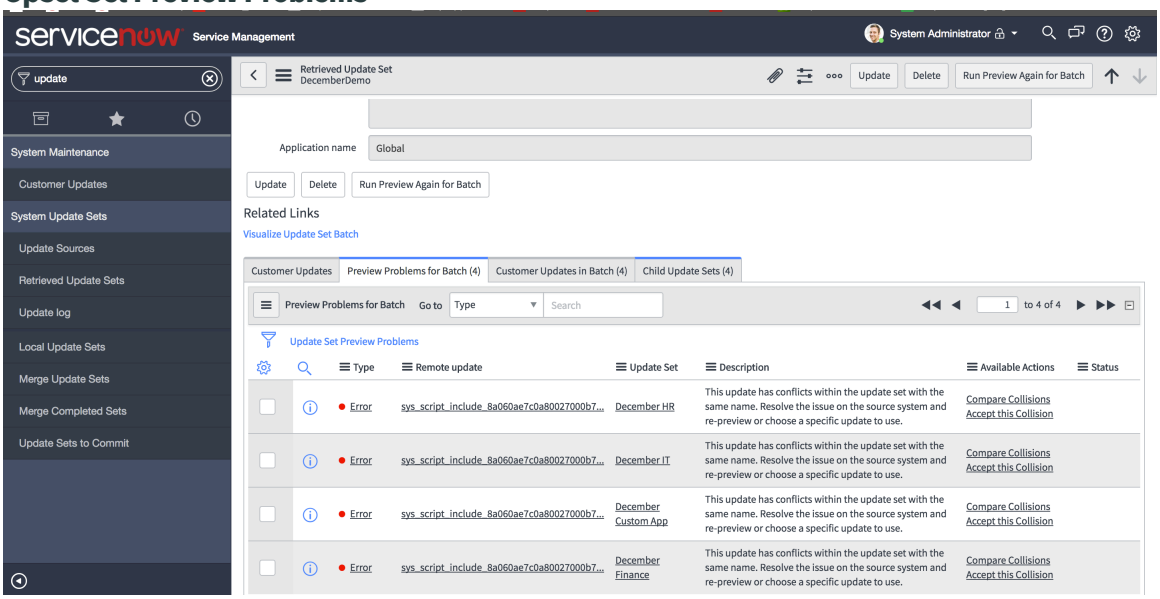
Resolve a preview problem in an update set

Preview an update set to detect and resolve problems that may occur if you commit the updates on the local instance.

Procedure

1. Navigate to **All** > **System Update Sets** > **Retrieved Update Sets**.
2. Open the update set record and scroll to the **Update Set Preview Problems** related list.

Upset Set Preview Problems



3. Review each problem description to determine the cause and resolve the problem.
   Update Set Preview Problems

   Missing Object

   Example problem text: `Could not find a record in sys_ui_policy referenced in this update.`

   Description: The object or a referenced object does not exist on the target instance. For example:

   - An update modifies the form layout for a table that has not been created in the local instance.
   - A UI policy action is included in the update set, but the parent UI policy is not.
   - Resolution: Create another update set on the source instance to transfer the missing object to the local instance, or create the object on the local instance. Use these **Available Actions** to assist in problem resolution:
     - **Find missing field** or **Find missing record**: Opens a new window and searches the source instance for the missing field (dictionary entry) or record.
     - **Find missing update**: Opens a new window and searches the source instance for the update record that corresponds to the missing field or record.

Collision

Example problem text: `Found a local update that is newer than this one`

Description: A change in the update set is older than a change to the same object on the local instance.

Resolution: Compare the two updates and determine which version to use. To use the version on the local instance, select **Skip remote update**. To use the version in the update set, select **Accept remote update**. Use these **Available Actions** to assist in problem resolution:

- **Compare with local**: opens the preview record, which provides a comparison of the differences between the local version and the version in the update set.
- **Show local field** or **Show local record**
- **Show local update**

Uncommitted update

Example problem text: `Could not find a table field (u_case.u_reference) referenced in this update, but did find it in another uncommitted update set`

Description: The object exists in another remote update set that has not been committed.

Resolution: Commit the other remote update set first or move this update to the other update set. Use these **Available Actions** to assist in problem resolution:

- **Show uncommitted update**: opens the update record in the other remote update set.
- **Show uncommitted Update Set**: open the other remote update set record.

Table to be deleted has data

Example problem text: `Found a row in the table that is going to be deleted`

Description: One difference between table deletes and other metadata deletes is that the table data is lost when the table is deleted. (If the table is empty (no rows), then no problem is generated.)

Resolution: The problem must be ignored (delete will happen) or skipped (delete will not happen) before the update set can be committed. You can restore the table, but the restore does not bring back the data.

You are not allowed to delete system tables (ServiceNow tables) or tables outside your application scope.

Application scope validation issue

Description: The previewer identifies the following combination of states as a problem:

- The scope for the update set is not Global and
- The application is not found on the target instance and
- The application is not included with the update set and
- The application is not found on the ServiceNow Store.

Resolution: Transfer the update set only to instances that include the application scope or ensure that the update set includes the application.

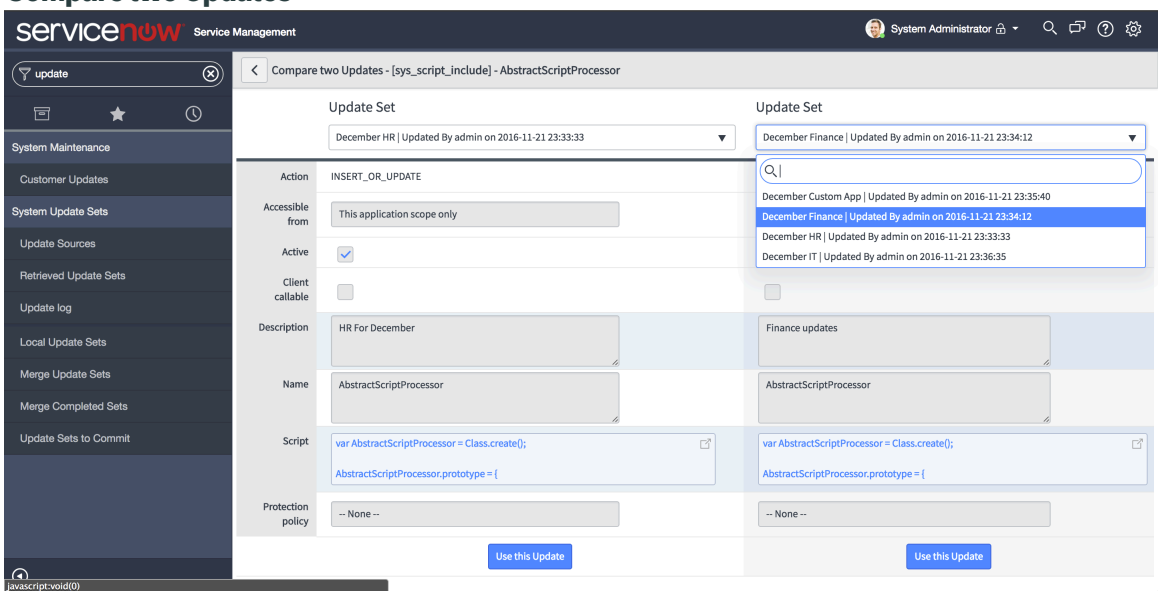
Conflict within a single batch

Example problem text: `This update has conflicts within the update set with the same name. Resolve the issue on the source system and re-preview or choose a specific update to use.`

Description: Two or more update sets within the same batch have conflicting changes. The **Update Set Preview Problems**list contains a record for each update set with a conflicting change.

Resolution: Compare the conflicting update sets and determine which version to use. If you know which update set is the correct one to use, select the row for that set and click **Accept this collision.** Otherwise, click **Compare Collisions** to compare the conflicting update sets.

Compare two Updates



From this screen, you can compare any two of the conflicting update sets and choose the update set to commit.

Commit an update set

When you have previewed an update set and have resolved any issues, commit the update set. Committing an update set applies all changes to the instance and creates a local copy of the update set that contains an update record for every change.

Procedure

1. Navigate to **All** > **System Update Sets** > **Retrieved Update Sets** and open the update set.
2. Resolve any problems.
   You cannot commit an update set until all problems are resolved.
3. Click **Commit Update Set**.

- Click **Cancel** to return to the preview and reevaluate the change. None of the updates are committed.
- Click **OK** to skip the change and continue committing the changes that are marked as **Commit**.

If the update set contains one or more DELETEs for schema, the system displays a warning. The warning lists up to five updates that may contain problems. If more than five updates have potential problems, the system provides a link.

When the system successfully commits an update set, it displays a completion page.

4. Click **Commit log** on the confirmation page, or navigate to **System Update Sets** > **Update log** and filter for the update set name.
   - Look for warnings that contain the text **unsafe edit**. The system automatically skips any changes that results in data loss, such as changing the type of a field that contains data. You must manually make any of these changes, if necessary. Use caution when making changes that affect production data.
   - Look for errors that indicate which records failed to commit and why. Create a new update set to address those failures, if necessary.
5. When you are no longer working on the update set but do not want it transferred to another instance, navigate to **System Update Sets** > **Local Update Sets** and open the local update set record.
   Change the State to **Ignore**.

What to do next
For completed update set on the production instance, you should always change the state to **Ignore**. This state ensures that the update set is not committed again when cloning the instance.

Back out an update set

You can back out changes to existing records for any committed update set.

Before you begin
Role required: admin

About this task
Backing out an update set creates delete updates in the current update set. If you commit, back out, and then reapply a remote update set, errors appear in the previewer because the deleted updates are considered more recent changes and cause collisions.

> **Warning:**
> Do not back out the **Default** update set. This action can damage the configuration of the instance.

The back out process reverses both record-level updates and changes to the dictionary. Some changes caused by a back-out can result in data loss. These are the expected results of the back-out process:

Customer Update	Result of the back out action
A new table	The table is dropped from the database, deleting any data from it.
A new field	The field is dropped from the database, deleting any data from it.
A deleted field	The field is restored to the database, but the original data is lost.

Customer Update	Result of the back out action
A resized field	The field resize is reversed. If the field has been increased, data is truncated first to avoid errors.
A configured form	The form is reverted to its previous state.
A record is inserted	The record is deleted. (See **Note** below)
A record is deleted	The record is restored with its original data.

Warning:
Backing out an update set that belongs to an update set batch may affect other update sets in the batch. For more information, see Back out batched update set.

- If the sys_package is global, it is deleted.
- If the sys_package is not global, and it has a value, a warning displays that there is no deletion. Rather, the sys_update_xml is put into the default update set and the record is left in place.

Procedure

1. Navigate to **All** > **System Update Sets** > **Local Update Sets**.
2. Open the update set record.
   Make sure it is complete and a date is set on the **Release date** field.

   Note:
   The currently selected application affects what options are available for the update set. Make sure you select the application, such as Global, that matches the contents of the update set.

3. Carefully review the contents of the update set and consider whether there will be problems if it is backed out.
   If backing out an update set is not sufficient or will cause issues, then, instead, create and commit a new update set to reverse the customizations.
4. Click **Back Out**.
   A Progress page displays actions, progress, and problems. Problems are changes in more recent update sets that affect the update set that is being backed out. The backout preview process generates a warning for each problem.
5. Resolve each problem before proceeding with the back out.
   - To keep the latest version, click **Decide to Keep Current**.
   - To back out to the previous version, click **Decide to Use Previous**.

   All changes are reversed as described in the table. The current update set tracks all of the new changes that occur.

The update set and all associated update records are deleted. If needed, you can still navigate to the retrieved update set, preview it, and commit it again.

Note:
If you commit, back out, and then reapply a remote update set, errors appear in the previewer because backing out an update set creates delete updates in the current update set. The deletes are considered more recent changes and cause collisions.

Cautions about deleting update sets

Deleting an update set is a bad practice. To revert a customization, back out the update set rather than deleting it.

Administrators can delete an update set only when it is not the current update set and it is empty (no `sys_update_xml` entries are associated with it). For example, after merging update sets, you might want to delete the original sets. This function is restricted by an access control rule (ACL) on the Update Set [sys_update_set] table.

Do not delete `sys_update_xml` entries, because this action:

- Does not undo the updates.
- Removes any record of who applied the customizations.
- Removes the `sys_update_xml` entries associated with the update set, so customizations are overwritten when the instance is upgraded.

When you try to delete an update entry, a warning message appears. Click **OK** to confirm the deletion.

Update set batching

Batch update sets enable you to group update sets together so you can preview and commit them in bulk.

Dealing with multiple update sets can lead to problems, including committing update sets in the wrong order or inadvertently leaving out one or more sets. You can avoid these problems by grouping completed update sets into a batch.

Note:
To preserve the update set batch hierarchy while cloning, set only the parent update set to 'Ignore' status and leave all other update sets as 'Complete'. This will prevent batch sets from cloning and preserves the batch update set hierarchy.

The system organizes update set batches into a hierarchy. One update set can act as the parent for multiple child update sets. A given set can be both a child and parent, enabling multiple-level hierarchies. One update set at the top level of the hierarchy acts as the base update set.

When you preview or commit the base update set, you preview or commit the entire batch. The system determines the processing order, and checks for collisions, based on the dates the changes were recorded, and on their sequential ancestry. Their ancestries are the specific instances in which the changes in the update sets took place.

Note:
For more details, see Compare local update sets and View customizations and compare with current version.

Example of batched update sets

The list of update set records reflects the batch hierarchy in the **Parent** and**Batch Base** columns.

