Xanadu Build workflows

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

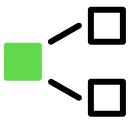
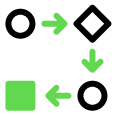
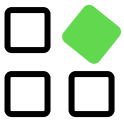
Table of Contents

Build workflows

Transform manual tasks and mundane work into digital workflows. Modernize legacy processes with hyperautomation. Enable citizen automation from one platform.

Get started

Increase the impact of your cross-enterprise workflows by automating manual processes. Transform business performance with automation. Optimize process efficiency and make work flow seamlessly.

Workflow Studio  Access workflow applications from a single interface.	**Workflow Studio flows**  Build multi-step flows from reusable components without having to code.	**Workflow Studio playbooks**  Author cross-enterprise workflows and create a single, unified process.
Workflow Studio playbook experience  Interact with a business workflow in real time from within Workspace.	**Workflow Studio decision tables**  Decouple decision logic from code by creating and maintaining decision rules.	**Service Creator**  Enables a department to offer custom services through the service catalog

Workflow Studio

The ServiceNow® Workflow Studio application provides a single location to access all process automation applications.

Workflow Studio flows

ServiceNow Workflow Studio flows are a ServiceNow AI Platform® feature that enables process owners to automate work. Create end-to-end digital workflows. Automate any process—from simple productivity to complex transformation—in a no-code, environment.

Workflow Studio playbooks

ServiceNow® Workflow Studio playbooks enable process owners to author cross-enterprise workflows and create a single, unified process. Build and manage multiple complex workflows easily with no-code playbooks and the enterprise application development platform.

Workflow Studio decision tables

Workflow Studio decision tables provides an intuitive interface to create and manage decision tables, which store sets of decision rules. Decision tables in Decision Builder embed business logic into a series of if-then decision rules. Decision tables read data from inputs. When all the conditions for a decision rule are met, the decision table returns one or more results. Decision Builder enables developers to decouple decision logic from the code base.

Playbook experience

Visualize complex processes in a simple and familiar task-oriented view built specifically for agents. Agents can use Playbook to update records, upload attachments, and complete tasks across multiple workflow activities.

Service Catalog item designer

The Service Catalog item designer enables non-administrators to create, maintain, and publish catalog items. It uses a structured design and publishing process to ensure consistency of usage.

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.

Classic workflow builder tools

The ServiceNow AI Platform supports these classic workflow builder tools.

Classic Approvals

Classic approvals are a legacy process to require authorization on tasks before the work is done. In earlier releases, you could create approval records to define approval tasks and associate users or groups to approve or reject them. You can replace classic approvals with Workflow Studio flows or classic workflows.

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. You can replace classic business rules with Workflow Studio flows.

Classic Events

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. You can replace classic events with Workflow Studio flows or Playbooks processes.

Classic Workflow

Workflow provides a drag-and-drop interface for automating multi-step processes across the platform. Each workflow consists of a sequence of activities, such as generating records, notifying users of pending approvals, or running scripts. The graphical Workflow Editor represents workflows visually as a type of flowchart. It shows activities as boxes labeled with information about that activity and transitions from one activity to the next as lines connecting the boxes. You can replace classic workflows with Workflow Studio flows or Playbooks processes.

Applications and features

Workflow Studio

Integrate workflow authoring, configuring, and monitoring into a single page experience. Consolidate Playbooks, Workflow Studio, Workflow Studio, Integration Hub integrations, and Decision Builder into one design environment.

Workflow Studio playbooks

Workflow Studio playbooks enable process owners to author cross-enterprise workflows and create a single, unified process. Build the underlying processes for playbooks that Playbook Experience agents and fulfillers use.

Workflow Studio flows

Flows automate a repeatable multi-step process. When the flow trigger conditions are met, the flow runs a sequence of reusable actions and flow logic to complete the process.

Workflow Studio actions

Actions automate a repeatable task or operation within a flow. Flows run actions by passing them data as inputs. Actions run a sequence of steps to complete the task, and pass data to the flow as outputs.

Workflow Studio decision tables

Workflow Studio decision tables enable developers to decouple decision logic from their code by creating and maintaining decision rules.

Playbook experience

Interact with a business workflow in real time from within Workspace. Agents can use Playbook to update records, upload attachments, and complete tasks across multiple workflow activities.

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.

Classic Approvals

Classic approvals are a legacy process to require authorization on tasks before the work is done. In earlier releases, you could create approval records to define approval tasks and associate users or groups to approve or reject them.

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.

Classic Events

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.

Classic Workflow

Workflow provides a drag-and-drop interface for automating multi-step processes across the platform. Each workflow consists of a sequence of activities, such as generating records, notifying users of pending approvals, or running scripts. The graphical Workflow Editor represents workflows visually as a type of flowchart. It shows activities as boxes labeled with information about that activity and transitions from one activity to the next as lines connecting the boxes.

Related ServiceNow applications and features

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.

ERP Data Hub

Work with remote tables in the ERP system of record, such as SAP, as well as APIs and ETLs, to create ERP data models to use as data sources for ServiceNow AI Platform applications.

ERP Customization Mining

Identify ERP customization apps that are candidates to migrate to the ServiceNow AI Platform. Use the ServiceNow AI Platform to build applications and features to automate ERP processes and workflows.

Integration Hub

Automate integration tasks using ServiceNow components for ServiceNow® Workflow Studio, or develop custom integrations. A separate subscription is required.

Integration Hub available spokes

Activate spokes to enhance your Workflow Studio experience with integration-specific content. Use prebuilt flows and actions to automate your integrations or create your own integration automation.

MID Server

The Management, Instrumentation, and Discovery (MID) Server is a Java application that runs as a Windows service or UNIX daemon on a server in your local network. The ServiceNow® MID Server enables communication and the movement of data between a ServiceNow instance and external applications, data sources, and services.

Robotic Process Automation (RPA) Hub

Use the ServiceNow® Robotic Process Automation (RPA) Hub to enable end-to-end automation for your organization. With a combination of UI interactions, element-based automations, and APIs that interact between the various business applications, you can emulate user actions and eliminate mundane and repetitive human activities.

Virtual Agent Designer

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.

Now Assist for Creator

Now Assist for Creator includes generative AI skills that can make developing on the ServiceNow AI Platform more efficient.

Get started

Install

Install Now Assist for Creator.

App generation

Generate simplified apps with AI-generated tables, experiences, and roles tailored to your text prompts.

Catalog item generation

Generate catalog items using generative AI.

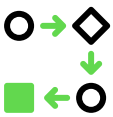
Code generation

Get help writing scripts quickly with AI-generated code based on text or code prompts.

Data visualization generation

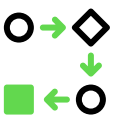
Generate Platform Analytics artifacts from conversational interactions.

Flow generation



Generate flow outlines from text directions.

Flow recommendations



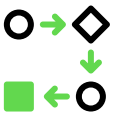
Generate recommendations for the next step of a flow.

Playbook generation

Generate playbook outlines from text directions.

RPA bot generation

Create automations, activities, and add automation logic from text instructions and preview options.

Spoke generation	UI generation	
		
Create spoke from third-party API documentation snippets.	Create experiences by describing what you want using natural language.	

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

- Workflow Automation product on the 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 skill usage metrics. 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.

Install Now Assist for Creator

Install the Now Assist for Creator application to add generative AI functionality to your workflows.

Before you begin

- Role required: admin
- Review the Now Assist for Creator application listing in the ServiceNow Store for information on dependencies, licensing or subscription requirements, and release compatibility.
- Upgrade to Washington DC Patch 1 or later. For more information about this release, see Available patches and hotfixes.
- Enable Next Experience. For information about activating Next Experience, see Considerations for activating Next Experience.

Procedure

1. Navigate to the Now Assist for Creator application on the ServiceNow Store .

   Important:
   Now Assist for Creator requires a separate subscription.

2. From the Now Assist for Creator application page, select **Request App**.
3. After approval has been granted, on your instance, navigate to **All** > **System Applications** > **All Available Applications**.
4. Find the Now Assist for Creator application (sn_now_creator) using the filter criteria and search bar.
5. Select **Install**.

What to do next

Turn on the Now Assist for Creator skills you want to support. Grant the now.assist.creator role to each user you want to use Now Assist for Creator skills.

Now Assist Creator role

The [now.assist.creator] role grants users access to Now Assist for Creator skills.

Contains Roles

This role contains no roles.

Groups

List of groups this role is assigned to by default.

None.

Special considerations

None.

Workflow Studio

Consolidate workflow authoring, configuring, and monitoring into a single page experience. Create and manage playbooks, flows, actions, decision tables, and integrations from one design environment.

Get started

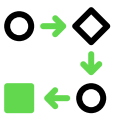
Explore

Learn about Workflow Studio concepts and features

Configure

Configure workflow builders

Using



Build workflows and workflow components

Reference

Review roles and system properties

Flow Assist

Generate flows from text prompts and generate recommendations for the next step of a flow.

Playbook Assist

Generate playbook outlines from text directions.

Troubleshoot and get help

- Workflow Automation product on the ServiceNow Community
- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

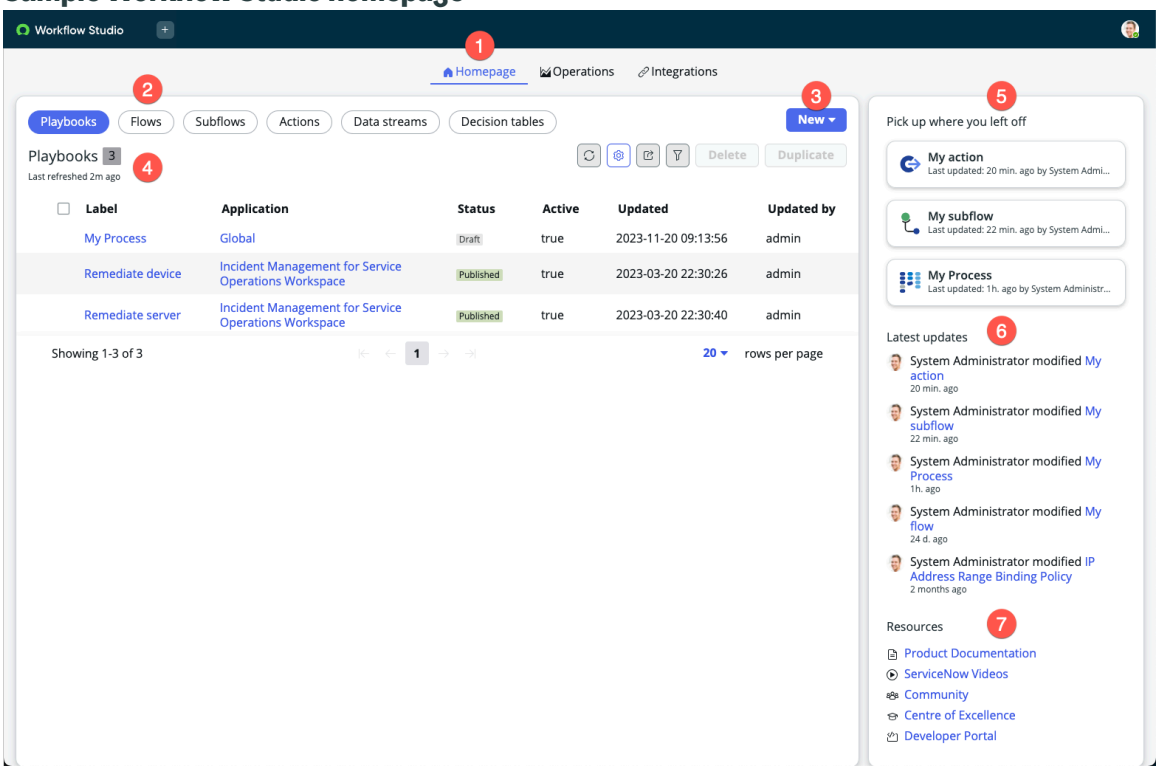
Exploring Workflow Studio

Work with playbooks, flows, subflows, actions, data streams, and decision tables from a single consolidated interface. See operational details of your workflow applications and configure integration connections and credentials.

Homepage

The Workflow Studio homepage displays all the workflow applications available for authoring and editing.

Sample Workflow Studio homepage



The Workflow Studio homepage consists of these elements.

1. **Homepage**

   Use the homepage to create and view workflows. You can see a list of workflows by type, a list of workflows you last worked on, a list of recent workflow updates, and a list of information resources.

2. **Workflow types**

   Select a workflow type to see a list of available items you can edit. Options include playbooks, flows, subflows, actions, data stream actions, and decision tables.

3. **Create a workflow**

   Select a workflow component to create. Options include playbooks, flows, subflows, actions, data stream actions, and decision tables.

4. **Current workflow list**

   See the list of workflow items available to edit. Each list shows the total number of workflow items available, a set of list controls, and a separate list row for each workflow item. List options include refresh list, list actions, copy URL, show filter panel, and delete.

5. **Pick up where you left off**

See the list of workflow items that the current user last worked on. Quickly resume working on a workflow item.

6. Latest updates

See the list of workflow items that have most recently been added or updated. See who worked on an item and when the updates were made.