List of batched update sets

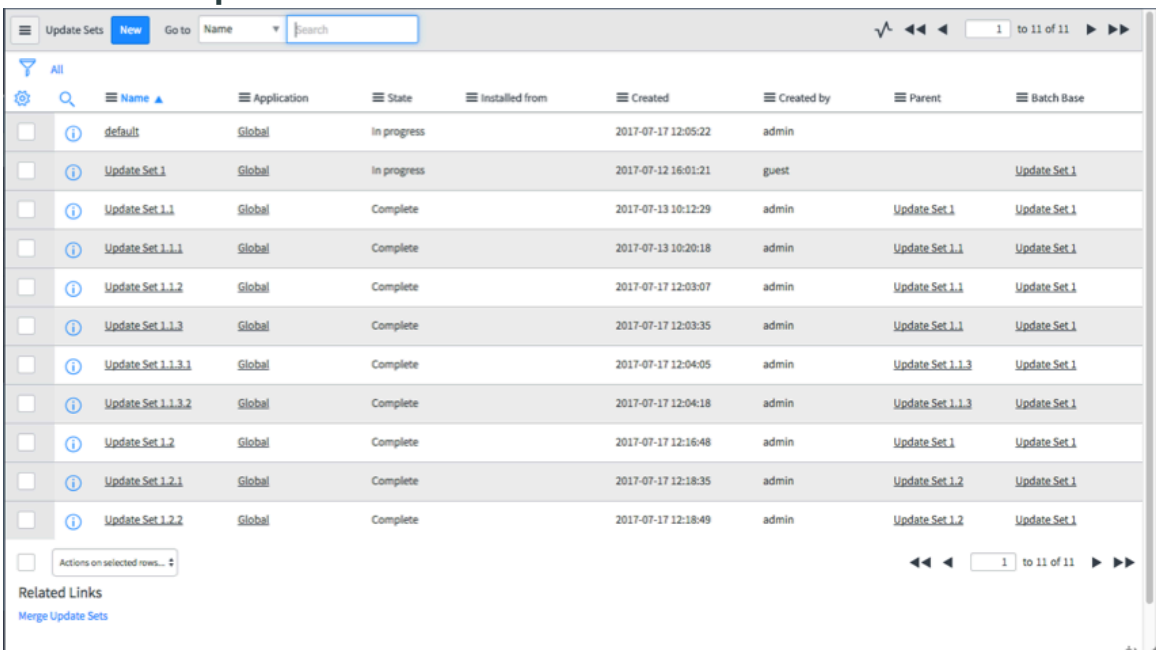
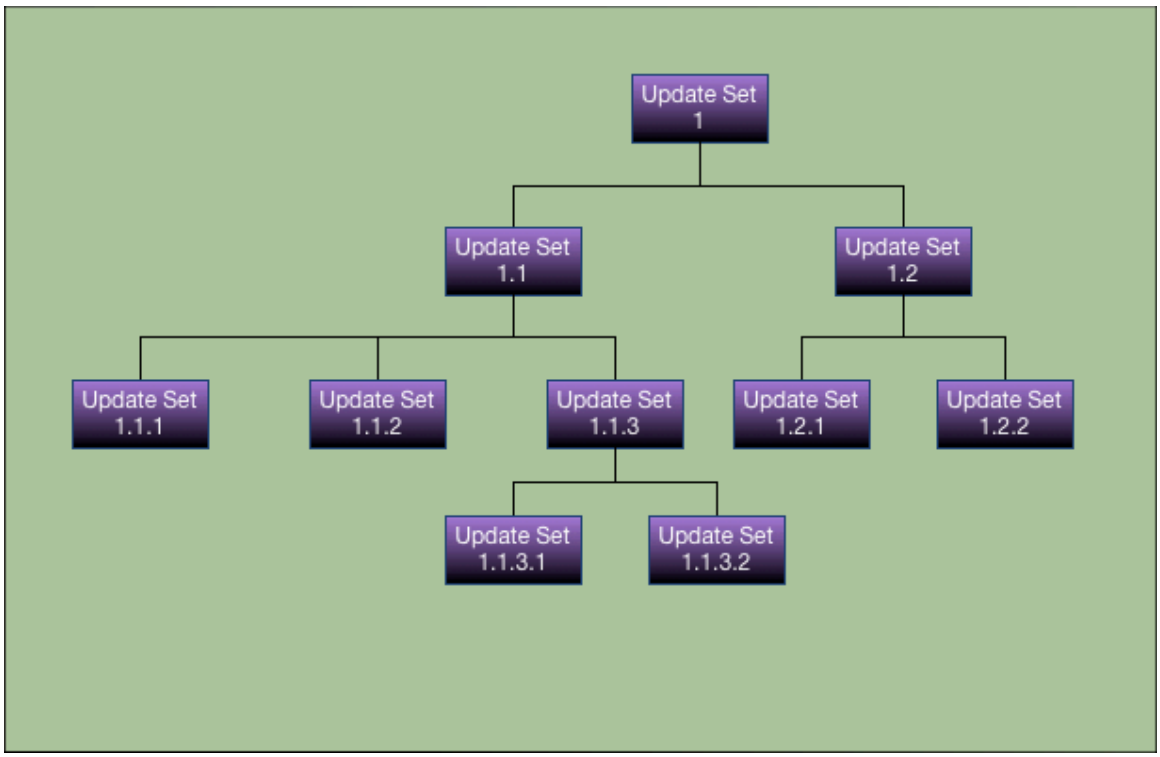


Diagram of batched update set hierarchy



Create a batch update set

You include an update set in a batch by specifying another update set as its parent.

Before you begin

Role required: admin

Adding a WIP update set to a completed batch resets the batch base to WIP.

Procedure

1. Navigate to **All** > **System Update Sets** > **Local Update Sets**.
2. Select the record for an update set that you want to include as a child in the batch.
3. On the Update Set record, navigate to the **Parent** field and select the update set to act as the parent.
4. Click **Update**.
   The system returns to the list of Update Sets. If the **Parent** column is visible, it shows the parent for the newly-created child.

Retrieve batched update sets

You retrieve a batch of update sets using the same process you would as for any individual update set.

Before you begin

Role required: admin

Procedure

To retrieve a batch of update sets, follow the same process for the batch base as you would for any individual update set.
The system will process the entire batch at once. For details, see Retrieve an update set.

Preview a batch of update sets

You can preview at once all the update sets belonging to a batch.

Before you begin

Role required: admin
You must have retrieved the update sets from the source instance.

Procedure

1. Navigate to **All** > **System Update Sets** > **Retrieved Update Sets**
2. From the list of retrieved update sets, select the batch base for the batch you want to preview.
   You cannot separately preview an update set that is a child in a batch. You must preview the entire batch by previewing the batch base. If necessary, you can remove the child update set from the batch by editing its record's **parent** field.
3. Click **Preview Update Set Batch**.
4. If the system found problems, preview the problems.
   a. Click the **Preview Problems for Batch** and resolve the problems as you normally would for any update set.
   b. When you have resolved all the problems, click **Run Preview Again for Batch**.

Commit a batch of update sets

You can commit at once all the update sets belonging to a batch.

Before you begin

Role required: admin
Before committing, you must have previewed the update sets from the source instance and resolved any collisions.

Procedure

1. Navigate to **All** > **System Update Sets** > **Retrieved Update Sets**
2. From the list of retrieved update sets, select the batch base for the batch you want to preview. You cannot separately commit an update set that is a child in a batch. You must commit the entire batch by committing the batch base. If necessary, you can remove the child update set from the batch by editing its record's **parent** field.
3. Click **Commit All Update Sets**.

Reorganize a batch of update sets

You can remove an individual update set from the batch or change its parent.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **System Update Sets** > **Local Update Sets**.
2. Select the record for an update set that you want to move or remove as a child in the batch.
3. On the update set record, navigate to the **Parent** field and select the new update set to act as the parent.
4. To remove the update set from the batch, delete any text from the **Parent** field and leave it blank.
5. Click **Update**.
   The system returns to the list of update sets. If the **Batch Base** column is visible, it shows the parent for the newly-created child.

What to do next

If the system property **glide.update_set.auto_preview** is set to **true**, the system automatically starts the preview process after the record is updated with a new parent. If this property is **false**, you must start the process manually. For more information on the preview process, see Preview a batch of update sets .

Back out batched update set

Back out a batched update set by following the backout procedure for the base update set for the batch. You can also back out any child update set independently.

The following rules apply when backing out an update set that belongs to a batch:

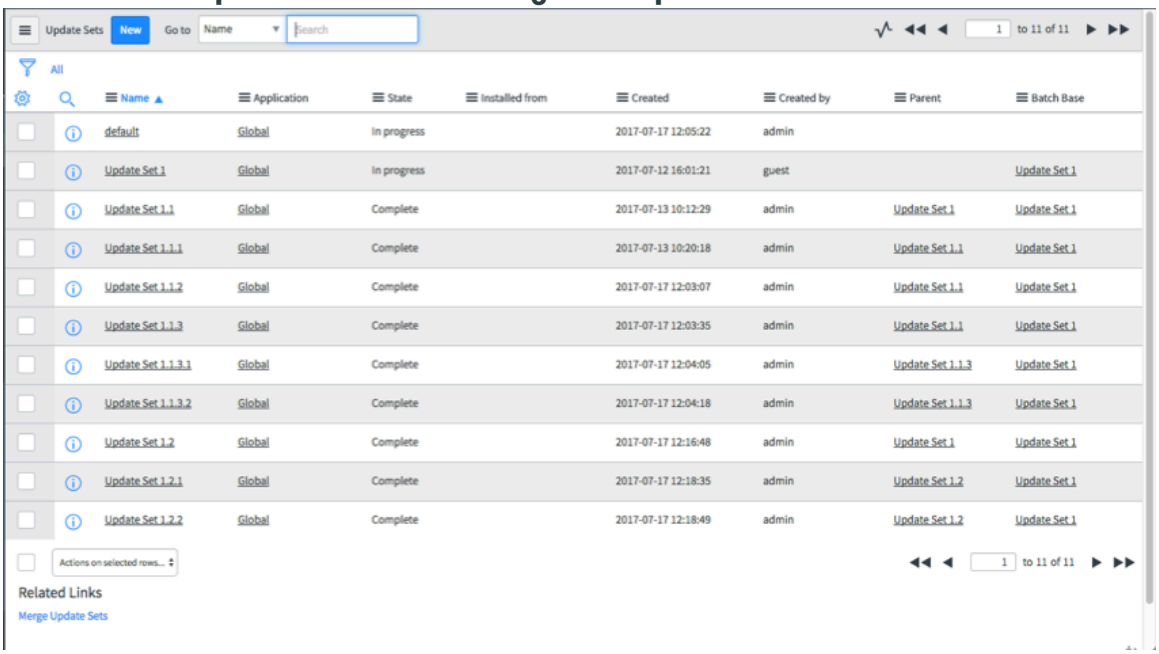
- If the update set has a parent value, the system clears the parent value and treats the update set as an independent update set, or as a new batch base if it has any children.
- The system backs out the selected update set, plus any children of the backed-out update set.

To learn more, see Back out an update set and Update set batching

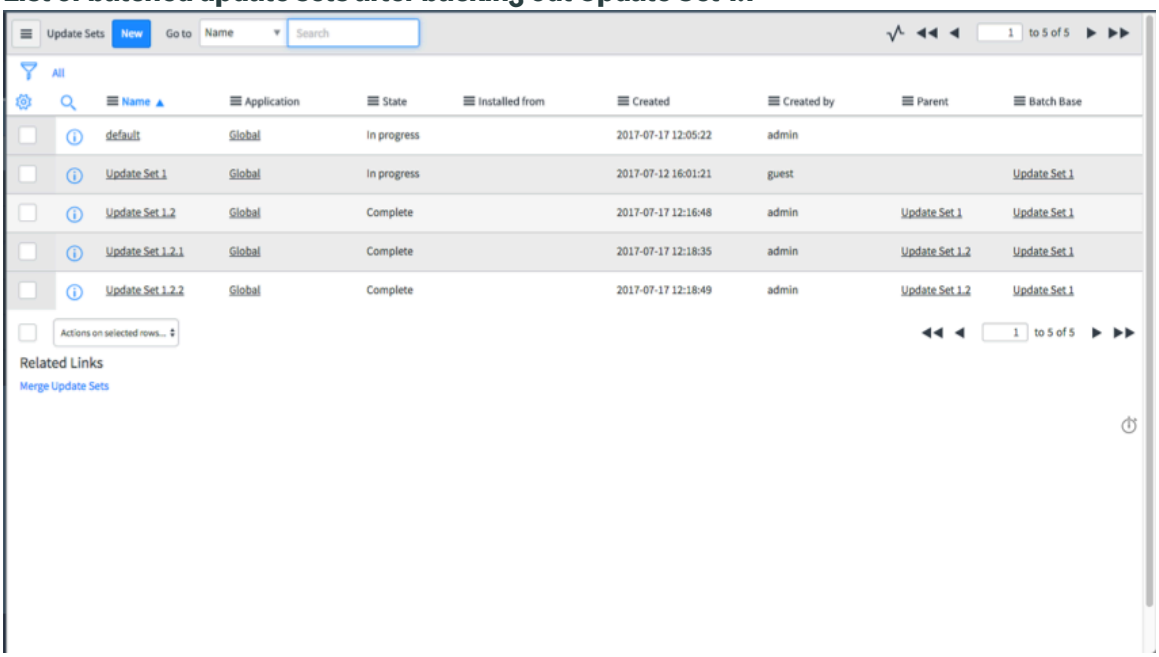
Example of backing out a batched update set

If you back out Update Set 1.1 from the batch shown in List of batched update sets before backing out an update set, the result is the batch shown in List of batched update sets after backing out Update Set 1.1.

List of batched update sets before backing out an update set

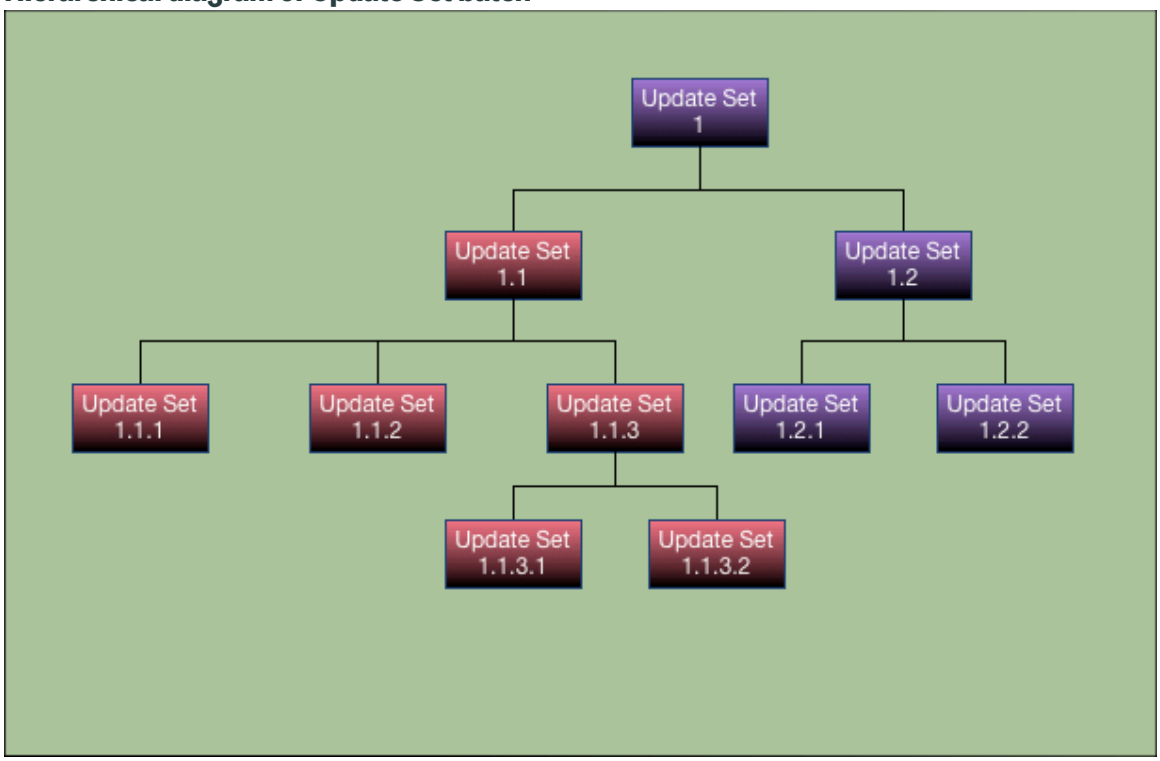


List of batched update sets after backing out Update Set 1.1



Hierarchical diagram of Update Set batch shows the hierarchy both before and after the back out. The red boxes show the update sets the system backs out if you back out Update Set 1.1.

Hierarchical diagram of Update Set batch



Administer your apps

The system offers several ways to manage applications. You must have the admin role to perform these procedures.

Legacy Application Manager

The legacy application manager provides information on how to administer your applications in your instance. For more information about installing, updating, or removing an application see Available actions.

Application Manager

A new Application Manager experience is now available. The new experience simplifies the process of managing your licensed applications by consolidating them in one convenient location, displaying all your licensed applications, plugins, installed applications, and available updates. For more information, see Application Manager.

Application sharing

Administrators can share applications that are complete and are ready for use on other instances by publishing to the application repository, publishing to the ServiceNow Store, publishing to an Update Set, or pushing to team development instances. For more information, see Application sharing.

Application sharing

Administrators can share applications that are complete and are ready for use on other instances.

Application developers can share applications using one of the following methods.

Application sharing methods

Sharing method	Makes available to	Typical use case
Publish to the application repository	All instances assigned to the same company	Transfer an application to a test or production environment.
Publish to the ServiceNow Store	All ServiceNow customers	Share or sell applications to other companies.
Publish to an Update Set	Any instance with access to the Update Set file	Save a version of an application for compliance or backup reasons.
Push to team development instances	Other instances in the team development environment	Push developer changes to the parent instance.

Note:

"Tracking schema": Deleting a table or a column in a scoped application is enabled by default for freshly zBooted instances. This is done by having the system property `com.glide.apps.include_my_schema` set to "true".

For upgraded instances, if you have no custom applications installed or in development, "tracking schema deletes" is enabled by default. Otherwise the property is set to "false" so that customers get the same experience for schema deletes in their applications as in previous releases before Paris. To learn more see the New York and Scoped Applications New Features article on the ServiceNow Community site.

Custom licensing for ISV applications

For applications that you are sharing, you can create a definition to track usage metrics on your application. For more information, see Custom licensing for ISV applications.

ServiceNow application repository

After you develop and test a custom application, you can make the application available to company instances by publishing it to the ServiceNow application repository.

The ServiceNow application repository is a central repository for all scoped applications that are published by all ServiceNow customers.

Note:
The ServiceNow application repository isn't available for self-hosted (on premise) customers.

The application repository enables ServiceNow customers to upload and distribute applications between their instances. When you access the application repository, you can see and manage only the applications that are published by your own organization. You can't see or manage applications that are published by other organizations.

After you have designed, developed, and successfully tested a custom application, you can publish your application to the ServiceNow application repository to share it to other instances in your company.

Entitlements

An entitlement refers to permission given to an instance to install a scoped application from the application repository. An instance must be entitled to an application in order for you to be able to install the application on the instance.

By default, after you publish an application to the application repository, all your company instances are entitled to the application automatically. To limit which company instances are entitled to the application, access the application repository by going to https://apprepo.service-now.com, and then change the entitlement type for the application. You can also entitle an instance again if the application entitlement has already been removed. For more information, see Manage application entitlements from the application repository.

Using the application repository

You can access the application repository by going to https://apprepo.service-now.com.

Note:
Customers using the Federal Store must log in to HIWave and then select the Federal Store option at the bottom of the page. Then, under your company name, select My Repository to proceed.

After you publish an application to the application repository, you can:

- Install an application from the application repository
- Manage application entitlements from the application repository
- Delete an application from the application repository
- Release a scope from the application repository

Manage applications

Learn how to manage the applications you publish to the ServiceNow application repository.

After you have designed, developed, and successfully tested a custom application, you can publish your application to the ServiceNow application repository to share it to other instances in your company. You can see, manage and customize only the applications that are published by your own organization or scoped applications that are installed via plugins using the ServiceNow Application Repository. You can't see or manage, or customize applications that are published by other organizations.

Note:
Do not combine the usage of both Update Sets and the Application Repository for scoped app development. This will result in numerous issues, including skipped changes, commit errors, and more. Once you have installed an application from the Application Repository, you must continue to develop and publish to the Application Repository for all future development. If you decide to develop an application using update sets, you must continue to use that method exclusively.

Overview

The Application Repository is part of the ServiceNow Store. ServiceNow customer instances are connected to a particular Store instance (for example, Commercial or Regulated Market). Customer applications published to the Application Repository are stored in a multi-tenant artifact storage, but the storage is designed so that customers cannot see or access other customers' published applications.

Application Repository has a limit of 20 versions of a given application. If your team is generating more builds or versions of your application than 20 at a time, the oldest is deleted.

Application Repository has a file size limit for applications at 1GB. Applications over this file size are not accepted during the publishing process.

Manage application entitlements from the application repository

Add or remove application entitlements to limit which instances the application can be installed on.

Before you begin

You can manage only the applications that you've published to the application repository. For more information, see Publish an application to the application repository.

Role required: none

About this task

By default, after you publish an application to the application repository, all your company instances are entitled to the application automatically. To limit which company instances are entitled to the application, access the application repository by going to https://apprepo.service-now.com, and then change the entitlement type for the application. You can also entitle an instance again if the application entitlement has already been removed.

Procedure

1. Go to https://apprepo.service-now.com.
2. Log in using your HI credentials.

> **Note:**
> Customers using the Federal Store must log in to HIWave and then select the Federal Store option at the bottom of the page. Then, under your company name, select My Repository to proceed.

3. Next to the application listing, click **Select Action** and then click **Manage Entitlements**.
4. Choose an entitlement type for your application.
5. Click **OK**.

Release a scope from the application repository

Release a scope from the application repository so that the scope can be used to create new scoped applications.

Before you begin

You can release an application scope only if the scope isn't being used by any application. If your scope is being used by an application, follow the steps in Delete an application from the application repository before releasing the scope.

Role required: Primary customer admin of the account

About this task

The application repository stores the scopes of all your custom applications. You cannot create a new application using a scope that is being stored in the application repository. To release a scope means to remove the scope from the application repository so that you can create new applications using the scope.

Procedure

1. Go to https://apprepo.service-now.com.
2. Log in using your HI credentials.

> **Note:**
> Customers using the Federal Store must log in to HIWave and then select the Federal Store option at the bottom of the page. Then, under your company name, select My Repository to proceed.

3. Open the Scopes tab.
4. Next to the scope listing, click the trash icon ( ).

Publish an application to the application repository

Publish a custom application to the application repository so that it can be installed on other instances in your organization.

Before you begin

To enable a developer to publish an application to the application repository, delegate the Publish to App Repo permission to the developer. For more information, see Delegate development and deployment permissions to personnel.

Role required: admin or delegated_developer with Publish To App Repo permission enabled

Procedure

1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. Open the In Development tab.
3. Open the application record that you want to publish to the application repository.
4. Select the **Publish to My Application Repository** related link.
5. Select **Submit**.

What to do next

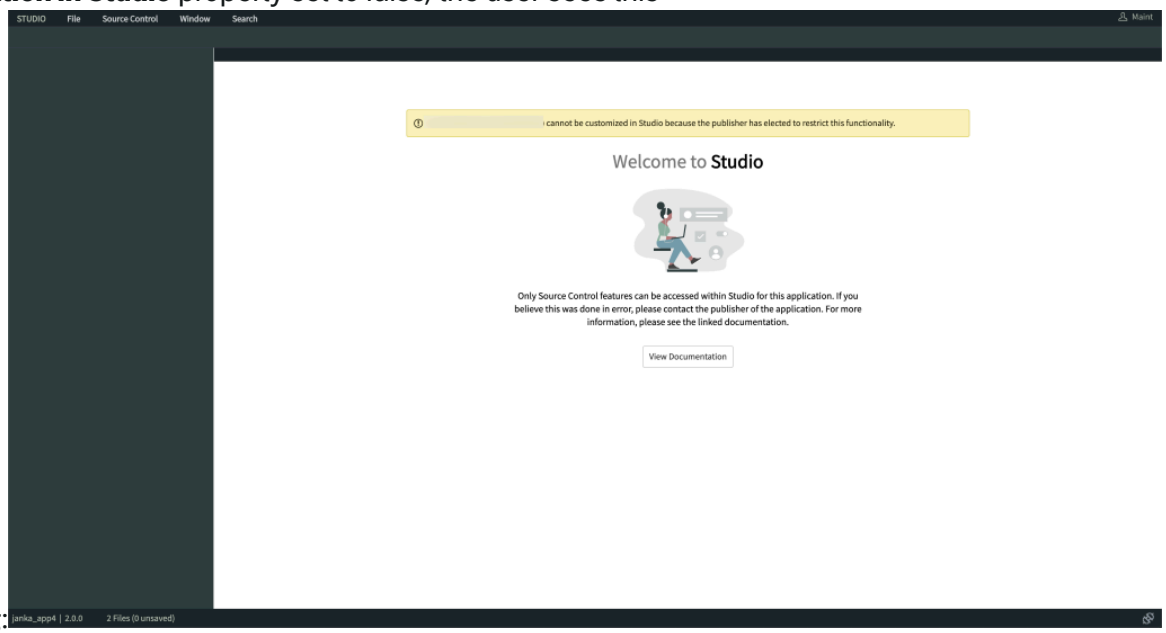
Install the application on company instances so that your organization can start using it. For more information, see Install an application from the application repository.

By default, after you publish an application to the application repository, all your company instances are entitled to the application automatically. To limit which company instances are entitled to the application, access the application repository by going to https://apprepo.service-now.com, and then change the entitlement type for the application.

Note:

The **Can Edit Application in Studio** property defaults to true for new applications, but you can set it false before publishing.

In Studio, when an application customization has the **Can Edit Application in Studio** property set to false, the user sees this warning:



For more information, see Manage application entitlements from the application repository and Legacy - Access ServiceNow Studio.

Install an application from the application repository

Install your application on an instance so that employees can start using the application that you developed.

Before you begin

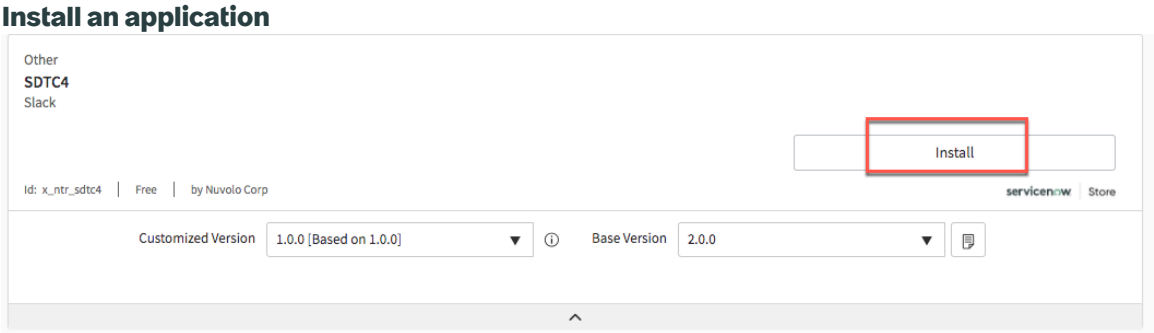
- Publish an application to the application repository.
- Check the entitlement type of the application to ensure that your instance is entitled to the application. For more information, see Manage application entitlements from the application repository.