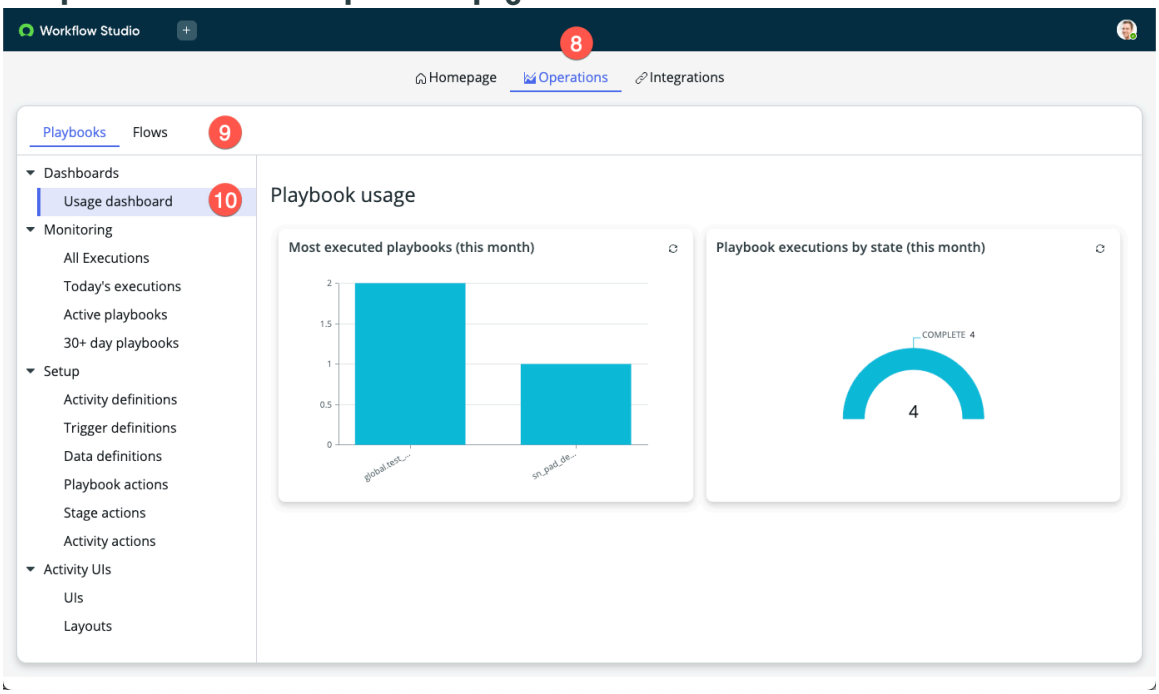
7. Resources

Learn more about Workflow Studio and its related workflow applications. Resources include links to the product documentation, ServiceNow videos, the ServiceNow Community site, the Centre of Excellence site, and the Developer Portal.

Operations

The Workflow Studio operations page displays a usage dashboard and execution details for the workflow components that are currently running and have completed running.

Sample Workflow Studio operations page



The Workflow Studio operations page consists of these elements.

8. Operations

Use the operations page to see dashboards and execution details by workflow type.

9. Workflow types

Select a workflow type to see available dashboards and execution details. Options include playbooks and flows.

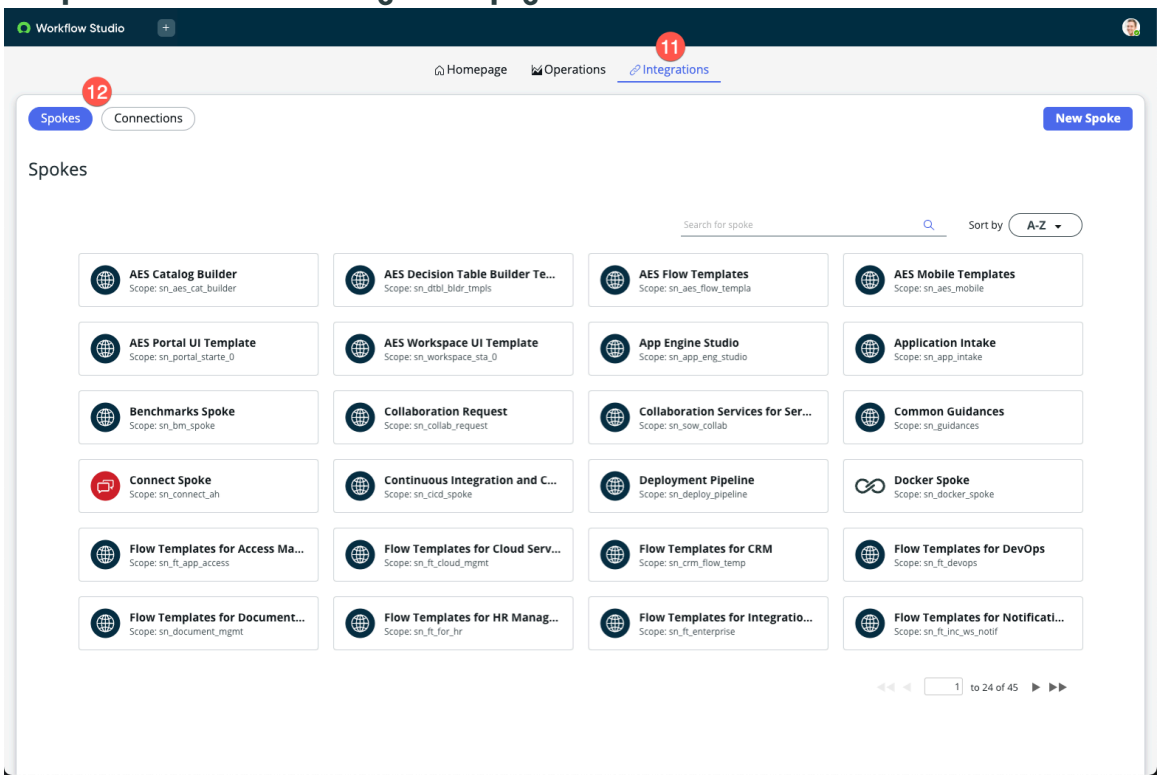
10. Dashboards, Monitoring, and Setup

Select a dashboard to see operational charts such as most executed playbooks and playbook executions by state. Only users with the admin role can view an operations dashboard. Select a monitoring option to see execution details. Select a setup option to see settings and properties.

Integrations

The Workflow Studio integrations page displays connection details for Integration Hub spokes. You can use this page to configure inbound and outbound spoke connections.

Sample Workflow Studio integrations page



The Workflow Studio integrations page consists of these elements.

11. Integrations

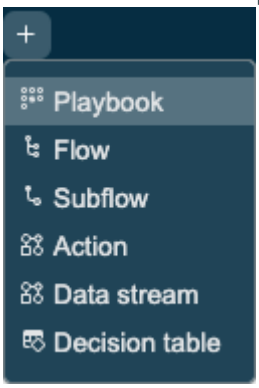
Use the integrations page to add, configure, or view available connections.

12. Integration types

See a list of available and configured spokes and connections. Configure or create a spoke to integrate data. Use the connection card to configure a connection or review its details.

Create button

Create button and options menu



Beginning with the Xanadu release, you can create new workflow items from any Workflow Studio page rather than just the Homepage. Use the Create button on the tab header to display a list of available workflow components.

- Playbook
- Flow
- Subflow
- Action
- Data Stream
- Decision table

Choosing Workflow Studio components

Flows, subflows, and actions are the basic building blocks of a process automation. Playbooks are built on flows, subflows, and actions, and come with a set of out-of-the-box activities driven by subflows that are already built. If you're trying to decide whether your automation should be a playbook rather than just a flow, start by determining if your flow needs user interaction.

- If your process automation requires user interaction, does it need manual user interaction? Examples of manual user interactions are a user reading a knowledge base article, going through a checklist, and gathering feedback.
  - If yes, use a playbook.
  - If no, does your process automation have several user interactions? Examples of user interaction are changing record field values such as changing the incident state or adding task work notes.
    - If yes, consider using a playbook for a better user experience.
    - If no, use a flow. Flows use less data storage in your instance.
- If your process automation doesn't require user interaction, is it a sequence of subflow calls?
  - If yes, is it hard to manage or difficult to view the flow?
    - If yes, consider using a playbook for a better user experience. Stages in playbooks are easier to view and also have additional configuration options.
    - If no, use a flow. Flows use less data storage in your instance.
  - If your process automation does not require any user interaction, keep it as a flow. Flows use less data storage in your instance.

Note:
You can also use flows that trigger a playbook just for interactive steps.

If you're trying to determine if a playbook should be a flow instead, start by determining whether your playbook is triggered 10s, 100s, or even 1000s of times per second.

- If your playbook is triggered 10s, 100s, or 1000s of times per second, are there any user interactions?
  - If yes, do the user interactions happen during every single run?
    - If yes, use a playbook. Playbooks offer visibility into where you're at in a process, particularly business processes with manual user steps.
    - If no, create a flow that triggers a playbook for the user interactions.
  - If no, use a flow. Flows use less data storage in your instance.
- If your playbook is not triggered very often, keep it as a playbook. Playbooks offer visibility into where you're at in a process, particularly business processes with manual user steps.

You can reference decision tables in flows, subflows, actions, and playbooks. Data streams are not used in flows, subflows, actions, playbooks, or decision tables. To learn more about data streams, see Data Stream actions and pagination.

Differences from previous releases

Workflow Studio combines playbooks, flows, subflows, actions, decision tables, and Integration Hub integrations into a single builder application. Each of the prior workflow applications retain their functionality and their user interface to create and edit workflow objects. Workflow Studio introduces these changes.

Access Workflow Studio from duplicate menu items

Workflow Studio uses the existing menu items for the separate workflow builder applications to open the corresponding workflow object type. For example, selecting the Process Automation Designer menu item opens the playbooks list from the Workflow Studio homepage.

Configure integration connections

Workflow Studio enables you to configure integrations in the same user interface as you build integration components. For example, you can create a data stream action and also define its connection details.

Create workflow components from any page

Beginning with the Xanadu release, Workflow Studio allows you to create a workflow component from any page rather than just the Homepage.

Display each workflow type in a separate tab

Workflow Studio displays each workflow object type in its own tab. For example, playbooks open in a playbook tab and flows open in a flow tab.

Edit different workflow types simultaneously

Workflow Studio combines all workflow object types into a single user interface. You can work on related workflow objects simultaneously rather than switch between separate workflow builder applications. For example you can create a flow that includes the Make a decision flow logic and also create the decision table used by that decision.

Set application specific settings from the existing property pages

Workflow Studio relies on the existing system properties pages to add and configure setting for each workflow application. For example, open the flows properties page to configure the behavior of flows, subflows, and actions.

View all available execution details

Workflow Studio contains execution details for all workflow object types. You no longer have to switch between applications to see execution details for a particular object type.

Exploring playbooks

Workflow Studio playbooks enable process owners to author cross-enterprise workflows and create a single, unified process. Build the underlying processes for playbooks that Playbook Experience agents and fulfillers use.

Note:

Starting in the Xanadu release, the following updates have been made:

- Playbooks is now part of Workflow Studio. Workflow Studio gives you a streamlined way to author, configure, and monitor playbooks, flows, subflows, actions, and decision tables in one place.
- Processes are now called Playbooks, though Playbook Experience remains a separate application that is not accessible from within Workflow Studio yet.
- The core Playbooks builder in Workflow Studio is available with the ServiceNow AI Platform® by default, but the latest updates are available for download through the application in the ServiceNow® Store.

Playbooks benefits

Workflow Studio Playbooks enables you, as a business playbook owner, to organize Workflow Studio content into unified and digitized cross-enterprise processes. With Playbooks, you gain these benefits:

- Connect multiple flows and actions into an end-to-end business workflow.
- Reuse existing Workflow Studio flows, subflows, or actions to automate playbook activities.
- Organize playbook activities in a digitized task board or diagram interface.
- Guide agents and fulfillers through complicated playbooks from start to finish, improving customer experience and task resolution. Build your playbooks in Workflow Studio, and then design and embed your Playbook Experience in legacy workspace, UI Builder, ServiceNow Mobile Platform, Service Portal, and more.
- Consolidate separate business processes across the organization.
- Define a consistent record life cycle from creation to completion.
- Pass data between the activities and stages of a business process.
- Specify the conditions and the order in which activities and stages run.
- Visualize and manage the activities and stages of your process.

Creating a well-designed playbook

The automated business processes that you design guide your end users and help them focus on the tasks and information that matter to them. A well-designed playbook can do these things:

- Start up, or trigger, automatically for the types of records that your end users care about
- Reuse activities from existing Workflow Studio content
- Has well-defined stages that end users can follow for a record
- Clearly show the next steps that end users must take to move through a record's life cycle

Playbooks content

Playbooks has these components:

Playbooks

A playbook is a ServiceNow AI Platform® representation of a cross-enterprise business process for your organization. A playbook owner is responsible for creating and maintaining a playbook.

Triggers

A trigger specifies when to start running your playbook.

Stages

A stage is a grouped sequence of activities in a playbook. A playbook owner creates a stage to specify a logical grouping of activities. A stage in your overall business process.

Activities

An activity defines the Workflow Studio content that powers the playbook's automation. An activity can also specify the user-facing experience that the playbook produces when it runs.

For more information about how to use and navigate the Playbooks user interface, see Playbooks.

Getting started

Before you get started with Playbooks, familiarize yourself with any features that your business uses to automate operations on the ServiceNow AI Platform, such as flows, subflows, and actions.

If you're a playbook owner who wants to learn the basics of digitizing your business process, check out the following resources:

- Get started with ServiceNow® Process Automation
- Get started with processes
- Design your first automated process
- View your process executions

If you're a ServiceNow Process Automation administrator who wants to set up and customize Playbooks, check out the following resources:

- Triggers
- Process definitions
- Activity definitions

Playbooks User Experience

Understand how Workflow Studio Playbooks work in the ServiceNow AI Platform® to automate cross-functional processes and consolidate them into task-oriented views for your end users.