Role required: admin

Procedure

1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. Find the application.
3. Next to the application listing, select a version to install.

4. Click **Install**.



Delete an application from the application repository

Delete an application from the application repository so that it's no longer available to your company instances.

Before you begin

- Publish an application to the application repository
- You can delete an application only if the application is not installed on any of your company instances. Uninstall the application on all your instances before deleting it from the application repository.

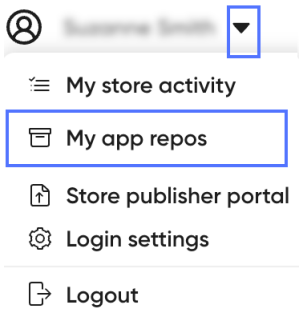
Role required: Primary customer admin of the account

Procedure

1. Go to https://apprepo.service-now.com.
2. Log in using your HI credentials.

> **Note:**
> Customers using the Federal Store must log in to HIWave and then select the Federal Store option at the bottom of the page. Then, under your company name, select My Repository to proceed.

3. Select the expand arrow next to your name and select **My app repos**.



4. Next to the application listing, click **Select Action** and then click **Flag for Deletion**.
5. On the confirmation window, click **Yes**.
   After you flag an application for deletion, the application is deleted automatically after 90 days.
6. To delete the application immediately:
   a. Open the Flagged Apps tab.
   b. Next to the application listing, click **Select Action** and then click **Delete Immediately**.

Manage customizations to applications

You can manage your company's customizations for applications that belong to other organizations or a scoped ServiceNow plug-in.

Before you begin

Role required: admin

The entitlements of your company's application-customizations are controlled solely by the entitlements of the respective store or vendor-released application. There are no separate entitlements for application-customizations, so you can't customize instances that don't have an entitlement to a base store application.

The development instance that you are using to create application-customizations should have its respective store or plugin application entitled and installed.

Note:
ISV partners who want to customize their own applications must first publish to the ServiceNow Store and install on an instance before they can use application-customizations via the Application Repository. This is the case even though all of the records are in the same scope. Installing on the instance enables ISV partners to publish a different set of capabilities for ServiceNow Store applications versus extensions used internally for their own use.

About this task

You cannot change the actual applications that belong to other organizations (purchased from the Store) or the global plugins provided in the ServiceNow base system. But you can create and manage your company's customizations for store applications or scoped applications that are installed via plugins, using the ServiceNow Application Repository.

There are two ways to create application-customizations.

- Create from ServiceNow Studio
- Create from the navigation pane

Important:
An App Customization package is intended to be authoritative and a full representation of the desired changes or customizations of the application that you're building on top of. Local customizations (typically tied to the sys_update_xml) are not honored because of this. If a specific field must be persisted (for example a MID Server) – consider using the Loader exempt attribute, see Altering tables and fields using dictionary attributes. If a specific property must be set – consider making it 'private', see Available system properties.

Procedure

1. Use these steps to create application-customizations from ServiceNow Studio.

   a. Open the Store application or the application installed via plugin from ServiceNow Studio. To learn more, see Legacy - Access ServiceNow Studio.
   The application opens in Studio.

   b. Make application changes from Studio.

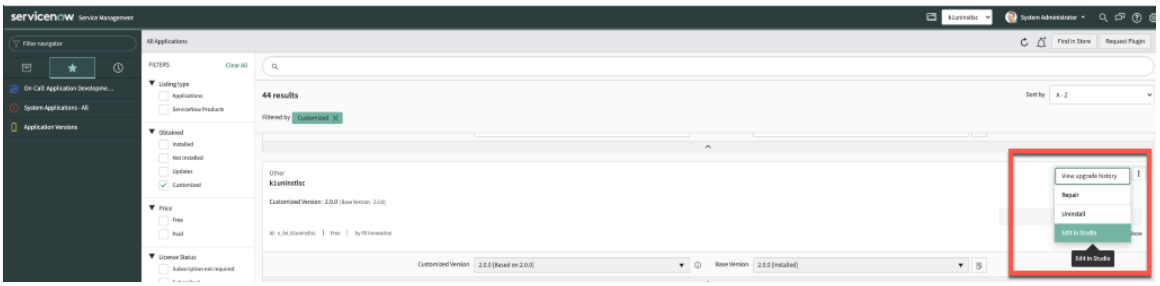
2. Use these steps to create application-customizations from the navigation pane.

a. Navigate to **System Applications >All**.

b. Search for the installed application you want to customize.

> **Note:**
> You can open the application in either Studio or select it as the current working application in the Store application record.

c. To open this application in Studio, select the stack menu, and then select **Edit in Studio**.



The application opens in Studio. You can make application changes from Studio.

d. To edit the Store application record, select the name of the application.

e. In the Related Links section of the Store application record, select **Switch to this Application**.
The application becomes the current working application in the Application picker. You can make application changes from the Store application record.

Related topics

Manage application entitlements from the application repository

Publish customizations to an application repository

After you have designed, developed, and successfully tested your customizations to a store application or a scoped plugin, you can publish it to the ServiceNow application repository to share it to other instances in your company.

Before you begin

Role required: admin, or delegated_developer with Publish To App Repo permission enabled

Only one customization can be created per vendor or customer code registered on the store and app repo.

To enable a developer to publish an application to the application repository, delegate the Publish to App Repo permission to the developer. For more information, see Delegate development and deployment permissions to personnel.

Procedure

1. Navigate to **System Applications** > **All Available Applications** > **All**.
2. Find the application by typing its name in the Filter Navigation box.

3. Open the application record for which you want to publish the customization to the application repository.
4. Click the **Publish Customizations to My App Repo** related link or in ServiceNow Studio, use the **File Publish** menu item.
5. Click **Submit**.

Convert custom applications to upgrade from the application repository

When your applications are placed in the Custom Applications table [sys_app], you can't upgrade them directly through the Application Repository. This procedure helps you do a one-time conversion when you want to migrate deploying your applications using the Application Repository.

Before you begin

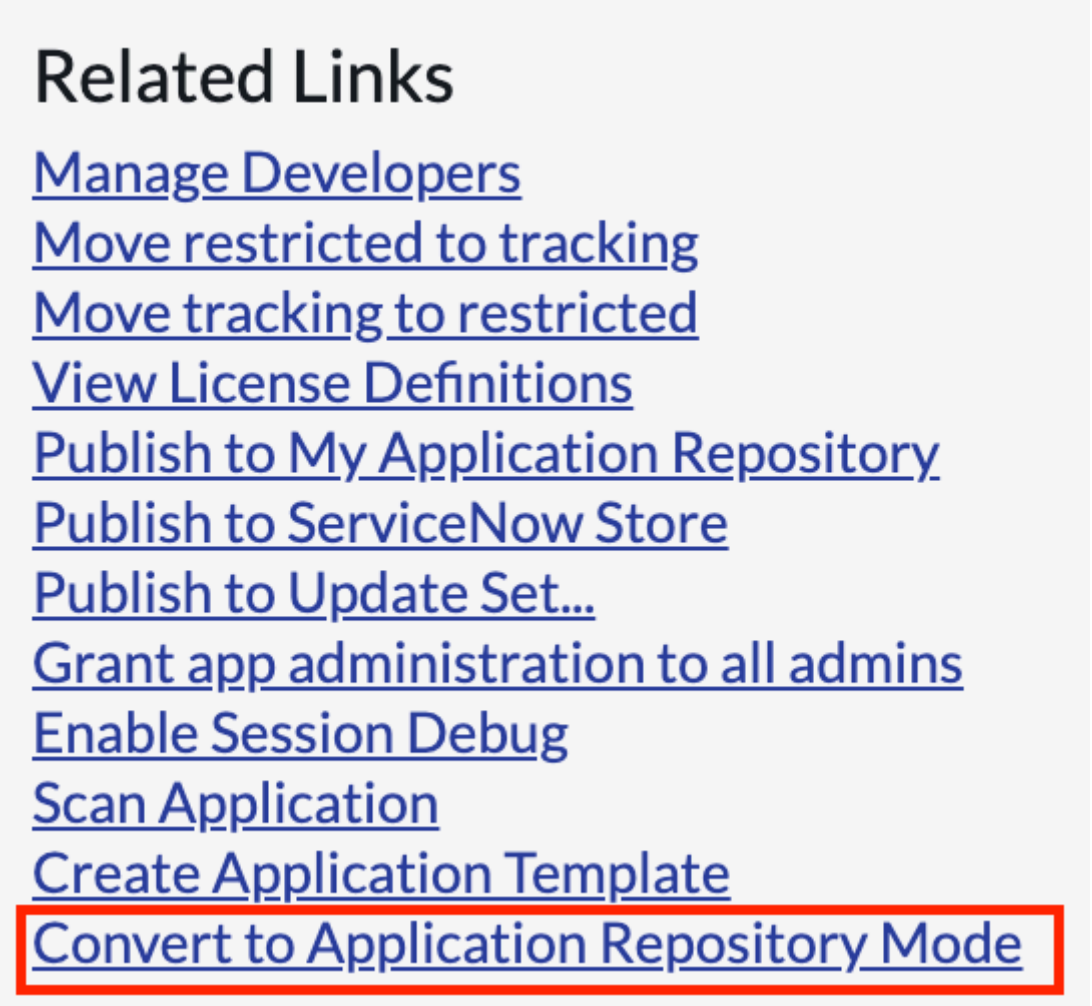
Role required: admin

When you start with Update Sets, all your applications in all your instances are placed in the Custom Applications table [sys_app]. However, if you decide to use the recommended ServiceNow® Continuous Integration/Continuous Delivery API/Spoke guidance for using the Application Repository in your CI/CD pipelines, the system doesn't allow the upgrades because the applications already exist in Custom Applications table. They are moving to the ServiceNow Store table [sys_store_app] so they can be installed from the application repository, which is why they can no longer be developed on that instance.

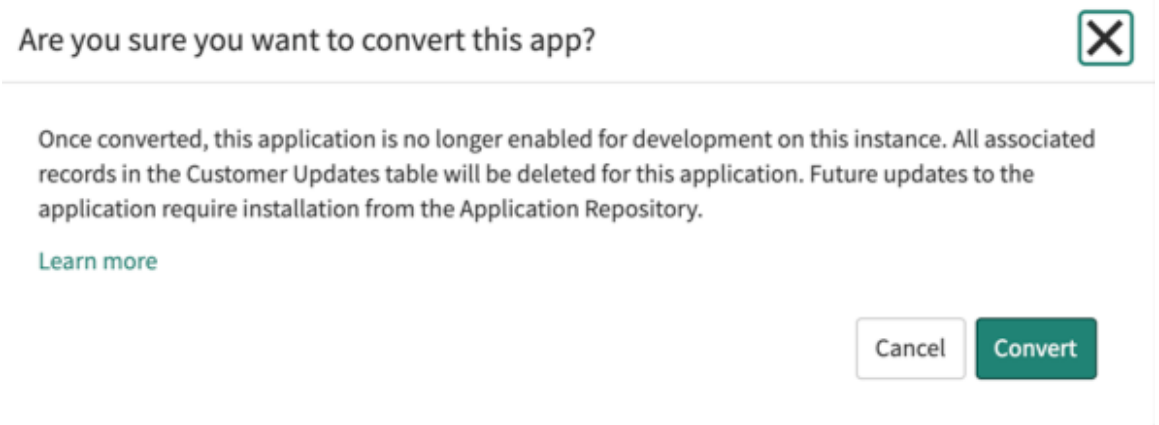
After you convert the application, it is no longer enabled for development on the instance where the conversion is performed. For a scoped custom application, all associated records in the Customer Updates table are deleted for this application.

Procedure

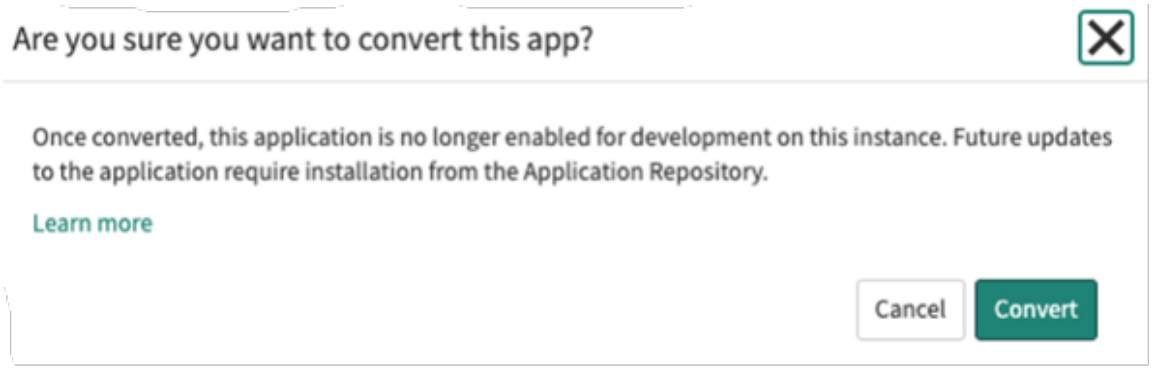
1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. Open the **In Development** tab.
3. Click the name of the application record that you want to convert.
4. Select the **Convert to Application Repository Mode** related link.