All > **Process Automation** > **Workflow Studio** > **Playbooks**is the design environment where playbook owners build playbooks. Meanwhile, the runtime experience is where end users, such as playbook agents, follow the playbook to complete a business process.

Design environment

The Playbooks design environment in Workflow Studio consists of these components:

Playbooks

A playbook is where a playbook owner configures and organizes multiple instances of Workflow Studio content into a coherent business process. A playbook consists of a trigger and a sequence of stages, which are made up of a sequence of activities.

Trigger definitions

A trigger definition specifies the conditions that must be met to run a playbook. A user with the admin, playbook.admin, or pd_trigger_author role typically creates and configures a trigger definition that playbook authors can use as a template. A trigger definition specifies the record operation and table conditions that must be met to start running a playbook. A playbook owner typically selects a trigger template when creating a playbook.

Trigger instances

A trigger instance is produced when you select a trigger template. The trigger instance stores the conditions that a record must meet to start running the playbook.

Stages

A stage is a logical grouping of activities in a playbook. A playbook owner creates a stage to group activities and specify the start rule for when the stage should start running. A stage in your overall business process.

Activity definitions

An activity definition maps subflow and action inputs and outputs to an activity instance. An activity definition contains:

- The automation plan to map the triggering input record data to action or subflow inputs
- The activity experience to map action or subflow outputs to a user-facing view of the playbook

A user with the admin, playbook.admin, or pd_content_author roles typically specifies the automation plan and activity experience when creating an activity definition.

Activity instances

An activity instance is produced when you add an activity to a playbook. The activity instance stores the automation plan data mappings from the activity definition. You can change these data mappings when the default values do not fit your playbook. The playbook can specify the start rules for when the activity should start running.

Start rules

A start rule specifies when a stage or an activity starts running. A playbook owner can use start rules to specify what parts of a playbook run simultaneously and what parts run serially.

For more information about how to use and navigate the Workflow Studio user interface, see Playbooks.

Runtime experience

Workflow Studio produces these runtime components for Playbooks:

Process executions

A process execution stores the details of running a playbook in a context record. You can use a process execution to troubleshoot and verify that playbooks run as expected.

Activity executions

An activity execution stores the details of running an activity instance in a context record. You can use an activity execution to troubleshoot and verify that playbooks run as expected.

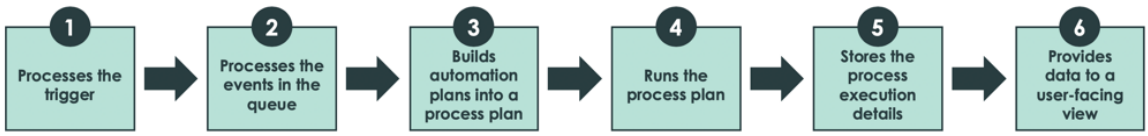
Playbook runtime

Playbook runtime is when a playbook runs for an agent or fulfiller. A playbook runs for agents only after Playbook Experience administrators configure how and where the playbook appears. See Set up a Playbook.

During runtime for a playbook, your instance:

1. Evaluates any conditions specified in the trigger definition and processes the trigger.
2. Processes the Events and starts running the playbook in the background.
3. Builds the automation plans from each activity into an entire process plan.
4. Runs the process plan for your playbook.
5. Stores the process execution information in the Process Execution [sys_pd_context] table.
6. Provides data for the running playbook view that agents and fulfillers experience.

Playbook processing



Your instance processes a playbook during runtime by evaluating trigger conditions, processing the event in the queue, building and running a process plan, storing process execution details, and providing data for the Playbook Experience.

Data security and HTML sanitization

Playbooks protects against cross-site scripting and code injection by evaluating all string data for HTML markup. The system only preserves HTML markup that is present in its inclusion list. All other HTML markup is removed from string data.

The inclusion list supports these HTML elements and attributes, which cannot be modified.

HTML inclusion list

HTML element	Included Attributes
a	class, href, target, title
abbr	class, title
address	class
area	alt, class, coords, href, shape
article	class
aside	class

HTML inclusion list (continued)

HTML element	Included Attributes
audio	autoplay, class, controls, loop, preload, src
b	class
bdi	class, dir
bdo	class, dir
big	class
blockquote	cite, class
br	class
caption	class
center	class
cite	class
code	class
col	align, class, span, valign, width
colgroup	align, class, span, valign, width
dd	class
del	class, datetime
details	class, open
div	class
dl	class
dt	class
em	class
emp	class
font	class, color, face, size
footer	class
h1	class
h2	class
h3	class
h4	class
h5	class
h6	class
header	class
hr	class
html	
i	class
img	alt, class, height, src, title, width

HTML inclusion list (continued)

HTML element	Included Attributes
input	aria-label, class, type, value
ins	class, datetime
li	class
mark	class
nav	class
ol	class
p	class
pre	class
s	class
section	class
small	class
span	class
sub	class
sup	class
svg	class
strong	class
style	
table	align, border, class, valign, width
tag	class
tbody	align, class, valign
td	align, class, colspan, rowspan, valign, width
tfoot	align, class, valign
th	align, class, colspan, rowspan, valign, width
thead	align, class, valign
tr	align, class, rowspan, valign
tt	class
u	class
ul	class
video	autoplay, class, controls, height, loop, preload, src, width

Playbooks

Playbooks are a component in Workflow Studio. Workflow Studio gives you a streamlined way to author, configure, and monitor playbooks, flows, subflows, actions, and decision tables in one place.

Workflow Studio landing page

You can view the Workflow Studio landing page by navigating to **Process Automation** > **Workflow Studio**. The landing page opens to Playbooks by default, but you can easily navigate to flows, subflows, actions, and decision tables. For more information on Workflow Studio, see Workflow Studio.

Playbooks builder

The builder for Playbooks consists of the main header, design space, and side panel.

Main header

The main header displays information about the playbook that you're currently designing. In the main header, you can:

- See the status and activation state of your playbook.
- Toggle between Diagram and Board view.
- Undo or redo your actions.
- See and navigate to your errors via the error tray.
- Turn Optional activities on or off.
- Test and activate your playbook so that it runs as expected when triggered. You can also preview the Playbook Experience during testing. For more information, see Playbook statuses and activation states.
- In the **More actions** menu, you can also deactivate or duplicate your playbook.
- Also in the **More actions** menu, access the properties of your playbook. You can add or edit the name or description for your playbook, enable playbooks to restart, and edit the behavior of your trigger. For more information, see Process definition properties.

Design space

Build your playbooks in either Diagram or Board view. You can perform most of the same functions in either view.

> **Note:**
> Decision activities are not available in Board view, and Optional activities are not available in Diagram view.

Organize activities into stages to design your playbook.

- An activity represents one step within your overall business process. An activity can automate operations on the ServiceNow AI Platform, such as creating or updating records, displaying record information, and running automated actions in the background.
- Organize a set of activities into stages within your business process.

For more information, see Stages and activities.

Side panel

The side panel lets you configure your activities, and stages. In the side panel, you can:

- Add or edit the name and description for your stage or activity.
- Define the start rule for your stage or activity.

- Define what your activity or stage does when restarted.
- Add or edit the inputs for your activity.
- Define additional properties for how your activity renders during runtime.

For more information on creating a playbook, see Getting started with Playbooks.

When to use flows and playbooks

Use these general guidelines to determine when to create a flow or a playbook.

When to use flows

Flows, subflows, and actions are the basic building blocks of process automation. Flows run when their trigger conditions are met, and each flow in turn runs a sequence of actions, flow logic, and subflows. The actions, flow logic, and subflows within a flow are what create and update data.

A flow is a good fit for process automations that met these criteria.

Expect few to no manual user interactions

As long as a flow has the input data it needs, it can run to completion without any user interaction. Some flow logic and actions require users to make record changes, but a flow can automatically pause until its wait conditions are met. Process automations that depend on user interactions such as reading a knowledge base article, going through a checklist, and gathering feedback are harder to manage with flows. Flows don't directly provide any UI elements for users to interact with. Flows depend on users knowing how to find an existing UI and making any needed changes. For example, a record-based flow depends on a user making a change in a specific record such as a case or an incident.

Expect to run at high volumes

An instance can run hundreds to thousands of flows per second. With flow reporting being disabled by default, an instance can run a high volume of flows before it sees any performance impact. If you expect to run a process automation at high volumes, a flow is a good fit over a playbook because it requires less overhead and system resources.

Expect to run few to no subflows

The more subflows a flow calls, the more difficult it becomes to manage from the flows interface. While you can use conditional flow logic or a decision table to choose a subflow to run, playbooks offer a better user experience for running a sequence of subflows.

When to use playbooks

Playbooks are built on activities, which use prebuilt flows, subflows, and actions as their building blocks.

A playbook is a good fit for process automations that met these criteria.

Expect several manual user interactions

Playbooks provide UI elements for users to interact with. The playbook experience guides users to make any changes required to advance the playbook.

Expect to run at low volumes

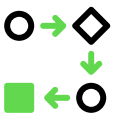
Playbooks require more system resources to run because they generate UI elements and store more execution details.

Expect to run many subflows

Playbooks offer a better user experience for running a sequence of subflows.

Playbook Assist

Get started

Explore  Learn more about what Playbook Assist supports	Configure  Enable Playbook Assist for an instance
Generate playbooks  Generate playbooks	Reference  Learn about the roles required to use Playbook Assist

Important:
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. Please 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 Playbook Assist

Use Playbook Assist to generate a playbook from text, an image or both. For example, you can upload an image of a business process you've outlined on a whiteboard or in diagramming software, describe more details of the process, and generate a playbook from those combined inputs. Playbook Assist is part of the Now Assist for Creator application.

Activate the playbook skills in the Now Assist for Creator app. Playbook skills include:

- Playbook generation
- Playbook generation with images
- Playbook recommendations

Note:
Playbook authors must be assigned the **now.assist.creator** role to use playbook generation. See Playbook Assist roles for more information about this skill.

Playbook authors can provide an image of a business process from a whiteboard or diagramming software and add details with a text description to create multi-stage playbooks. As of version 26.2 for Now Assist for Creator, Now Assist inserts the activities that align most closely with your inputs. For activities that Now Assist isn't able to guess, placeholder activities are inserted.

Note:
For placeholder activities, you can use the recommendations skill to find activities that might best replace placeholder activities, or manually find and configure activities before activating your playbook.

To learn more about recommendations, see Playbook recommendations.

Activation

Playbook generation, **playbook recommendations**, and **playbook generation with images** are skills that are installed with the Now Assist for Creator (sn_now_creator) application. You can install this application from the ServiceNow Store website.

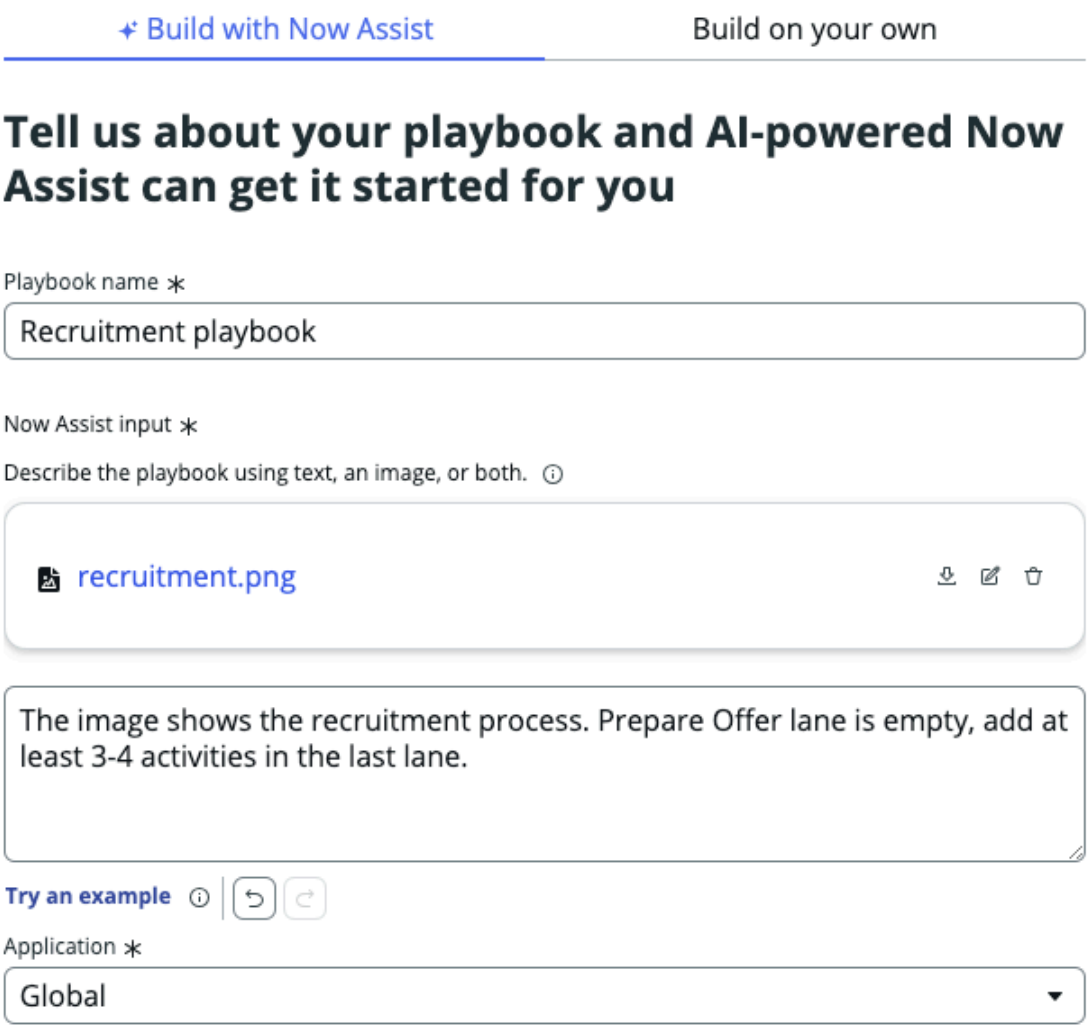
Supported Now Assist skills

Playbook Assist currently supports the playbook generation, playbook recommendations, and playbook generation with images skills in English. The playbook generation with images skill is not available for ServiceNow instances hosted in the APAC region.