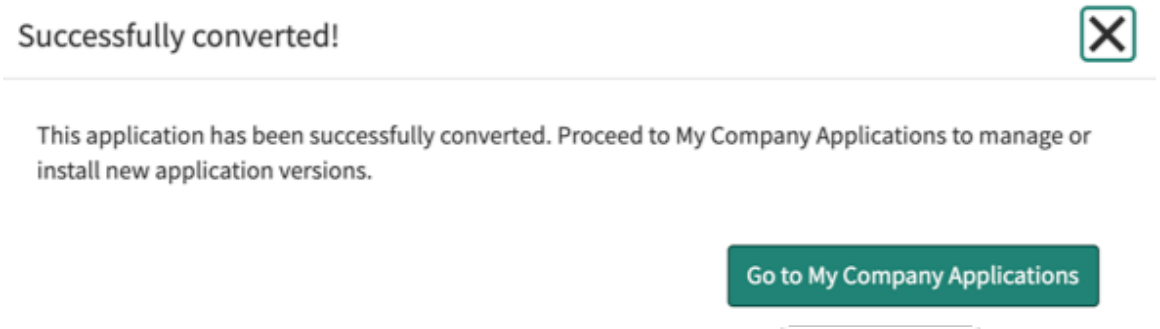
5. Select **Convert**.
   - In scoped custom applications, the following message displays.



   - In global custom applications, the following message displays.



- After the successful conversion, the following message displays.



What to do next

Go to **System Applications->My Company Applications** and update the application using the software from the application repository. See Install an application from the application repository to learn more.

Convert your installed applications to development mode

Convert the installed applications that your company owns to development mode after you install an application onto a non-production instance to use for development or clone a production instance into a non-production instance for development. With this conversion, you enable newer versions of the application to be created, committed, and published.

Before you begin

You must set the instance scope to the desired application scope. For example, if converting Agent Workspace, the application scope is set to Agent Workspace.

Role required: admin

About this task

The system properties feature [sn_appclient.store_app_convert_enabled] controls this action, which is enabled by default. You can disable this feature by setting sn_appclient.store_app_convert_enabled to false.

You can convert only the applications that your company owns. The application scope must match a key that is set in sn_appauthor.all_company_keys.

Note:
Installed global applications are eligible for conversion.

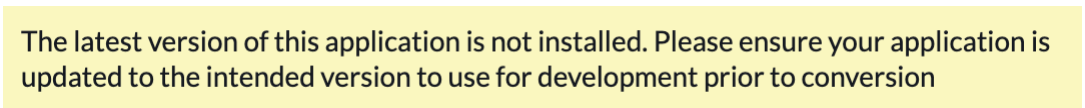
After you convert the application, it can't receive updates from the application repository on this instance. Ensure that the version of the application that is installed on the instance is the version that you intend to develop as a new version.

Note:
Applications that are installed with demo data lose their reference to this data on installation. If an application is converted, you must re-create any sys_metadata_link records to the demo data before you link the application to the source control or publish a new version.

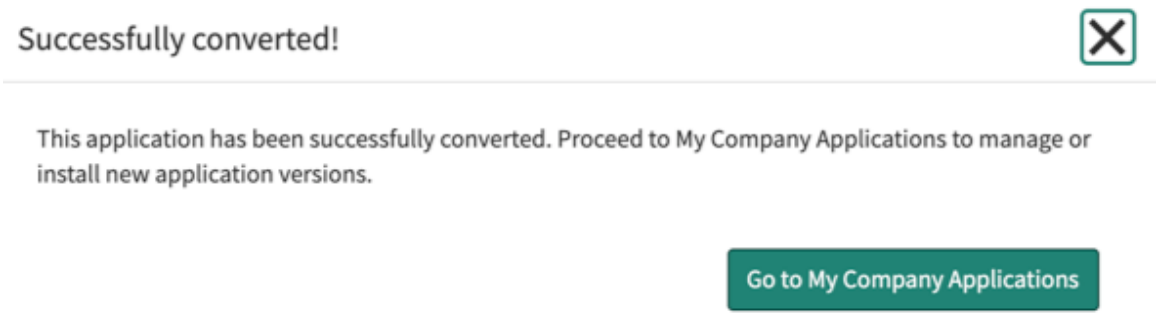
Procedure

1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. In the **Installed** tab, click the name of the application record that you want to convert.
3. Select the Convert to Development Mode related link.
4. Select **Convert**.
   - If the installed version of the application is equal to the latest version that is published to the Application Repository, you see the following dialog:

   - If the installed version of the application is lower than the latest version that is published to the Application Repository, you see the following warning:



     - If an instance is connected to a self-hosted application repository, you may see this warning regardless of the availability of a higher version in the repository. Check your repository to ensure that the development version that you require is installed in this case.
   - After the successful conversion, the following message is displayed:



     - Close the window to be redirected to the development application form.

Install customizations from an application repository

Install your customization to a store application on an instance.

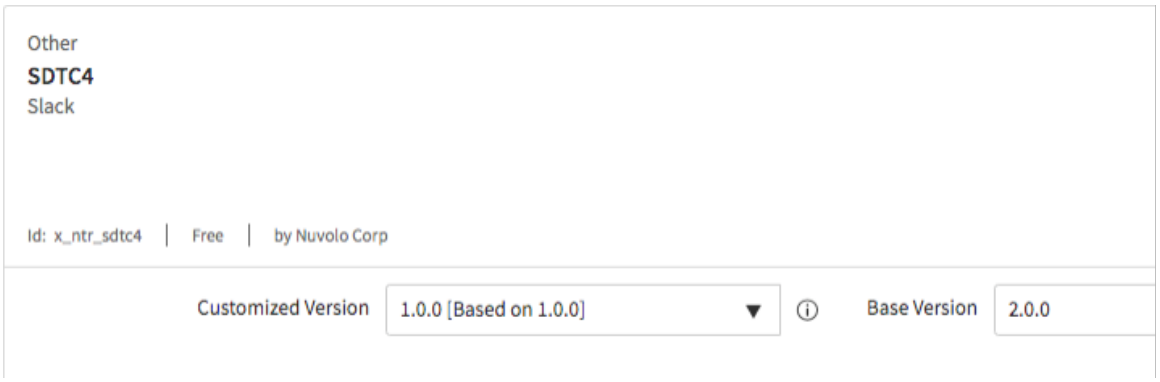
Before you begin

Role required: admin

Publish a customization of a store application to the application repository. Check the entitlement type of the application to ensure that your instance is entitled to the application.

Procedure

1. Navigate to **System Applications** > **All Available Applications** > **All**.
2. Find the application.
3. Select a customization version to install.



Note:
For scoped applications installed via plugins, the base system version is the ServiceNow release version that you are currently using (Paris, Quebec, Rome, and so on).

4. Click **Install**.

Preserve applications and customizations in development during a system clone

Manually preserve a copy of each application and customization that you currently have in development before you can clone the application version to the target (development) instance.

Before you begin

Ensure that you have write access to the application record.

Ensure that you have access to a source control repository.

Role required: admin

About this task

The cloning process does not preserve version differences for applications and app customizations in development. Instead, the system clones only the copies of the application and app customization versions that are installed on the source instance onto the target instance. If the target instance had a development version of the same application, the application is editable after the clone, but is at the version that was installed on the source instance. If the application was missing from the source instance, the cloning process deletes the application from the target instance.

Procedure

1. To preserve the application on the clone target instance, do one of these actions:

Version differences between instances

Application version state	Action to take
The application version on the clone target instance is different than the source instance version.	Export each application from the clone target instance. The choices include: ○ Link each application to a source control repository. **Note:** If the application is already linked to a source control repository, commit the latest version to it. ○ Publish each application to an update set.
The application is available only on the clone target instance.	
The application version on the clone target instance is the same as the source instance.	None. The system clone process copies this application version onto the target instance during the clone.

2. Request a system clone of the source instance to the target instance.

Example

For example, clone your production instance over your development instance.

3. After the clone process finishes, log in to the clone target instance.

4. **Note:**
   If source control is linked, then post clone, the platform will automatically retrieve applications and customized applications. If this is disabled via `glide.source_control.post_clone_import_enabled` you will need to manually retrieve by doing the following.

   If you saved each application to a source control repository, use one of these actions to retrieve them from the source control repository:

   Note:
   For what to expect after application customization post clone, see Results post cloning for application customizations.

Retrieve applications from a source control repository

Application installation state	Action to take on clone target
The application and customization were previously installed on the source instance.	Apply remote changes from the source control repository.
The application was never installed on the source instance.	Delete the repository configuration (sys_repo_config) and import the customization from the source control repository.

Remote changes after clone

Field	Description
glide.source_control.post_clone_import_enabled	To disable the apply remote changes automation, set to **False**. The default is **True**.
glide.source_control.post_clone_import_delay_time_sec	To provide a delay time, which will delay processing of the queue, provide a value. The default is zero.
glide.source_control.post_clone_import_pause_refresh_time_sec	To provide a time interval in which the refresh repository job will not run, provide a value. The default is three hours (10800).

5. If you saved each application to an update set, do one of these actions to retrieve them from the update set:

Retrieve applications from an update set

Application installation state	Action to take on clone target
The application was previously installed on the source instance.	**a.** Delete the application version that was cloned from the source instance. **b.** Load the update set that contains the current application version.
The application was never installed on the source instance.	Load the update set that contains the current application version.

Result
The applications previously in development are available for further development on the clone target instance.

Example: Preserve the Marketing Events application

Let's say that your company previously created version 1.0 of a custom application called Marketing Events. You have already published version 1.0 of the Marketing Events application to the application repository and installed it on your production instance.

Over time, users have submitted enhancement requests for the application, and you decide to develop version 2.0 of the Marketing Events application on a non-production instance to address these requests. As development nears completion, you want to update your non-production instance to the latest copy of production for some comprehensive testing.

Because you previously used a source control integration to develop version 1.0 of the Marketing Events application, you have already linked the Marketing Events application to a source control repository. You commit version 2.0 of the Marketing Events application to the source control repository.

You schedule a clone of the production instance over the development instance. After completion, you log in to the development instance and see that it has version 1.0 of the Marketing Events application, because that was the version installed on the source instance.

Because the application was already installed on the source instance, you apply the remote changes from the source control repository to receive the latest application version. The development instance now has version 2.0 of the Marketing Events application and is available for further development and testing.

Results post cloning for application customizations

The results to expect post cloning for application customization display the expected behaviors based on the state of the application, and the actions to recover your application customizations.

The following are the expectations based on the state of the application customizations, and the actions to recover your customizations.

Application Customization expectations post cloning

State of the App Customization on Source Instance	State of the App Customization in the Target (Development) instance pre-clone	State of the App Customization in Target (Development) instance post clone	Action to take to recover App Customization in the Target (development) instance post clone
Base application is not installed	Base application version 1.0 and App Customization version 1.0	Base application is not installed	Reinstall Base application and App Customization
Base application not installed	Base application version 1.0 and App Customization in Source Control	Base application is not installed, and the sys_repo_config has an empty app field.	Remove the repository configuration (sys_repo_config) record (the app field has been emptied) and import again from Source Control and install Base application.
Base application version 1.0	Base application version 1.0 App Customization version 1.0	Base application version 1.0	Reinstall App Customization
Base application version 1.0	Base application version 1.0 and App Customization in Source Control	Base application version 1.0, repo_config is retained and the App Customization version displays as none in the all apps page.	Apply the remote changes from the Git repository.
Base application version 1.0 and App	Base application version 2.0 and App	Base application version 1.0 and App	Reinstall App Customization

Application Customization expectations post cloning (continued)

State of the App Customization on Source Instance	State of the App Customization in the Target (Development) instance pre-clone	State of the App Customization in Target (Development) instance post clone	Action to take to recover App Customization in the Target (development) instance post clone
Customization version 1.0	Customization version 2.0	Customization version 1.0	version 2.0 and Base application version 2.0
Base application version 1.0 and App Customization version 1.0	Base application version 2.0 and App Customization version 2.0 in Source Control	Base application version 1.0 and App Customization version 1.0, sys_reconfig is retained	Apply the remote changes from the Git repository.
Base application version 2.0 and App Customization version 2.0	Base application version 1.0 and App Customization version 1.0	Base application version 2.0 and App Customization version 2.0 is installed.	Both Base and App Customization are at the latest versions.
Base application version 2.0 and App Customization version 2.0	Base application version 1.0 and App Customization version 1.0 in Source Control	Base application version 2.0 and App Customization version 2.0 is installed with repo_config retained, so the branch version is on 1.0	Both Base and App Customization are at the latest versions.

Application Repository for self-hosted, air-gapped customers

This application feature enables self-hosted customers to use their own internal application repository just as cloud customers do. These customers don't install the Application Repository via the ServiceNow® Store; they use an alternative method.

Overview

Customers whose instances are hosted on the ServiceNow® cloud have access to an application repository by default via the public ServiceNow® Store. However, self-hosted and air-gapped customers don't have such access, and thus can't take advantage of advanced application development features.

> **Warning:**
> Most customers install applications through their networks from the ServiceNow® Store. This topic covers customers who aren't using networked sources.

The term "air-gapped" refers to the condition when a computer or network has no network interfaces, either wired or wireless, connected to any outside networks. When you want to move data from one source to another, you must use an alternative method. To move data between sources that aren't on a network, you must use an alternative such as the one described in this topic.

The ServiceNow® Application Repository is a scoped application that can be installed on a customer instance, and enables usage of the latest application development features inside their own private network without the need to communicate with external networks.

Recommended installation

It is recommended to install the Application Repository on a dedicated instance used only for that purpose. Installing in this way reduces the load on other instances from other transactions and improves the experience of installing other Store applications. After you have successfully installed the application, other networked client instances can publish and install application artifacts to and from this instance.