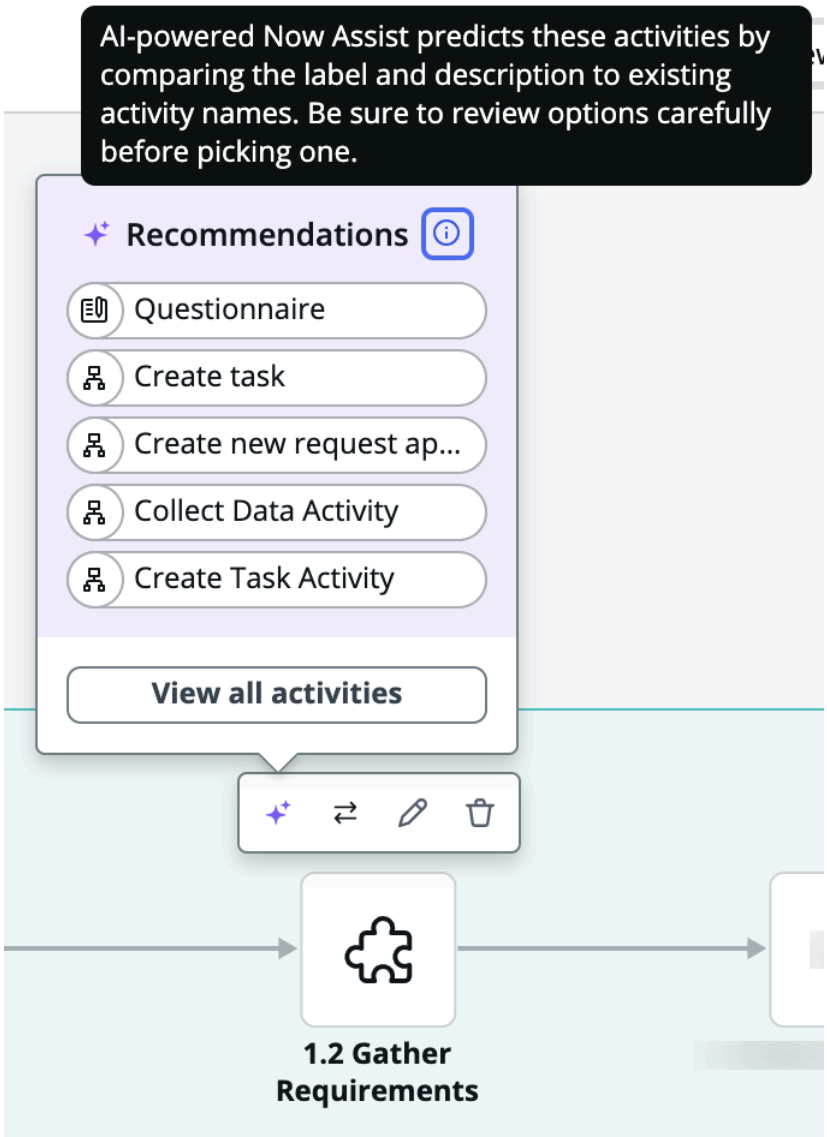
Supported user interfaces

Access the playbook generation skill when you’re:

- Creating a playbook in Workflow Studio from image, text, or both.



- Replacing placeholder activities.



Providing inputs and reviewing playbooks

Follow these guidelines when creating playbooks with playbook generation skills.

Provide inputs:

Provide a meaningful name for the playbook

The more descriptive the playbook name is, the better the playbook that Now Assist can create.

Make sure your image is the right format and size.

- The image should be in JPG, JPEG, PNG, or WEBP format.
- The image should be 10MB or less.

Make sure your image is clear and visible.

- Upload an image that is clear and visible to the human eye. Make sure that any writing is legible. If you can't what's in an image, neither can Now Assist.
- When uploading an image, it is the primary input. You can use the text description to add more context about the business process in your uploaded image.

Be precise and descriptive in text inputs

- Describe each stage and activity in as much detail as you can.
- Specify the order that stages and activities run in.
- Specify if any stages or activities run at the same time.
- Add additional context for an image input.

Experiment with inputs

Save your inputs somewhere, including any modified versions. Saving your inputs enables easy comparison of results.

Note:
Inputs used only to generate a preview are not saved, but inputs used for a saved playbook are in the playbook's **Properties** setting.

Tip:
To learn more about playbook generation inputs, see General Guidelines for Building Playbooks with Now Assist.

Review playbooks:

Check for accuracy

Review each stage and activity in a generated playbook to determine accuracy, efficiency, and how well it outlines your business process.

Configure placeholder activities

Configure placeholder activities before you activate your playbook. Use playbook recommendations to help choose activity definitions.

Configure the activity definition in the activity side panel

Configuring Playbook Assist

Enable Playbook Assist skills in the Now Assist for Creator application so that you can get started with building playbooks faster.

Install Playbook Assist

Install the Now Assist for Creator application to add the Playbook Assist generative AI capability to your playbooks.

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.
- Upgrade to Process Automation Designer 25.2 or later. 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.
- Enable Next Experience. For information about activating Next Experience, see Considerations for activating Next Experience.
- Role required: admin

Procedure

1. Navigate to the Now Assist for Creator application on the ServiceNow Store .

   Important:
   Now Assist for Creator requires a separate subscription.

2. On the Now Assist for Creator application page, select **Request App**.
3. After approval has been granted, navigate to **All** > **System Applications** > **All Available Applications**.
4. Find the Now Assist for Creator application (sn_now_creator) by using the filter criteria and search bar.
5. Select **Install**.

Result
Now Assist for Creator is installed on the instance.

What to do next
Configure Now Assist for Creator to turn on the Playbook Assist skills. Grant the now.assist.creator role to each user that you want to use Playbook Assist skills.

Turn on the playbook generation skill

Turn on the Now Assist for Creator playbook generation skill to use generative AI to create playbooks.

Before you begin

- Install the Now Assist for Creator application
- Role required: admin

About this task

Important:
The playbook generation skill requires a separate subscription to Now Assist for Creator.

Procedure

1. Navigate to **All** > *Now Assist Admin* > **Features**.
2. In the workflow list, select **Creator**.
3. In the Playbook Assist card, select **View details**.

4. Select **All available Playbook Assist skills** > **Playbook generation** > **Turn on skill**.
5. In the Playbook Assist card, verify that the playbook generation skill is activated under **All Playbook Assist skills**.

Result
Playbook Assist skills are active on your instance.

What to do next
Grant the now.assist.creator role to users who have permission to use the playbook generation skill.

Turn on playbook generation with images

Turn on the Now Assist for Creator playbook generation skill to use generative AI to create playbooks from an image.

Before you begin

- Install the Now Assist for Creator application
- Activate the **Playbook generation** and **Playbook recommendations** skills
- Role required: admin

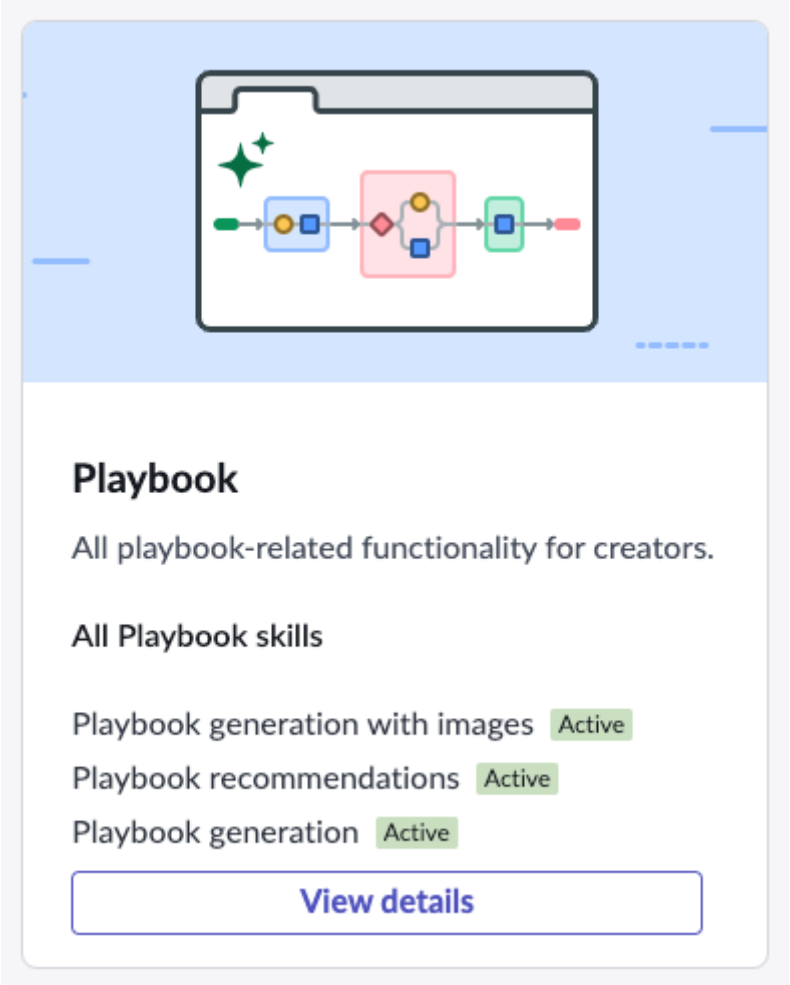
About this task

Important:
The playbook generation skill requires a separate subscription to Now Assist for Creator.

Procedure

1. Navigate to **All** > *Now Assist Admin* > **Features**.
2. In the workflow list, select **Creator**.
3. In the Playbook Assist card, select **View details**.
4. Select **All available Playbook Assist skills** > **Playbook generation with images** > **Turn on skill**.

5. In the Playbook Assist card, verify that the playbook generation with images skill is activated under **All Playbook Assist skills**.



What to do next

Grant the **now.assist.creator** role to each user who will use the playbook generation with images skill.

Change the maximum image size

Adjust the maximum size that is accepted for images used for playbook generation.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **System Properties** > **All Properties** for the [sys_properties.list] table.
2. Search for and open the *com.glide.attachment.max_get_size* property.
3. If the property doesn't exist yet, create the *com.glide.attachment.max_get_size* property.

   a. Select **New** in the upper right corner.

   b. Enter values for the following fields:

Field	Description
Name	Name the property *com.glide.attachment.max_get_size*.
Type	Select **integer** from the drop-down.
Value	Enter the maximum bytes allowed for an uploaded image. The default value is 5242880 bytes (5 MB), but you can increase this up to 10485760 bytes (10 MB) or down to 102400 bytes (100 KB). **Megabytes (MB)** If the equivalent MB value has a decimal and is between 1 MB and 10 MB, the value is rounded down to the next whole MB. For example, if you enter 4980736 bytes (4.75 MB), the maximum size will be 4194304 (4 MB). **Kilobytes (KB)** If the value is less than 1 MB, the equivalent KB value is rounded down to the next whole KB. For example, if the value is 580608 bytes (567 KB), the maximum size limit is rounded down to 512000 bytes (500 KB). **Less or greater than 5 MB** Images smaller than 5242880 bytes (5 MB) are scaled down. Images that are larger than 5 MB are not scaled and the system creates a link instead. **Greater than 10 MB** If the value is more than 10 MB, such as 12582912 bytes (12 MB), the maximum size defaults to 10485760 bytes (10 MB). **100 KB minimum** If the value is less than 102400 bytes (100 KB), such as 91136 bytes (89 KB), the maximum size defaults to 102400 bytes (100 KB).

4. Select **Submit** in the upper right corner.

Result
The maximum size for an image upload is set or changed.

Related topics
- Generate a playbook
- Administering attachments
- General Guidelines for Building Playbooks with Now Assist

Turn on playbook recommendations

Turn on the playbook recommendations skill to get recommendations for the activity definition to use in your placeholder activities with AI Search.

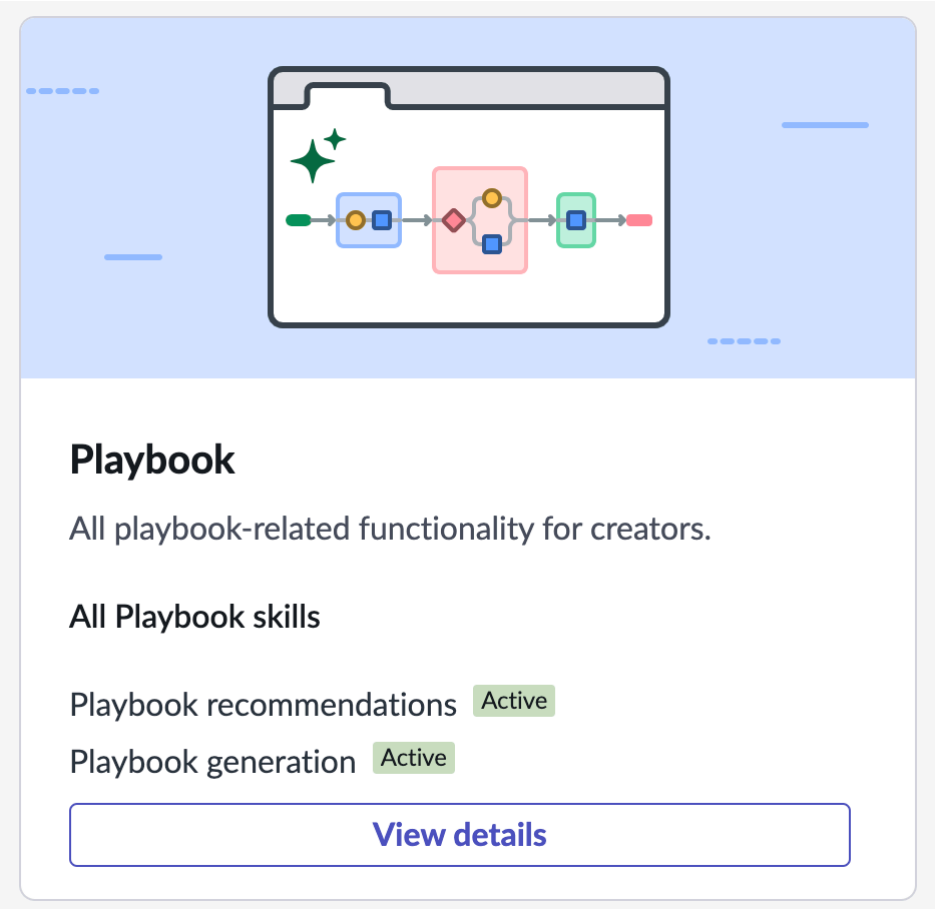
Before you begin
- Role required: admin
- Activate the **Playbook generation** skill

- Install the Now Assist for Creator application.

> **Important:**
> The playbook recommendations skill requires a separate subscription to Now Assist for Creator.

Procedure

1. Navigate to **All** > **Now Assist Admin** > **Features**.
2. In the workflow list, select **Creator**.
3. In the Playbook Assist card, select **View details**.
4. Select **All available Playbook skills** > **Playbook recommendations** > **Turn on skill**.
5. In the Playbook Assist card, verify that the **Playbook recommendations** skill is activated under **All Playbook Assist skills**.



What to do next

Grant the **now.assist.creator** role to each user who will use the playbook recommendations skill.

Generate a playbook

Provide an image, text, or both inputs to Now Assist to generate a playbook.

https://player.vimeo.com/video/1024019900?h=0543dba1e7&badge=0&autopause=0&player_id=0&app_id=58479

Before you begin

Learn how to write prompts to generate better playbooks. For more information, see General Guidelines for Building Playbooks with Now Assist.

Role required:

- admin, playbook.admin, pd_author, or a delegated developer permission
- now.assist.creator

Procedure

1. Navigate to **All** > **Workflow Studio**.
   The Workflow Studio landing page appears. Playbooks is shown by default, but you can toggle to flows, subflows, actions, and decisions.
2. Open the **New** drop-down menu and select **Playbook**.
   The new playbook opens in a Workflow Studio tab. **Build with Now Assist** is the default method, but you can switch to **Build from scratch**.
3. On the **Build with Now Assist** tab, fill in the following fields.

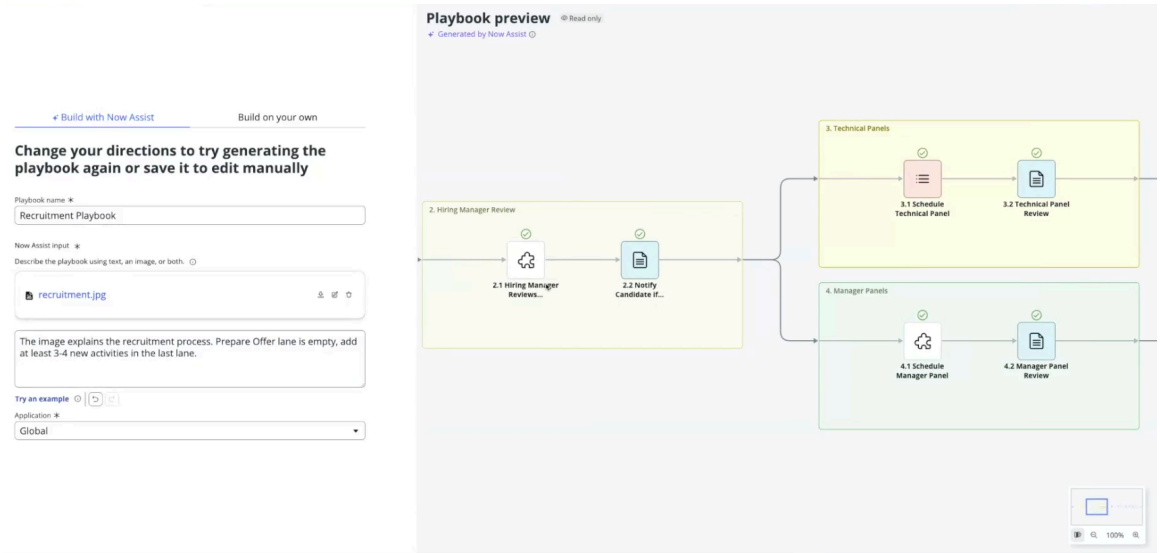
Field	Description
Playbook name	Unique, user-facing name for your playbook. This name also appears to agents and fulfillers during runtime.
Now Assist input	Describe the playbook using text, an image, or both. **Make sure your image is the right format and size.** ○ The image should be in JPG, JPEG, PNG, or WEBP format. ○ By default, the image should be 5MB or less. **Note:** To change the max image size, change the **com.glide.attachment.max_get_size** system property. To earn more about changing the max image size, see Change the maximum image size. **Make sure your image is clear and visible.** ○ Upload an image that is clear and visible to the human eye. Make sure that any writing is legible. If you can't what's in an image, neither can Now Assist. If the image is unclear in any way, Now Assist returns an error and cannot generate a preview. This error is also

Field	Description
	displayed if an image is not a business process. ○ When uploading an image, it is the primary input by default. You can use the text description to add more context about the business process in your uploaded image. **Note:** Since an image is the primary input, you will receive an error for a unusable or unclear image even if you include a valid text input. **Be precise and descriptive in any text input.** ○ Describe each stage and activity in as much detail as you can. ○ Specify the order that stages and activities run in. ○ Specify if any stages or activities run at the same time.
Application	Application scope that you want your playbook to run in. Selecting **Global** lets your playbook run in any application scope. For more information, see Application scope. **Important:** You can't change the application scope of a playbook after you've generated a preview for it.

4. Select **Generate playbook preview**.

   Workflow Studio uses your image and text inputs to build a playbook. If successful, Workflow Studio displays a preview of the playbook in diagramming view.

   Now Assist does its best to guess which common activities to use in your generated playbook, and inserts placeholder activities when it is unable to guess.

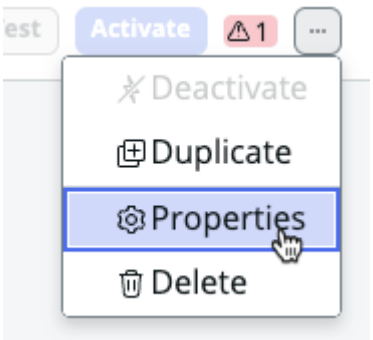


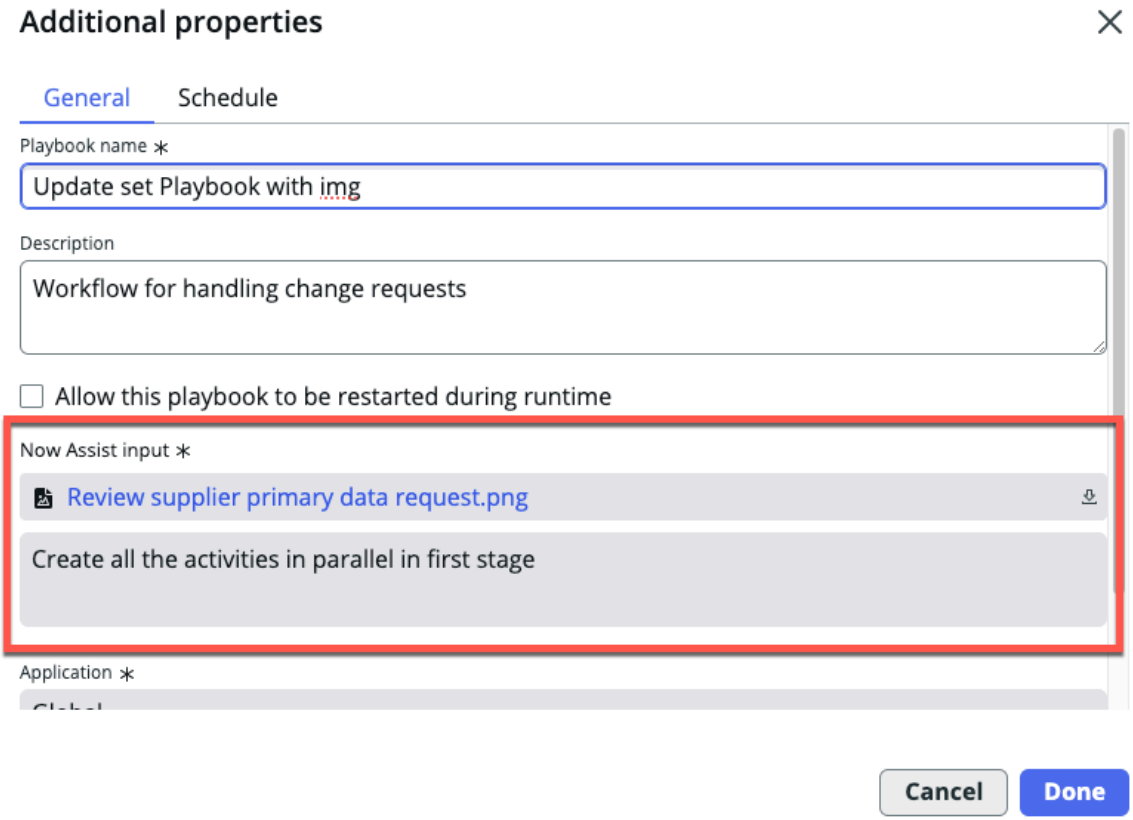
5. **Optional:** Scroll horizontally in the preview, and zoom in and out as needed.
6. Review the preview of the playbook for accuracy.
7. **Optional:** If the playbook doesn't meet your requirements, try updating your image or text inputs according to General Guidelines for Building Playbooks with Now Assist, and select **Regenerate preview**.
8. If you're ready to generate your playbook, select **Save and edit playbook**.

> **Note:**
> Generating or regenerating a playbook preview counts as 10 assists. To track your Now Assist usage, see Monitoring Now Assist usage in Subscription Management. Opening a playbook does not count as an assist.

A playbook outline is generated, and common activities are inserted where possible.

9. **Optional:**
   To view the prompt that was used to generate a playbook, navigate to **More actions menu** > **Properties** > **General** > **Now Assist directions**.





10. Configure your trigger.
    For more information about triggers, see Configure your trigger.
11. Configure placeholder activities by manually selecting the placeholder activity.

    Tip:
    To generate recommendations for activity definitions from Now Assist instead, see Generate playbook recommendations.

    a. Select a placeholder activity that you want to configure ( ).

    You can also hover over the placeholder activity and select the **replace activity** icon (in the mini-picker to directly open the activity picker.

    b. **Optional:** Update the **Label** and **Description**, if needed.

    c. Under the **Activity definition** field, select the edit button ( ).
    The activity picker is displayed.

    d. In the activity picker, search for the activity, subflow, or action to add.

    Note:
    Select the application first, and then the activity from the resulting list. For more information about subflows or actions, see subflow, or action.

    e. Configure the activity inputs.
    For more information about common activities and their inputs, see Playbooks reference.

12. **Optional:** If you don't see the activity that you want to add in the activity picker, create an activity definition.
    For more information, see create an activity definition.
13. After you configure all your stages and activities, select **Activate** in the header.
    Activating your playbook publishes it so that it runs when triggered.
14. **Optional:**

    Note:
    When you change your playbook after activating it, the system saves your changes but deactivates your playbook.

    To publish any new changes to your playbook, select **Activate** again.
    For more information, see Playbook statuses and activation states.

Result

When your playbook's trigger conditions are met, your playbook runs. As a result, the system creates a Process Execution record and renders user-facing configurations for Playbook Experience. For an example of how to digitize a manual business process that renders as a playbook, see Design an automated process.

What to do next

Design the Playbook Experience for your agents and fulfillers in UI Builder. To learn how to design and customize the runtime playbook experience in UI Builder, see Customize the Playbook Experience.

Playbook recommendations

Get AI-generated recommendations for placeholder activities. The system generates recommendations based on an activity's name and description.

https://player.vimeo.com/video/1024019900?h=0543dba1e7&badge=0&autopause=0&player_id=0&app_id=58479

Activation

Now Assist Recommendations is a skill installed with the Now Assist for Creator (sn_now_creator) application. You can install this application from the ServiceNow Store website.