Acquiring the application

Before attempting to install the application, ServiceNow self-hosted "air-gapped" customers should reach out to their account representative. They will work with Support and your team to set up the installation process. To learn more, see the Knowledge Base article, How On-Prem customers can request a Store App.

How the setup process works

Instances are authenticated with the repository via a “pairing” process that stores instance credentials on the server-side, and needs to be done only once per instance. After an instance is “paired” with the repository, it can start interacting with the repository via its various application development features.

Installing the Application Repository store app on an air-gapped instance

Because self-hosted, air-gapped customers don't allow their instances to communicate with external networks, they don’t install the Application Repository directly from the ServiceNow® Store. These customers should use the following procedure instead.

Before you begin

Note:
Customization of the Application Repository is not supported or recommended.

Role required: You will need the maint role to install and configure, and then just the admin role once the configuration is complete.

Warning:
If your instances are already connected to the public ServiceNow® Store, you should not install the Application Repository. Doing so causes these instances to lose access. This product is intended for customers who can't connect to the public store.

Note:
Install on a dedicated ServiceNow® instance separate from your other instances.

Procedure

1. Install the application repository using the instruction in the following Knowledge Base article: How On-Prem Customers Can Request a Store App.
2. After installing, complete the steps in Configure the application repository on an air-gapped instance.

Configure the application repository on an air-gapped instance

After installing the application repository, you must configure it using the following procedure.

Before you begin

Role required: You’ll need the maint role to install and configure, and then only the admin role after the configuration is complete.

Procedure

1. Pair an instance with the application repository.
   a. Log in to the instance you want to connect to the application repository.
   b. Set the system property *sn_appclient.repository_base_url* to the URL of your application repository instance.
   For example, `http://localhost:8080/`.
   c. Clear the values of the *sn_appclient.upload_base_url* and *sn_appauthor.upload_base_url* properties.
   d. Set the value of the *sn_apprepo.credential* property in a Global scope to any non-null value, such as "secret."
      i. SSH into the instance.
      ii. Change the directory to `/root/instance/instance_<portno>/conf/overrides.d` using `cd /root/instance/instance_<portno>/conf/overrides.d`.
      iii. Open / Create the `glide.properties` file.
      iv. Add the credential property to the end and save the file `[sn_apprepo.credential=<value>]`.
      v. (Shutdown.sh/Startup.sh) Restart the glide, or run `Packages.com.glide.util.GlideProperties.loadPropertyFile(new Packages.java.io.File(gs.getProperty("glide.home.dist")+"/conf/overrides.d/glide.properties"));` in background scripts to dynamically load the properties file at runtime without restarting the instance.
      vi. Verify by printing the property in a background script `gs.info(gs.getProperty("sn_apprepo.credential"));`.
   e. Add *sn_appclient.repo_auth_name* with **sn_repo.AppRepo** as its value.
   f. Set the *glide.test_instance* property to **False** on both the application repository instance and the client instance.
   g. Set the *sn_appclient.client_calls_allowed* property to **True**.

      Note:
      A scheduled job might set this property to **False** when not connected.

   h. Set the *sn_appclient.app.install.offline* property to **False** on the client instance.
   i. Select **Submit**.
2. Log in to the instance where the application repository is installed and complete the following steps.

a. Navigate to the **core_company.list** table and ensure there is a record with the **Primary** field set to **True** or create one with any user-defined name.

> **Note:**
> The details of this record are not important.

b. Access the **sn_repo_instance.do** screen and create a new instance record for the client instance that you want to connect.

i. Ensure that **State** is set to Pairing.

ii. Enter the name of the instance that you want to connect (on the **stats.do** screen on that instance) in the **Name** field.

iii. Leave all other fields empty. They're populated automatically.

c. Repeat the previous step for any additional instances that you want to connect.

3. Log back in to the instance you used in Step 1 (the instance that you want to pair with the application repository) and navigate to the **Scripts - Background** module.

a. Select **sn_appauthor scope** from the drop-down list.

b. Run the following script: `new ConfigChecker().checkForChanges()`.

4. **Optional:** To remove the instance, navigate to the Instance record (sn_repo_instance table) and change the State to **Blocked**, which temporarily restricts access to the instance, or delete the instance.
If you need the instance again, you can change the State to **Paired** again.

> **Warning:**
> If the instance name, instance ID, or credential of a paired instance changes, it must be paired again. Manually updating any of these values on the instance record isn't recommended.

What to do next

After an instance is paired, it's fully set up to use the application repository. You can test your configuration by publishing a scoped application as described in Publish an application to the application repository. After publishing, you can verify that the app was successfully published by locating it in **All** > **Application Repository** > **Artifacts** > **Internal Apps**.

Publish an application to the ServiceNow Store

Publishing an application to the ServiceNow Store makes it available to everyone.

Before you begin

To publish an application to the ServiceNow Store:

- Create an application within a private application scope.
- Join the Technology Partner Program.
- Have the application certified.

> **Note:**
> Applications in the global scope cannot be published to the ServiceNow Store.

Role required: admin, or delegated_developer with Publish To App Store permission enabled

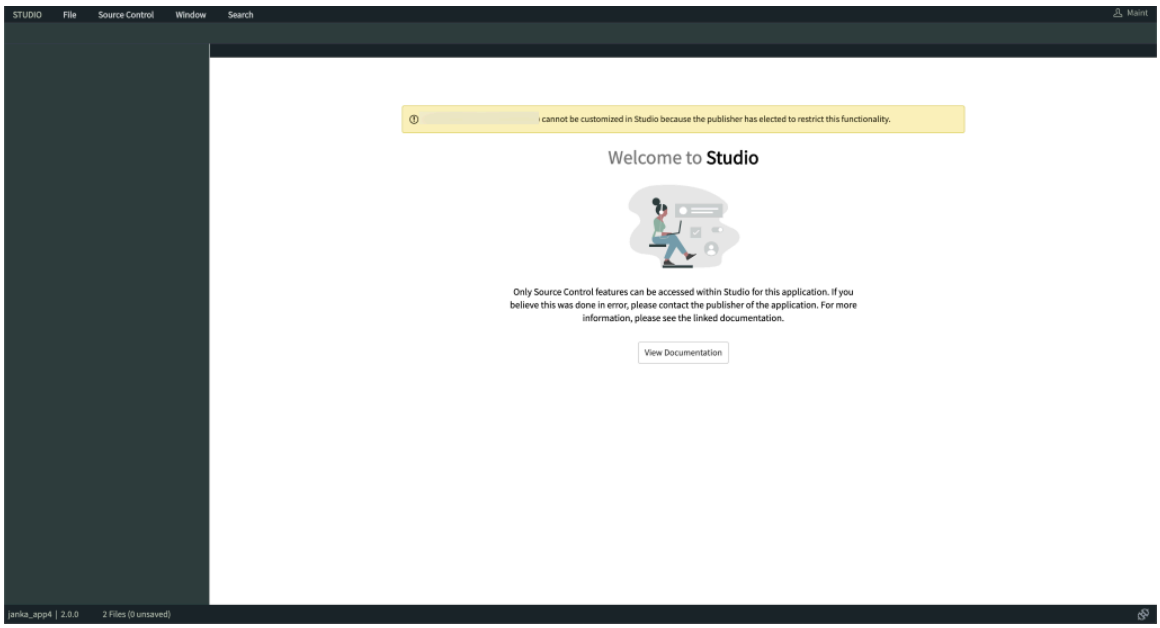
About this task

After you have met the prerequisites, you can publish the application to the ServiceNow Store

Note:

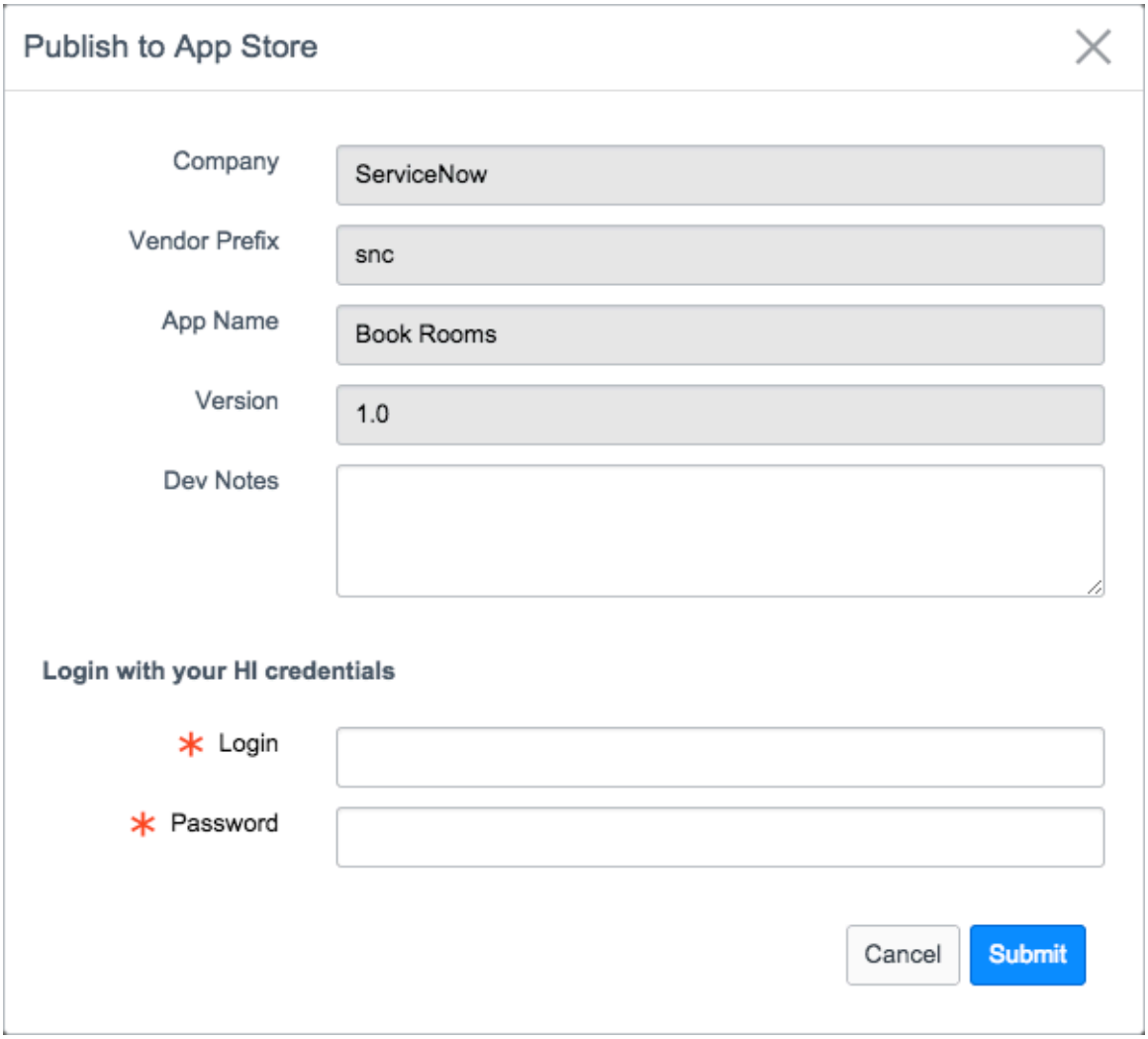
The **Can Edit Application in Studio** property defaults to true for new applications, but you can set it false before publishing.

In Studio, when an application customization has the **Can Edit Application in Studio** property set to false, the user sees this warning:



Procedure

1. Navigate to **All** > **System Applications** > **Applications**.
2. Click the **In Development** tab.
3. Open the application record you want to publish to the ServiceNow Store.
4. Click the **Publish to Store** related link.
5. **Optional:** Fill in the fields, as appropriate (see table).



6. Enter your HI credentials.
7. Click **Submit**.
   The system uploads the current version of the application to the ServiceNow Store allowing other users to download it.

Related topics

Delegate development and deployment permissions to personnel

Create application files to include sample data

Include sample records from an application data table when sharing a custom application.

About this task

The system can export selected records as application files that are included as part of the application update set when you share it. Including application files in an update set is not intended for the export and import of large numbers of records between instances. If you are trying to move data between instances, see Import from another instance instead.

The application data only includes the version of the records that existed when the records were shared. The system does not update this snapshot of the application data when the records change. Application designers can include data on a table by table basis.

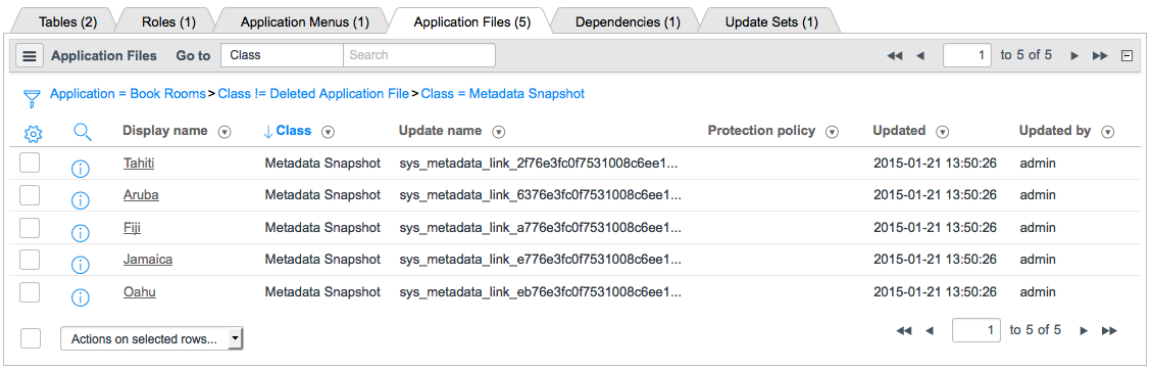
Procedure

1. Navigate to the list for an application data table.
2. Filter the list to display the records you want to include.
3. Perform the appropriate action for the list version.