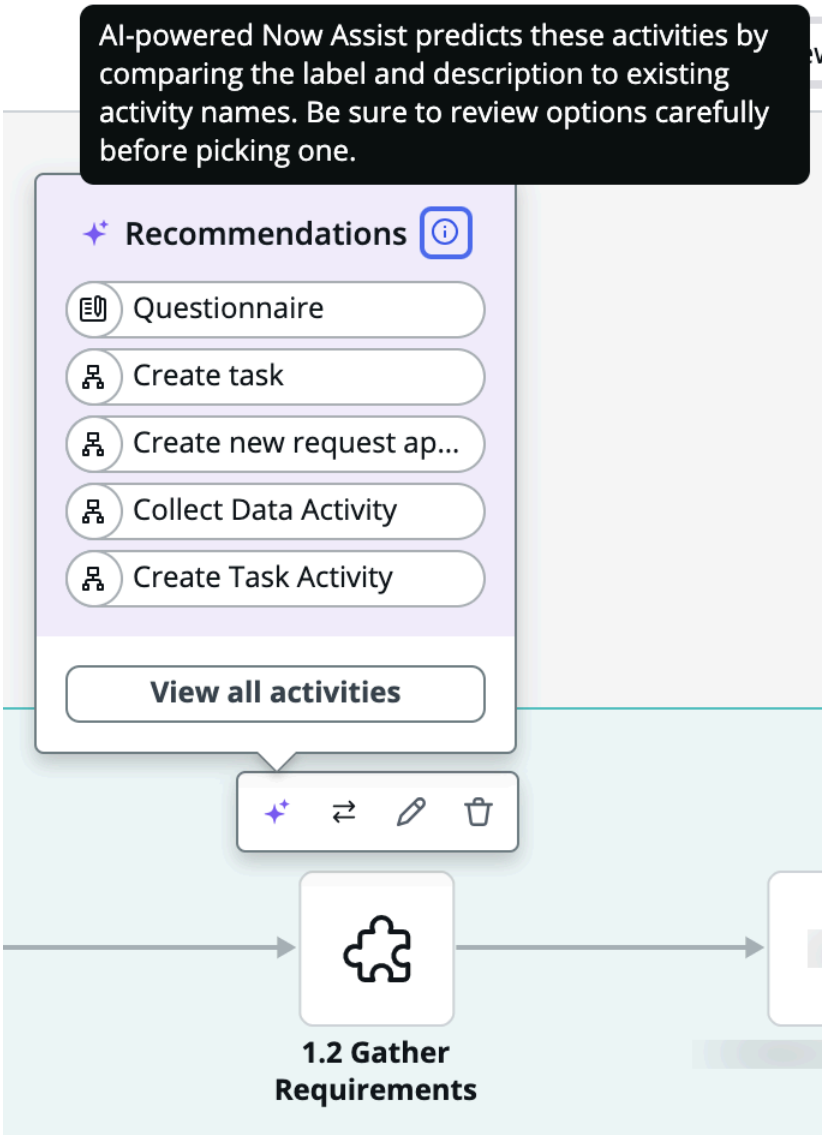
Benefits

Activating the Now Assist Recommendations skill helps to search through all available activity definitions, flows, subflows, and actions on the instance, which enables quicker configuration of placeholder activities in your playbook outline, which then reduces the total time to playbook activation.

Supported user interfaces

Access the Now Assist Recommendations skill from the Playbooks user interface.

Example Now Assist recommendations



The Now Assist Recommendations skill uses the name and description of the activity to generate one to five recommendations for the activity definition to use for a placeholder activity. If there are no recommendations listed, then no activity definitions are considered relevant to the activity name and description.

Generative AI model training

This Generative AI large language model was pre-trained with internal ServiceNow playbooks to learn playbook creation patterns. The goal was to understand what playbook activities are most relevant for a certain position in a playbook given the trigger and previous activities.

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.

Generate playbook recommendations

Select the activity definition for a placeholder activity from a list of AI-generated recommendations. The system generates recommendations based on an activity's name and description.

Before you begin

- Make sure the playbook recommendations skill is turned on. To learn how to turn on the recommendations skill for playbooks, see Turn on playbook recommendations.
- You can only generate recommendations for placeholder activities in a generated playbook. To learn how to generate a playbook, see Generate a playbook.
- Role required:
  - admin, playbook.admin, pd_author, or a delegated developer permission
  - now.assist.creator

Procedure

1. In your generated playbook, hover over any placeholder activity and select the recommendations icon ( ) in the mini-picker.
2. Select one of the recommended activity definitions, if appropriate.

   Note:
   If there are no recommendations listed, then no activity definitions are considered relevant to the activity name and description.

3. **Optional:** Try updating the **Label** and **Description** to improve results.
   a. Open the placeholder activity.
   b. Update the **Label** and **Description**.
   c. Under the **Activity definition** field, select the recommendations button ( ) to try to generate recommendations again.
   d. Select one of the recommended activity definitions, if appropriate.
4. Continue on with activity configuration from Step 11.e of the Generate a playbook procedure.

Result

When your playbook's trigger conditions are met, your playbook runs. As a result, the system creates a Process Execution record and renders user-facing configurations for Playbook Experience. For an example of how to digitize a manual business process that renders as a playbook, see Design an automated process.

What to do next

Design the Playbook Experience for your agents and fulfillers in UI Builder. To learn how to design and customize the runtime playbook experience in UI Builder, see Customize the Playbook Experience.

Playbook Assist reference

Reference topics provide additional information about configuration properties, roles, and more.

Playbook Assist roles

The following roles are installed for use with the Now Assist for Creator playbook generation skill.

You can grant users entitlement to the applications that you purchase on the ServiceNow AI Platform by allocating subscriptions in Subscription Management. You allocate subscriptions by adding one or more groups with measured roles to a product subscription.

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 playbooks by using AI-powered playbook generation.

Groups

This role is assigned to no groups by default.

Contains Roles

This role contains no roles.

Elevated

This role isn't an elevated role.

Special considerations

None

General Guidelines for Building Playbooks with Now Assist

See guidelines and examples for using the Now Assist Playbooks capabilities.

Image inputs

The following examples can help you to generate playbooks using an image input:

Make sure your image is the right format and size.

- The image should be in JPG, JPEG, PNG, or WEBP format.
- By default, the image should be 5MB or less.

> **Note:**
> To change the max image size, change the **com.glide.attachment.max_get_size** system property. To earn more about changing the max image size, see Change the maximum image size.

Make sure your image is clear and visible.

- Upload an image that is clear and visible to the human eye. Make sure that any writing is legible. If you can't what's in an image, neither can Now Assist.

  If the image is unclear in any way, Now Assist returns an error and cannot generate a preview. This error is also displayed if an image is not a business process.

- When uploading an image, it is the primary input by default. You can use the text description to add more context about the business process in your uploaded image.

> **Note:**
> Since an image is the primary input, you will receive an error for a unusable or unclear image even if you include a valid text input.

Don't use a nested process.

Nested processes are not currently supported for image to playbook generation.

Make sure activities don't reference previous stages.

Processes with activities that reference previous stages are not currently supported for image to playbook generation.

Don't use decision activities.

Decision activities are not currently supported for image to playbook generation.

Don't use triggers.

Triggers cannot currently be created for playbook generation.

Don't use variants.

Playbook variants are not currently supported for image to playbook generation.

Don't use pictures that contain pictures.

Pictures that contains pictures are not supported.

Make sure your image is unencrypted.

Encrypted images are not currently supported for image to playbook generation.

Don't use loops in the process.

Processes that contain loops or have arrows that go back are not currently supported.

Don't use multiple starting points.

Processes with multiple start nodes are not currently supported in Workflow Studio.

Make sure your activities and stages are connected properly.

Lines should clearly connect activities and stages.

Example text inputs

The following examples can help you to generate playbooks using only a text input:

Example playbook prompt 1: Employee onboarding

You can use this prompt to create a playbook for the onboarding process of new hires.

```
Create an employee onboarding playbook that integrates
 new hires into the organization.
HR initiates it upon job offer acceptance, gathering
 documents, assigning mentors, providing orientation,
setting up IT access, ensuring compliance, and
 job-specific training. The playbook ends with feedback
 from the employee and HR,
resulting in an onboarding checklist.
```

Example playbook prompt 2: Customer support

You can use this prompt to create a playbook with the steps that an agent takes for customer support cases.

```
Create a customer support playbook which is the primary
 point of contact for handling customer inquiries,
problems, and requests. It starts by receiving and
 categorizing customer inquiries based on urgency and
 complexity.
```

```
Support tickets are then assigned to agents who
 troubleshoot and resolve them, with the option to
 escalate to higher
support tiers if needed. After resolution, follow-up
 with the customer is crucial. A satisfaction survey
 gathers feedback for
improvement, and all interactions are documented,
 updating the knowledge base for future reference.
```

Example playbook prompt 3: Travel reimbursement

You can use this prompt to create a playbook to manage employee travel expenses.

```
Create a travel expense reimbursement playbook for
 managing employee travel expenses efficiently.
Employees submit expense reports with valid receipts.
 Finance verifies expenses against policies, ensuring
 budget compliance.
Managers approve expenses, initiating payment
 processing.
```

Example playbook prompt 4: Control document

You can use this prompt to create a playbook to manage a control document.

```
Create a playbook for generating, reviewing, and
 approving a control document. Suppliers and Quality
 managers collaborate
to upload documents and conduct appropriate refinement
 before a final document is approved.
```

Example playbook prompt 5: Supplier evaluation

You can use this prompt to create a playbook to evaluate whether potential suppliers meet qualification standards.

```
Create a playbook for qualifying a new supplier.
 Suppliers request qualification. Quality managers
 review and approve
qualification requests, plan qualification
 deliverables, execute qualification activities, and
 summarize qualification findings in a report.
The playbook concludes by determining if the supplier
 has achieved the requested qualification.
```

Example playbook prompt 6: Hardware inventory

You can use this prompt to create a playbook to manage hardware inventory.

```
Create a playbook that manages hardware inventory. This
 playbook should efficiently track stock levels, update
 in real-time,
and generate alerts for low inventory. It must support
 categorization, bar codes, and seamless integration
 with sales and procurement systems.
```

Note:
Generating or regenerating a playbook preview counts as 10 assists. To track your Now Assist usage, see .

Getting started with Process Automation

Learn how Process Automation applications can help you use the ServiceNow AI Platform® to transform your manual business processes into digitized, automated workflows.

ServiceNow Process Automation applications help you digitize, visualize, and manage the cross-enterprise workflows for your business. Digitizing your business processes with Process Automation applications gives you these benefits:

- Management of process compliance
- Ownership of continual process improvement
- Collaboration across divisions and departments
- Visibility into process outcomes

Process Automation applications

The ServiceNow Process Automation applications that you can use to digitize your business processes into automated workflows on the ServiceNow AI Platform include:

Flows

Flows enables process owners to automate approvals, tasks, notifications, and record operations without having to code.

Playbooks

Workflow Studio playbooks enable process owners to author cross-enterprise workflows and create a single, unified process. Build the underlying processes for playbooks that Playbook Experience agents and fulfillers use.

Roles involved in Process Automation

Depending on the Process Automation application that you use, you may need to coordinate with other teams or individuals to ensure that your automated processes run smoothly. One possible way in which you could organize the roles for Process Automation applications is:

- A developer uses Workflow Studio to work on flows, actions, and activity definitions to automate individual pieces of a business process.
- A playbook owner uses Playbooks to organize the pieces of the business process into a digitized, cross-enterprise workflow.
- A Playbook Experience administrator configures the appropriate views of the business process for the right system users in playbooks.
- A Playbook Experience agent, the end user in this case, works on and manages the individual tasks of the business process in playbooks on the ServiceNow AI Platform.

User roles

Many different user roles are involved in creating effective digitized, automated processes on the ServiceNow AI Platform.

Learn more about Process Automation

To find out more about how you can use Workflow Studio and Playbooks to digitize your business processes, try checking out these resources:

- Flow Designer fundamentals course
- Exploring flows
- Getting started with Playbooks

Playbooks across ServiceNow AI Platform®

Playbooks capabilities are available in other ServiceNow® products and for specific industries.

Connect applications

Subscribing to ServiceNow AI Platform® features and applications adds Playbooks content.

Note:
For activation information for integrations, see Activate playbooks.

Integration	Content available
App Engine Studio	Build playbooks from App Engine Studio.
Creator Studio	Create basic request and fulfillment apps without code.
Configure Playbooks for Customer Service Management	Create playbooks for Customer Service Management.
Field Service Management	Create playbooks for Field Service Management.
HR Service Delivery Playbook	Create playbooks for HR services.
Software Asset Management Guided Experiences	Create playbooks for Software Asset Management.

Integration	Content available
Healthcare and Life Sciences	Create playbooks for Healthcare and Life Science processes.
Playbooks for Financial Services Operations applications	Create playbooks for Financial Services Operations.
Playbooks for Public Sector Digital Services	Create playbooks for Public Sector Digital Services.
Run actions to resolve alert issues in Service Operations Workspace for ITOM	Create playbooks for IT Operations Management.
Exploring account onboarding	Create playbooks for Account Lifecycle Events.
Onboard Jira to DevOps Change Velocity — Workspace	Create playbooks for IT Service Management.
Security Incident Response playbooks	Create playbooks for Security Operations.
Working with Sourcing and Procurement Operations playbooks in the Source-to-Pay Workspace	Create playbooks for Procurement Service Management.
Complete publishing checklist and request policy approval	Create playbooks for Governance, Risk, and Compliance processes.

Playbook Experience

Guide agents and fulfillers through complicated playbooks from start to finish, improving customer experience and task resolution. Build your playbooks in Workflow Studio, and then design and embed your Playbook Experience in legacy workspace, UI Builder, ServiceNow Mobile Platform, Service Portal, and more.

Domain separation and Playbooks

Data separation is supported for Playbooks. The domain value of the triggering input record determines the domain context. 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 for Playbooks

The system domain separates Playbooks content according to these rules:

Playbooks content runs from the domain it is triggered from

Activities and Playbooks run from the domain of the record or user who initiates them. For example, when a user from the child domain ACME triggers a playbook belonging to the parent domain TOP, the playbook runs in the context of the child domain ACME.

Workflow Studio content runs in the domain of the calling Playbooks activity

Whenever an activity definition calls an action or flow, the system runs the Workflow Studio content in the same domain context as the triggering Playbooks content. If the Workflow Studio content has a matching domain-specific process override, then the system runs the override version instead. While Playbooks does not support process overrides, it uses any process overrides defined within Workflow Studio.

Note:

- A process override refers to a type of override in domain separation.
- Service providers (SPs) can use domain separation, to logically separate data, processes, and administrative tasks into defined groups.

Playbooks configuration files are visible to all domain users

Playbooks configuration files are not domain separated. Any user with the appropriate roles can view all playbooks, trigger definitions, and activity definitions.

Related topics

Domain separation for service providers

Exploring flows

Flows automate a repeatable multi-step process. When the flow trigger conditions are met, the flow runs a sequence of reusable actions and flow logic to complete the process.

https://player.vimeo.com/video/1007933352?badge=0&autopause=0&player_id=0&app_id=58479

Flows overview

A flow is an automated process consisting 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 **VTB Sample Flow** creates and assigns a VTB card whenever a priority 1 incident is created.

Flows consist of the following components.

Trigger

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. Workflow Studio supports a variety of trigger types such as record, Service Catalog requests, and scheduled triggers. For a description of available trigger types, see Workflow Studio flow trigger types.

Flow execution details

A flow execution details page allows a flow author to view run-time information about an action or flow directly from the design environment. You can view details

such as the current state, actions or steps run, output values generated, and errors produced. See Flow execution details.

Flow error handler

A flow error handler enables a flow to catch and report errors from the flow execution details. Run a sequence of actions and subflows to identify and correct issues. For example, have flows log output values, send notifications, and run corrective subflows when they produce an error. See Flow error handler.

Subflows

A subflow is an automated process consisting of a sequence of reusable actions, data inputs, and outputs. In contrast to flows, subflows do not have a trigger but instead run when called from a flow, from another subflow, or from a script. Building and managing subflows requires that you have some familiarity with the ServiceNow AI Platform tables and fields that the application or process uses. Process analysts can create subflows using available actions or use an existing subflow as a template. See Building subflows.

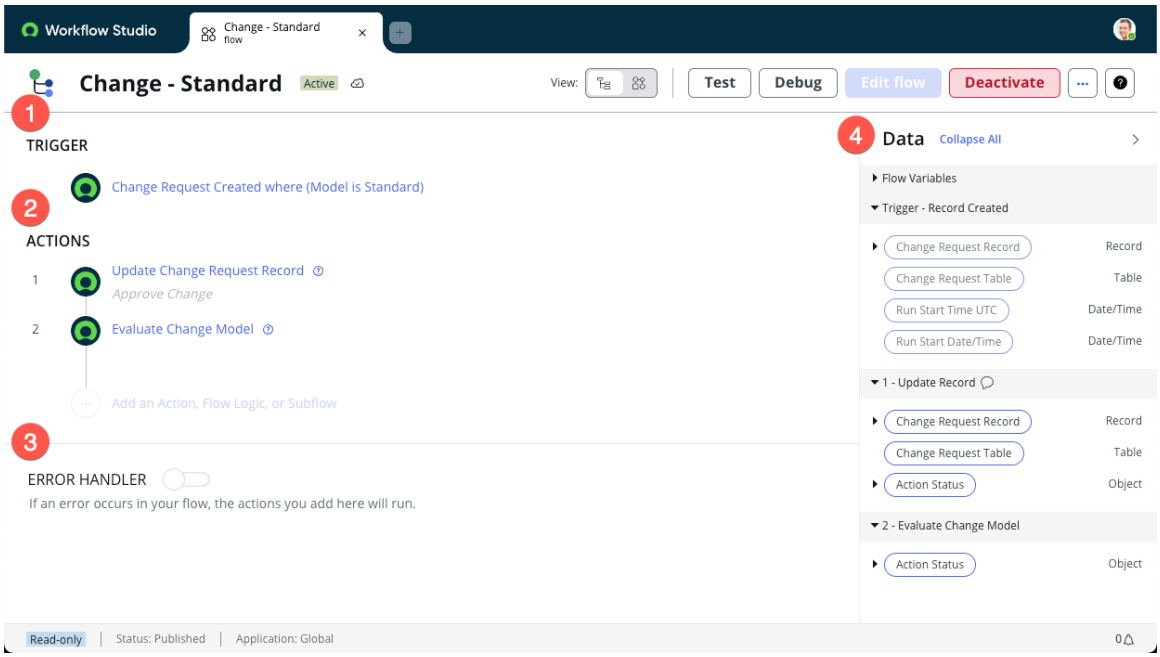
Actions

An action is a reusable operation that enables process analysts to automate ServiceNow AI Platform features without having to write code. For example, the **Create Record** action allows process analysts to generate records in a particular table with particular values when certain conditions occur. ServiceNow core actions like Create Record require some familiarity with ServiceNow AI Platform tables and fields. Action designers can create application-specific actions to pre-set configuration details. For example, creating a Create Incident Task action ensures that the process analyst uses the correct table and field configuration each time the action is used. You can add application-specific actions by activating the associated spoke. See Workflow Studio actions.

Spokes

A spoke is a scoped application containing Workflow Studio actions and subflows for managing specific tables. For example, the **ITSM Spoke** contains actions for managing Incident and Problem records. You can activate additional spokes from the ServiceNow Store or activating the appropriate plugin. Building your own spoke requires familiarity with application development on the ServiceNow AI Platform. For a list of available spokes, see Spokes.

Sections of a flow



A flow consists of four sections.

1. Trigger section

The Trigger section is where you select and define when the flow runs. Each trigger type defines when a flow starts and the starting data available to it. There are triggers for record operations, dates, and application operations.

Flows support record-based, schedule-based, and application-based trigger types. For more information about available trigger types, see Workflow Studio flow trigger types.

2. Action section

The Actions section is where you select and configure the steps that make up your flow. You can add actions, flow logic, and subflows to a flow by opening the associated picker.

Sequential numbers appear next to each item in the design canvas. Action numbers start with 1 and then increment by 1 as you add more items to the flow.

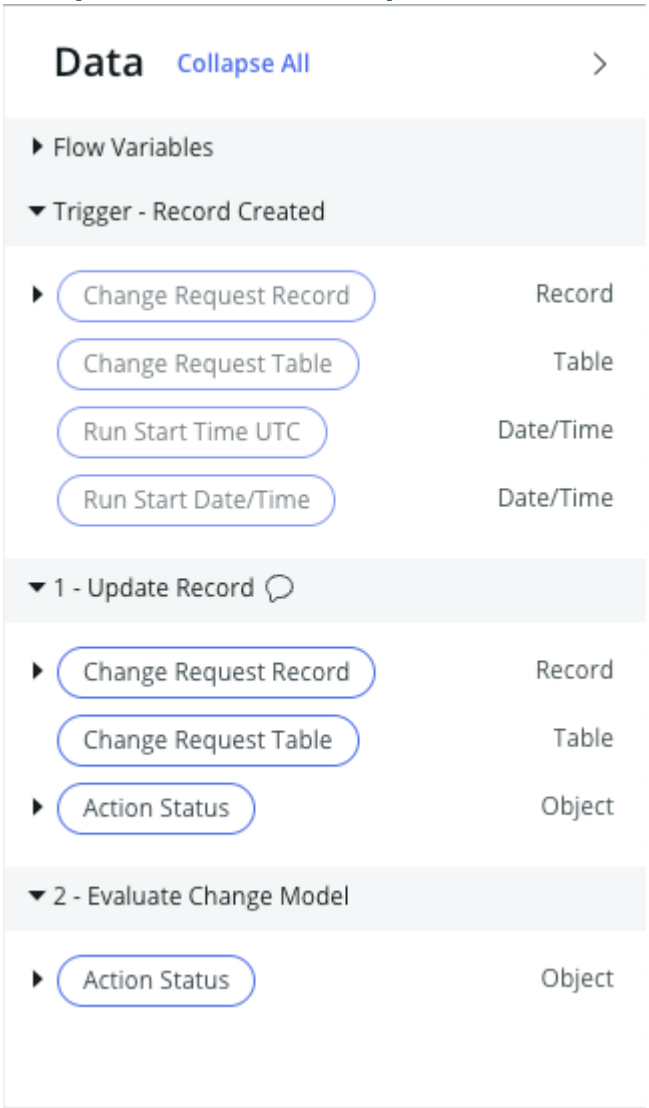
3. Error handler section

The Error handler section of the flows is where you can enable the flow to catch and correct errors. For more information about using the error handler section, see Flow error handler.

4. Data panel

The Data panel stores any data gathered or generated by the flow as variables. Each data variable has its own pill that Flow designers can use to drag the variable value to an action input or output. Flows generate the pill name based on the contents and its data type. The system specifies the variable data type next to the pill.

Example flow data in the data panel



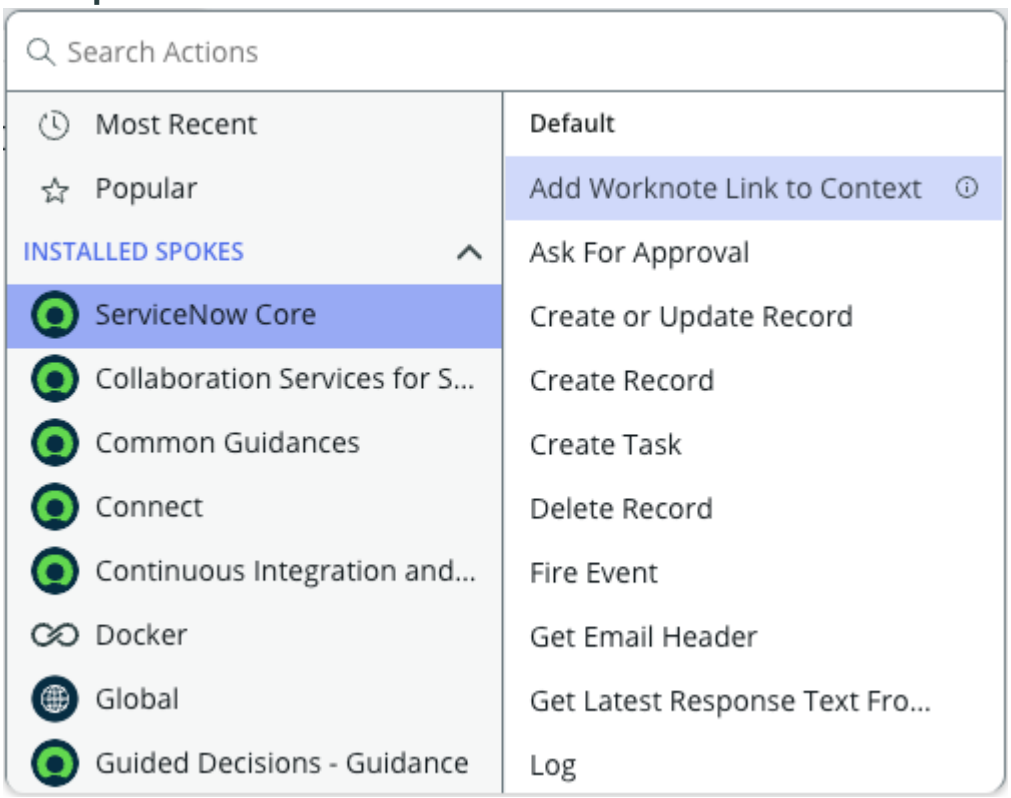
Action configuration

Add an action to a flow by opening the action picker. Configure an action by specifying its input values, which may be data from the flow trigger or the output of other actions. Enter values for inputs directly, or use data from the Data panel to configure an action's inputs. At the start of a flow, the data may be limited to the flow trigger or subflow inputs. As you add actions to a flow, the output values of each action appear as data pills in the Data panel.

Action picker

In the Actions section of a flow, select **Action** to open the action picker. You can either enter text in the search window to search for an action, or browse the action categories to find an action that you want to add to your flow.

Action picker menu



The Action picker consists of several features to make finding actions easier.

Search actions

The Action picker filters the list of actions it displays as you type. The picker only displays actions that match your search query.

Most recent

The Action picker displays a list of the most recent actions that you have selected during this session.

Popular

The Action picker displays a list of actions that other users have frequently used.

Installed Spokes

Some applications include spokes which add application-specific actions. Spoke actions are typically read-only but can be copied and customized. The Action picker groups actions by the application scope that they belong to. Select a spoke to see its list of actions.

ServiceNow Core actions

Your instance comes with a collection of core actions, or frequently used ServiceNow AI Platform operations, that can be added to any flow.

Global spoke actions

Developers may create custom actions in the Global application scope to make them available to all applications. ServiceNow provides some actions in the Global scope for use in any flow.

Custom actions

Developers may also create custom actions in an application scope that they own. Custom actions appear in the action picker under the application scope of the spoke.

Workflow to create flows using Workflow Studio

The following illustration describes the basic tasks involved in creating a flow using Workflow Studio. For detailed instructions for creating a flow, see Create a flow in Workflow Studio.