4. For **Load When**, select when the application record includes application data.

Option	Description
New Install and Upgrades	Includes application data whenever the application is installed or upgraded.
New Install	Includes application data only when the application is installed.
New Install with Demo Data	Includes application data only when the application is installed with demo data.

5. Click **OK**.
   The system adds the records to the application files related list.



6. Repeat steps 1–5 for each application data table you want to include.

Publish an application to an Update Set

Publishing an application creates an update set containing the current version of all application configuration records.

Before you begin

Role required: Role required: admin, or delegated_developer with Publish To Update Set permission enabled

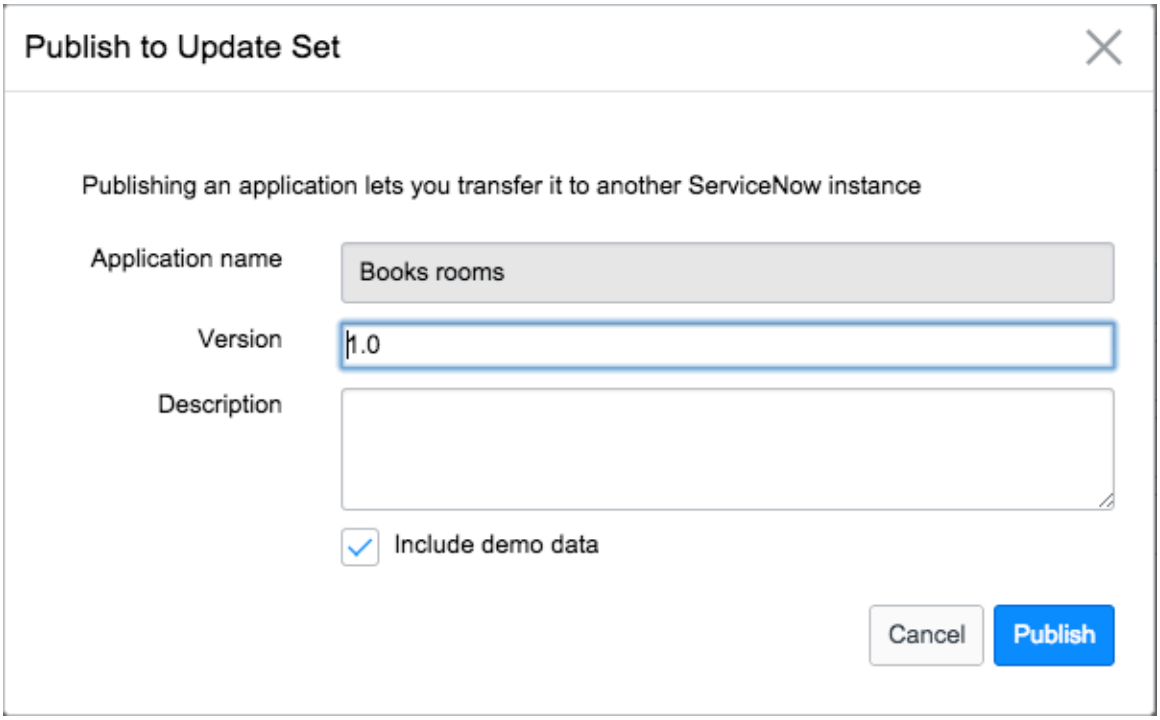
About this task

You can use this update set as a backup file for auditing purposes or to transfer the application to another instance.

Procedure

1. Navigate to **All** > **System Applications** > **My Company's Applications**.
2. Click the **In Development** tab.
3. Open the application record you want to create an update set for.
4. Click the **Publish to Update Set** related link.

5. **Optional:** Fill in the fields, as appropriate (see table).



Publish to Update Set

Field	Description
Application name	[Read-only] Displays the name of the application that you are publishing.
Version	(Optional) Enter version information to append to the Update Set name in dot-notation such as 1.2.3. The platform saves the information you enter here in the application **Version** field. The Update Set has the name <Application name> - <Version>. If you leave this field blank, the initial Update Set has the name <Application name> and subsequent Update Sets have the name <Application name> - <Sequential number>.
Description	Enter a description for the Update Set. By default, this field contains the short description of the application.
Include data	(Optional) Select the check box to include a limited number of data records from each table in the application. Use this feature to package sample data with your applications. **⚠ Warning:** (Optional) Using this feature to migrate large quantities of data records between instances can cause performance issues, as it is not intended for this purpose. To migrate data, use an instance-to-instance import. You can adjust the maximum number of data records to include with an application. See Import sets.

Field	Description
	○ If your sample data includes tables with record numbering, the current counter value is also transferred. When the update set is applied on another instance, the counter is set to the larger of the sample data or the target instance counter. ○ For translated fields, only records in English are transferred.

6. Click **Publish**.
   A new update set is created and the latest update of each application file in the application is copied into it. The update set is marked as complete.
7. Transfer the update set to another instance according to your test process.
   - Retrieve the update set from the source instance.
   - Save the update set as an XML file.
8. Run any fix scripts that are included in the application.

Related topics

Retrieve an update set

Save an update set as a local XML file

Run fix scripts

Delegate development and deployment permissions to personnel

Queued Application Operations

CICD APIs that must obtain the update **instance wide lock / mutex** to perform the requested operations are queued instead of being rejected when the update **instance wide lock / mutex** is occupied by the other operations.

Beginning in Tokyo, the CICD APIs that require to obtain the update **instance wide lock / mutex** to perform the requested operations will be queued instead of being rejected when the update **instance wide lock / mutex** is occupied by the other operations. When a CICD request is received, the corresponding CICD service constructs an application operation NowMQ (Now Message Queue) message and insert the message to the queue using NowMQ APIs. The queued messages are then polled by the scheduled job and handled one by one or in parallel if parallel processing is enabled and the operation satisfies necessary criteria.

Application Operation NowMQ Message

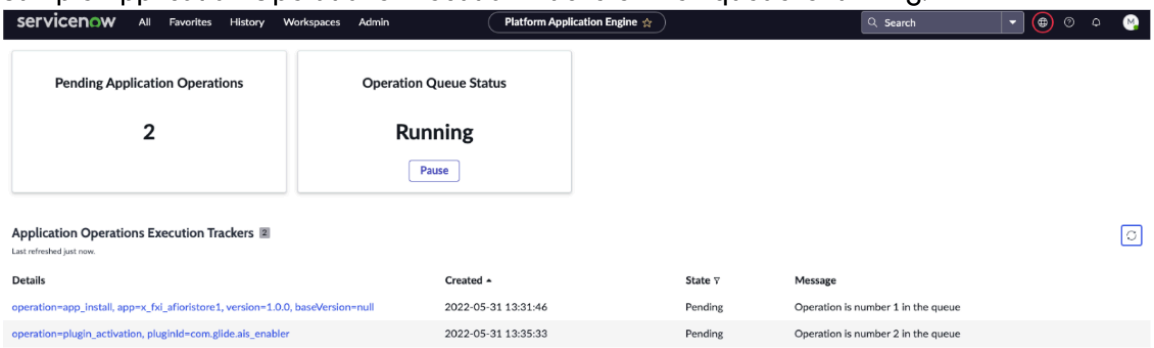
The application operation NowMQ messages have a common subject that is "sys.applifecycle.operation". The message body of the application operation NowMQ message contains a JSON object that includes the sys_id of the Execution Tracker (also refer to as the progress id returned in the CICD API response), the operation type that can be one of the following: app_install, plugin_activation, batch_install, rollback, import_app and apply_changes. It also contains the information like plugin id for plugin activation, app id or scope for application install.

Execution Tracker for Application Operation

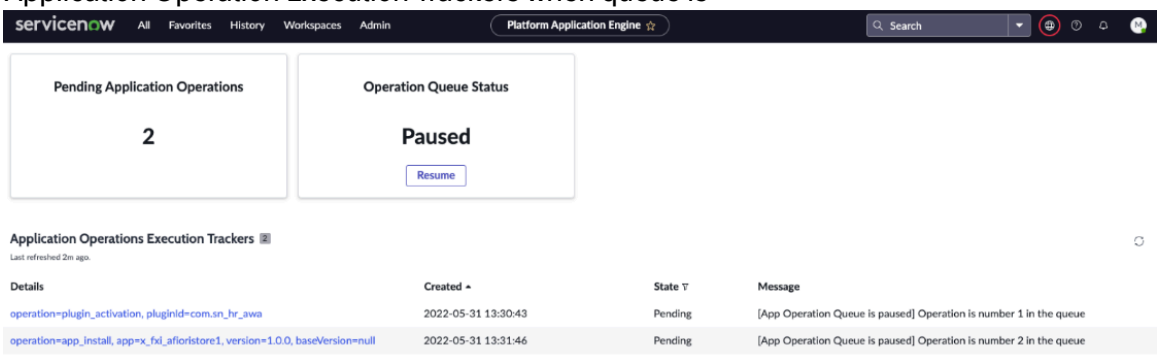
When the Application Operation NowMQ message is constructed and inserted, the Execution Tracker record for the corresponding CICD request is created and its sys_id is added into the body of the NowMQ message. The Execution Tracker is in Pending state initially. The Execution Tracker's "Details" column contains the information about the type of operation, and the

important input parameters for the CICD request. Its “Message” column contains the information about the queue position. When the queue is paused, the message is prefixed with “[App Operation Queue is paused]”.

Sample Application Operations Execution Trackers when queue is running.

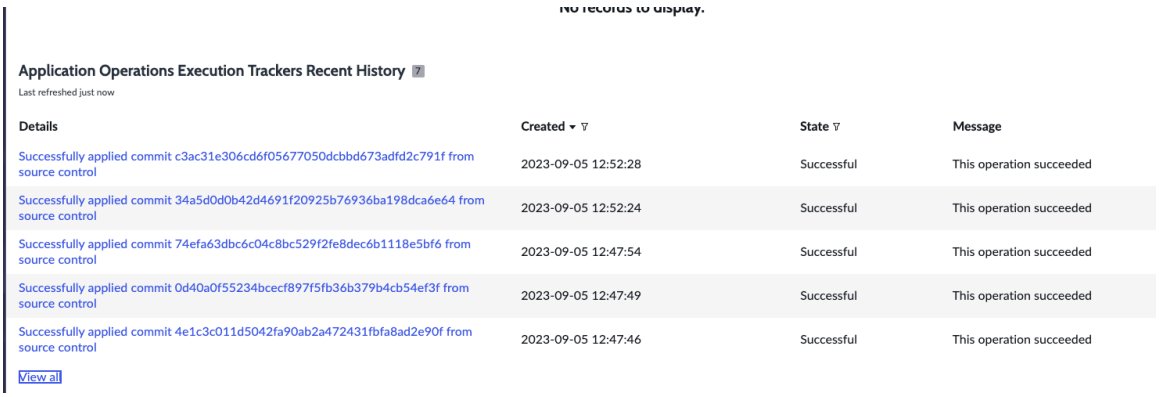


Sample Application Operation Execution Trackers when queue is paused.



Note:
Operations that are in a pending state will not display in recent history.

Sample Applications Operations Execution Trackers recent history. Recent history displays historical operations that were queued and processed in the past 24hrs.



Manage Application Operation Queue

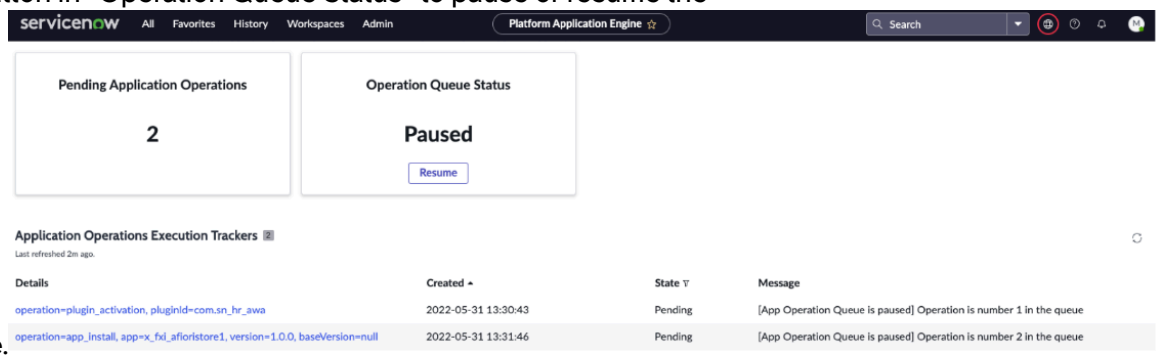
While manually installing products from All Applications is queued, manually installing an application from UI like All Applications or Application Manager isn’t queued.

Sometimes, the instance may be receiving, queueing, and handling many CICD requests which may cause the manual install from UI starving for update **instance wide lock / mutex**. When this happens, admin can temporarily pause the Application Operation Queue.

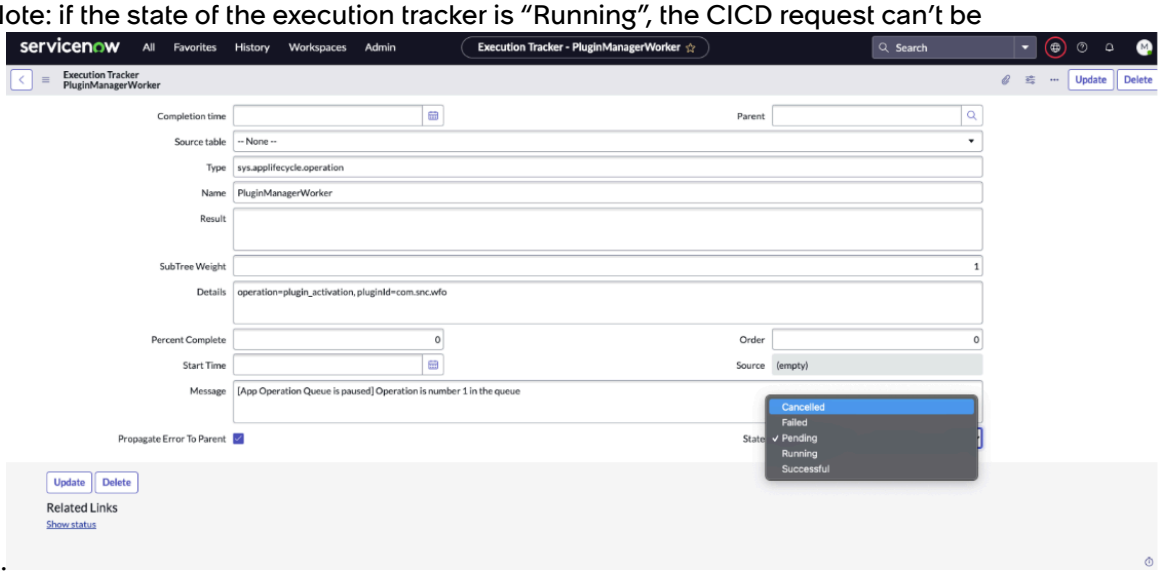
Admin can manage the Application Operation Queue through System Diagnostics->Application Operation Queue UI page. On “Operation Queue Status” panel, the admin can pause or resume the queue. Admin can also cancel the pending execution tracker, this will eventually remove the corresponding queued message from NowMQ by the App Operation Queue Health Monitor job.

Application Operation Queue UI page

Sample Application Operation Queue UI page. Admin can click the button in “Operation Queue Status” to pause or resume the queue.



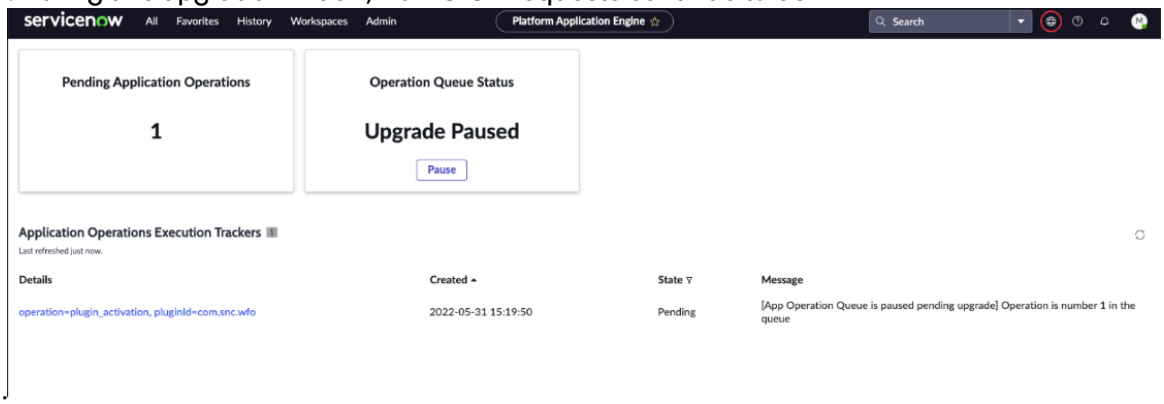
Click the “Application Operations Execution Trackers” list item, it opens the Execution Tracker form. If the queued message is pending in the queue, update the execution tracker state to “Cancelled” and save the change will cancel the corresponding queued CICD request. Note: if the state of the execution tracker is “Running”, the CICD request can’t be cancelled.



Application Operation Queue and Upgrade Window

By default, 2 hours (can be customized through sys property “com.glide.update_operation.queue_upgrade_window”) before the scheduled upgrade, Application Operation Queue stop processing queued messages.

The Application Operation Queue status is changed to “Upgrade Paused”. During this upgrade window, new CICD requests continue to be queued.



When upgrade completes, the Application Operation Queue resumes processing the queued messages automatically.

Impact to CICD Pipeline

The existing request/response contracts for the CICD APIs aren’t changed. The operation failure due to update **instance wide lock / mutex** conflicts that are observed in a pre-Tokyo release won’t be seen. The request is queued and served one by one or in parallel depending on they job type.

CICD APIs supporting queueing

The following CICD APIs are queued for processing.

- api/sn_cicd/app_repo/install
- api/sn_cicd/v1/app_repo/install
- api/sn_cicd/v2/app_repo/install
- api/sn_cicd/app_repo/rollback
- api/sn_cicd/v1/app_repo/install
- api/sn_cicd/v2/app_repo/rollback
- api/sn_cicd/sc/apply_changes
- api/sn_cicd/v1/sc/apply_changes
- api/sn_cicd/v2/sc/apply_changes
- api/sn_cicd/app/batch/install
- api/sn_cicd/sc/import
- api/sn_cicd/plugin/{plugin_id}/activate
- api/sn_cicd/plugin/{plugin_id}/rollback

Parallelize Application Installs and Plugin Activations

Starting in Tokyo, the following operations can execute in parallel with other operations.

- api/sn_cicd/app_repo/install
- api/sn_cicd/v1/app_repo/install
- api/sn_cicd/v2/app_repo/install
- api/sn_cicd/plugin/{plugin_id}/activate

All queue processing takes an **instance wide lock / mutex** and holds this mutex until any queued operation is completed. This lock is called **UpdateMutex**, and its status can be viewed in the **sys_mutex** table. During this time, operations that take this same lock (app installs, plugin activations, source control operations) are not performable via the UI. The queue can still be paused via the **Application Operation Queue** page to release the lock after currently running jobs have been completed.

Parallelization is enabled by default. It can be turned off by using the property `com.glide.update_operation.parallel_operation_enabled`, and all operations will run sequentially from the queue as in previous releases.

Limits on Parallelization

The queue processor determines if an enqueued job can run. If a job can run, it is scheduled to be picked up by an instance node at first availability. If not, it is returned to the queue, and the processor evaluates the next job in the queue for processing.

There is a limit to the maximum number of jobs that can be executed in parallel, which defaults to 2. This property can be toggled via `glide.update.app_operation_queue.parallel.max` but keep in mind that there is an upper limit of available threads to perform the install and increased parallelization takes additional memory that can slow the instance for active users.

Obtaining Resource Locks

The following plugins and applications can't be installed in parallel.

- Any two operations that share the same scope including customizations.
- Any two operations that cause schema changes.
- Any two operations containing fix scripts.

Queue processing determines if these criteria are met for enqueued operations and defers not processable jobs if necessary. The Progress ID of the job is updated to reflect if the operation is waiting for appropriate resource locks to be obtained. An executing operation maintains a list of the resources (scopes, schema, fix scripts present) in the sys_padlock table, and the insert and release of these locks can be observed in this table by the Progress ID. If a queued job is deferred due to inability to obtain necessary locks on resources, it is put on a cool-down to allow other messages to process. The cool-down period can be modified with property `com.glide.update_operation.job_cancel_timeout_minutes`. The job is still in the queue and is visible on the **Application Operation Queue** page.

If the queue is unable to download an application package or find a plugin to check for necessary resources to obtain, it logs an error against that queued operation. If the operation fails to preview its resources a determined number of times, the tracker fails, and the job is removed from the queue. The maximum number of times defaults to 3 and can be modified by the `com.glide.update_operation.max_failure_count` property.

Note:
Failing to obtain necessary locks doesn't count as a failure attempt. Only errors encountered such as failing to download an application package from AppRepo count.

Custom licensing for ISV applications

Monitor the usage of ISV applications with Subscription Management. Create a definition for your store application with the metadata you want collected. After publishing the application with the definition to the store, Usage Analytics runs and aggregates your defined metrics.

Overview

Applications that you create with the can be published to the ServiceNow Store. Monitor the usage of your applications with custom licensing.

You can use custom licensing on ISV applications that you publish to the ServiceNow Store if they meet the following criteria:

- The application is licensable.
- The application is not scoped in ServiceNow or Global.
- The application has a capacity subscription model.

Custom licensing functions through the usage analytics (com.glide.usageanalytics) and scoped app author (com.sn_appauthorr) plugins. These plugins are active by default in the App Engine. Additionally, your instance must be connected to an application repository or application store instance.

Custom license definitions

Custom licensing is achieved by creating a definition. A definition is a set of licensing metrics that you define based on what usage information you want to collect for your application. See Create a definition for your store application.

Create a definition for your store application

Create a definition for your store app to define the licensing metrics you want collected.

Before you begin
Role required: admin or developer

About this task
Create a definition for an application in development or for an application you have already published. However, if you assign a definition to a published application, you will have to re-publish it with the new definition. Additionally, you can only assign one definition to an application.

Note:
You can only create a definition for capacity subscriptions. See Types of subscriptions in Subscription Management.

Procedure

1. Navigate to **All** > **System Applications** > **My Company Applications**.
2. Select the application.
3. Under the **Subscription Management** section, set **Subscription Model** to **Capacity**.
4. Select the search icon next to **License Definition**.
   The **Application License Definitions** pop-up loads.
5. Select **New**.
6. Fill in the **Application License Definition** record.

Field	Description
Name	Name of the definition.
Description	Brief summary of the definition.

Field	Description
Metric Type	Type of collected metric.
Frequency	How often metrics are collected.
Performance Validated	Check the box if you have tested the performance of the definition.
Table	Table used to determine usage. **Note:** Only lists tables associated with the current application.
Query	Criteria to use on the table.
Aggregation	Type of aggregation. ○ Set to Count to count all records. ○ Set to Count (Distinct) to count unique records only.
Aggregation Column	Table column used for aggregation. **Note:** Column must be indexed to be aggregated.
Group By	Columns in the table to group the data by. Max of 3. **Warning:** Case sensitive.

7. Select **Submit**.
   Your new definition appears in the **Application License Definitions** pop-up.
8. Select the definition to apply it to the application.

What to do next

Create a new version of the application and publish it to the ServiceNow Store. While your application is in review, any changes to the definition may cause subsequent versions of your application to be rejected. Once the application is approved and published to the ServiceNow Store, the definition becomes read-only.

To change the definition of an application already on the ServiceNow Store, you must create a new definition, submit a new version of the application, and go through the review process.

Domain separation and Creator Workflow apps

Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

Domain separation is not supported for Creator Workflow apps.

Support level: No support

- The domain field may exist on data tables but there is no business logic to manage the data.
- This level is not considered domain-separated.

For more information on support levels, see Application support for domain separation.

Maintaining your application

Use your app builder tool to update and modify your application. Use your testing tool to verify that your application still functions properly.

Analytics and Dashboards

Performance Analytics Reporting  Integrate data visualization and analytics functionality into the workspace experience.	Analytics Hub  Use this exploratory view of indicators for more detailed analysis. You can see trends, predictions, breakdowns, and associated records for a specific indicator.
Analytics Center  Use the Platform Analytics Workspace with a single page for all the dashboards, visualizations, analytics answers, and insights on your instance.	Performance Analytics Dashboards  Continually visualize historical and real-time process health statistics in role-based dashboards that enable individual stakeholders to make informed decisions.

Security

Software Asset Management

Systematically track, evaluate, and manage software licenses, compliance, and optimization.

Security Operations

Bring incident data from your security tools into a structured response engine that uses intelligent workflows, automation, and a deep connection with IT to prioritize and resolve threats based on the impact they pose to your organization.

Service Mapping

Discover all application services in your organization and build a comprehensive map of all devices, applications, and configuration profiles used in these application services.

Monitoring

Monitor application health

Use the Application Service Dashboard to monitor the health state of application services. Then you can reduce the number of incomplete application services by populating them and adding any missing details.

For more information

You can either create a new workspace with an Analytics Center page in App Engine Studio or create a page from the Analytics Center page template. Either way, you have the same functions as the Analytics Center in the Platform Analytics Workspace.

Analytics Hub

In the Analytics Hub, analyze indicator scores by aggregating data, comparing scores, or viewing changes over time, and filter scores by breakdown. Enhance the Analytics Hub by adding targets, thresholds, trend lines, and useful comments for significant changes.

Monitor application health

Monitor counts indicating the health state of application services using the Application Service Dashboard. Next, reduce the number of incomplete application services by populating them and adding any missing details. To use application services effectively, ensure that application services are fully configured and populated.

Performance Analytics Dashboards

Your associates can get relevant, personalized insight by clicking a button instead of exporting data from databases and spreadsheets. You no longer manually create reports that quickly become stale and outdated.

Performance Analytics Reporting

Explore KPIs, and get answers and insights on analytics. Within the Platform Analytics Workspace, you can ask a natural language question and be shown the relevant analytics in an on-the-fly visualization, view any Next Experience dashboard that you own or has been shared with you, and create dashboards and data visualizations from your workspace runtime.

Security Operations

The ServiceNow® Security Incident Response application tracks the progress of security incidents from discovery and initial analysis, through containment, eradication, and recovery, and into the final post incident review, knowledge base article creation, and closure.

Service Mapping

Service Mapping enables IT departments of companies, organizations, and cloud companies providing platform as a service to create a service-aware view of infrastructure.

Software Asset Management

The ServiceNow® Software Asset Management (SAM) application systematically tracks, evaluates, and manages software licenses, compliance, and optimization. You can reclaim unused software rights, purchase new software rights, and manage allocations for entitlements.