Flows benefits

Flows provide process owners and developers these benefits:

- Automates repetitive work to improve efficiency and experience.
- Describes a workflow in natural language to help non-technical users understand what it does.
- Displays a workflow as a diagram to help builders see available paths and connections.
- Enables creating and testing a workflow from a single interface to ensure it works as expected.
- Promotes process automation by enabling subject matter experts to develop and share reusable actions with flow authors.
- Reduces upgrade costs, with upgrade-safe ServiceNow AI Platform logic replacing complex custom script.
- Reduces development costs by providing a library of reusable actions.
- Scales with separate subscriptions for integration and Robotic Process Automation (RPA) functionality.

Benefit	Feature	Users
Build an automated workflow from an existing library of automated operations.	Flow	Application developer, process owner, or administrator
Run an automated workflow when a set of trigger conditions are met.	Flow	Application developer, process owner, or administrator
Use a set of trigger conditions as input data to run an automated workflow.	Flow	Application developer, process owner, or administrator

Getting started with flows

Create a sample flow with a trigger and base system actions that requires an approval.

Introduction to spokes, setting up an application, granting access, creating a flow, setting up an ATF test, and publishing the flow.

Watch this 11-minute video for an introduction to using Workflow Studio.

Before you begin
Role required: admin

Note:
While Workflow Studio is designed to use the flow_designer and delegated_developer roles in most scenarios, this tutorial uses the admin role to illustrate functionality without requiring additional roles to set up records and approve requests.

The ITSM application is required to access the Task table.

About this task

A flow can include these components:

- Trigger: An activity that initiates the flow, such as a record created in a specified table or a scheduled job.
- Conditions: Statements that determine when or how an action runs. For example, run an action only if a field is over a certain value.
- Actions: Operations executed by the system, such as a field value updated, approval requested, or a value logged.

To understand basic flows, create an expense approval flow. This flow:

1. Runs when an Expenses record is created.
2. Uses the total amount to determine which action to run.
3. Approves the request if it is under the specified dollar amount.
4. Requires manager approval if it is over a specified dollar amount. Another approver can be manually added.

Procedure

1. Create a custom application for the flow. Creating flows and actions within an application enables you to publish flows and actions to an application repository and deploy them on other instances. While this example does not use delegated development, you can optionally delegate action and flow designer development by assigning developers to the application.
   a. Navigate to **System Applications** > **Studio**.
   b. Create a custom application called **Expenses Getting Started**.
2. In Studio, create an Expenses table.
   a. Click **Create Application File**.
   b. Under Data Model, select Table and click **Create**.
   A Table form opens.
   c. Complete the form with the following values.
      - Label: **Expenses**
      - Extends table: **Task**
   d. Save the form.
   e. Add three additional columns to the table.

Column label	Type	Reference
Amount	Floating point number	None
Destination	String	None
Requested for	Reference	User [sys_user]

3. Add four records to the Expenses table to use in Workflow Studio tests.
   When you test your flow in later steps, you can specify which record is used as the trigger, enabling you to test specific record values.

   a. On the Expenses table record, click the **Show list** related link.

   b. Click **New**.

   c. Configure the form to add the **Amount**, **Destination**, and **Requested for** fields, and the **Approvers** related list.

   d. Complete the **Destination** and **Requested for** fields.
   Make sure that the user in the **Requested for** field in the test record has a manager assigned in the system. If the user in the test record does not have a manager, configure the User form to add the **Manager** field, and assign a manager to the user.

   e. In the **Amount** field, add a value under 100.00.

   f. Submit the form.

   g. Add another record to the table with an amount under 100.00.

   h. Add two more records to the table with a value over 100.00 in the **Amount** field.

4. In Studio, create a new flow.

   a. Click **Create Application File**.

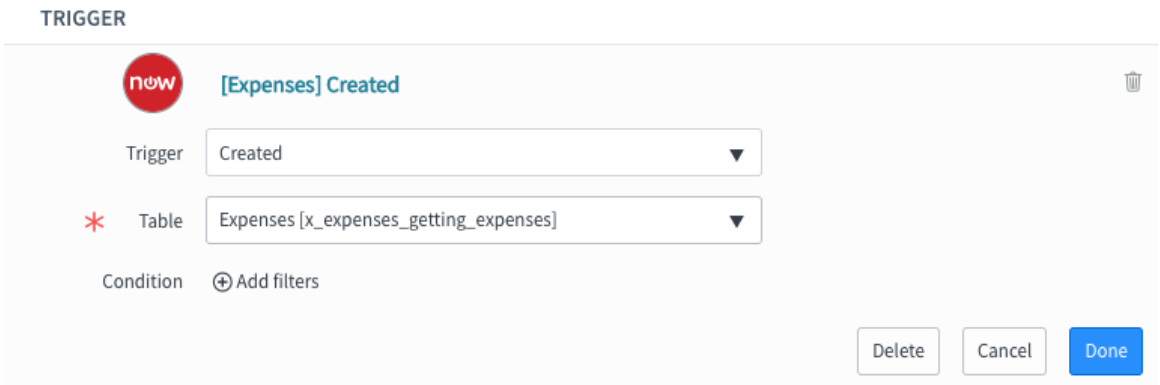
   b. Under Workflow Studio, select Flow and click **Create**.

   c. Select the **Flow** option, click **Next**.

   d. In the **Flow Name** field, enter `Expense Approval`.

   e. Click **Submit**.
   An Expense Approval flow is created in the Expenses Getting Started scope.

5. Create a trigger that runs the flow when a record is created in the Expenses table.
   - Trigger: **Created**
   - Table: **Expenses [x_expenses_getting_expenses]**



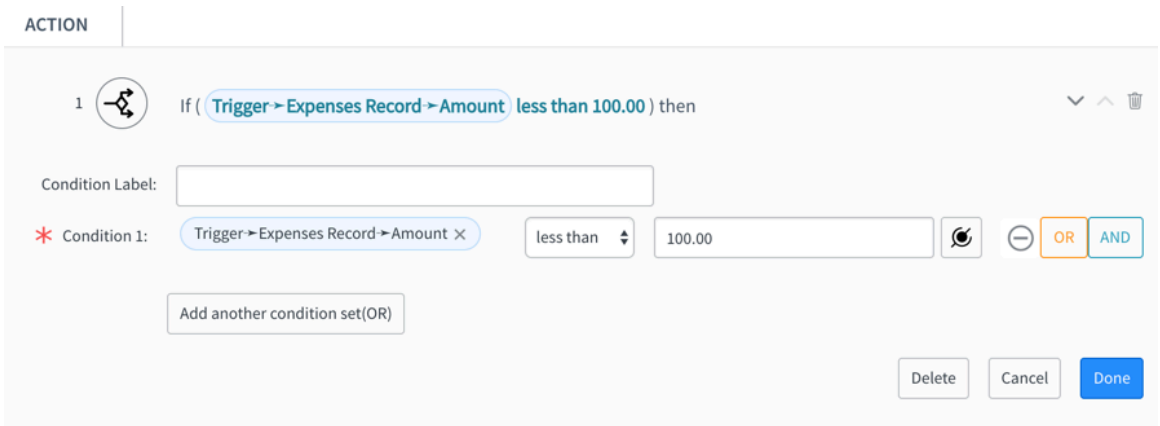
6. Add an if condition to the flow.

   a. Select **Flow Logic** > **If**.

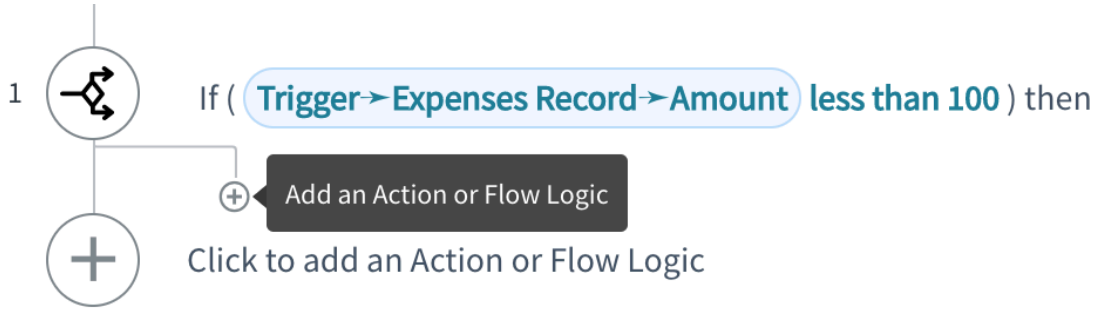
   b. In the right-hand pane, expand the **Trigger - Record Created** category and the **[Expenses Record]** pill.

   c. Drag-and-drop the **[Amount]** pill into **Condition 1**.
   A data pill represents the value of a record or a field at a particular stage in your flow. Dragging the **[Amount]** data pill from the trigger populates the condition with the value of the field in the triggering record.

   d. Set Condition 1 to **[Trigger->Expenses Record->Amount] [less than] [100.00]**.

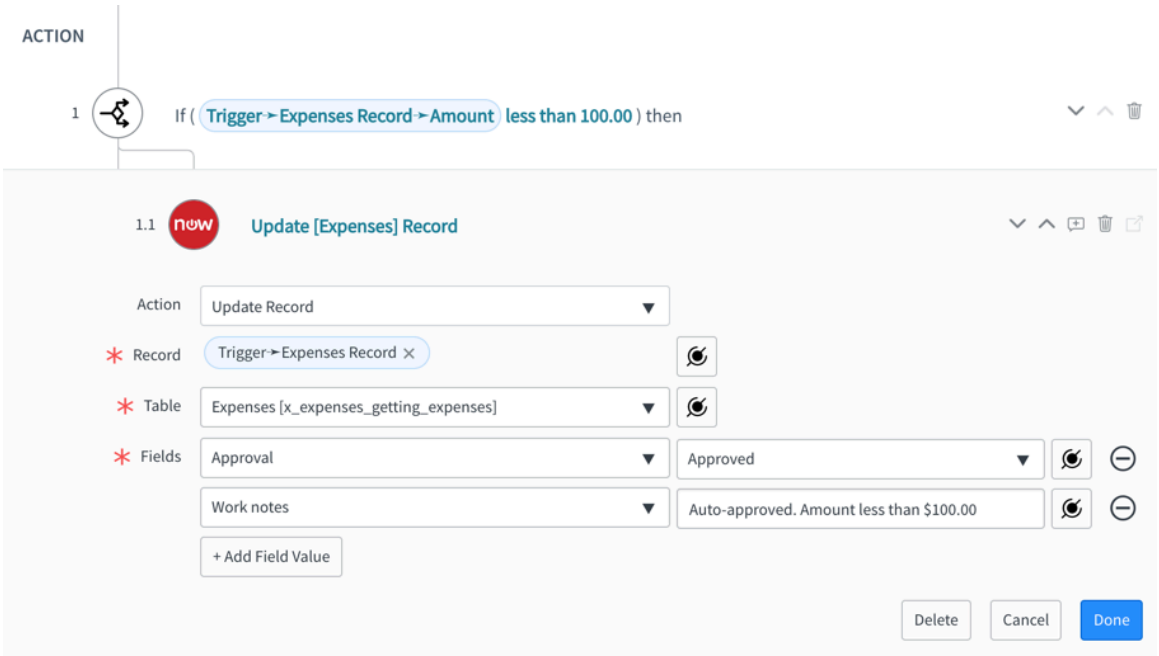


7. Underneath action 1, click **+** to add an action that runs when the If condition is met.



8. Create an Update Record action that approves the request.

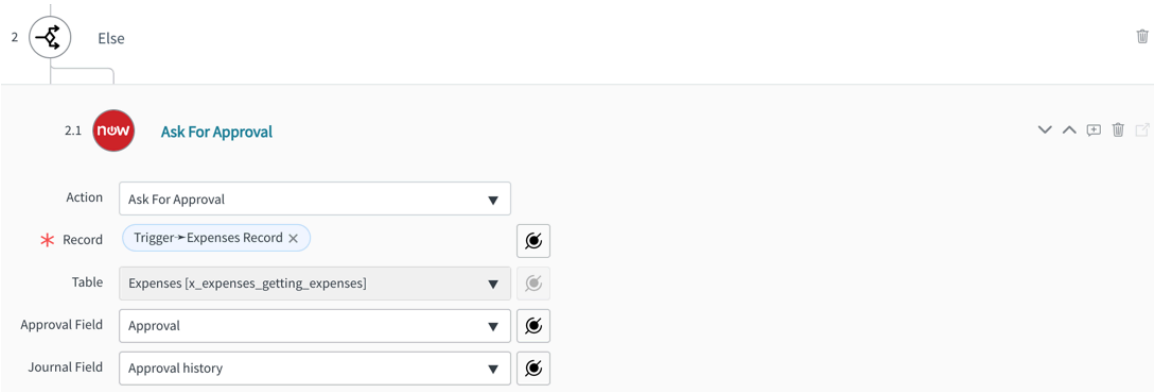
- Action: **Update Record**
- Record: Expand the **Trigger - Record Created** category and drag the **[Expenses Record]** data pill from the right-hand pane.
- Table: Set to **Expenses [x_expenses_getting_expenses]**.
- Fields:
  - Approval: **Approved**
  - Work notes: `Auto-approved. Amount less than $100.00`



9. Add an else condition to the flow.
   a. Select **Flow Logic** > **Else**.
10. Underneath action 2, click **+** to add an Ask for Approval action that runs when the Else condition is met.

    a. Complete the fields in the Ask for Approval step.

    - Action: **Ask for Approval**
    - Record: Expand the **[Trigger - Record Created]** category and drag the **[Expenses Record]** data pill from the right-hand pane.
    - Table: Set to **Expenses [x_expenses_getting_expenses]**.
    - Approval Field: Set to **Approval**.
    - Journal Field: Set to **Approval history**.

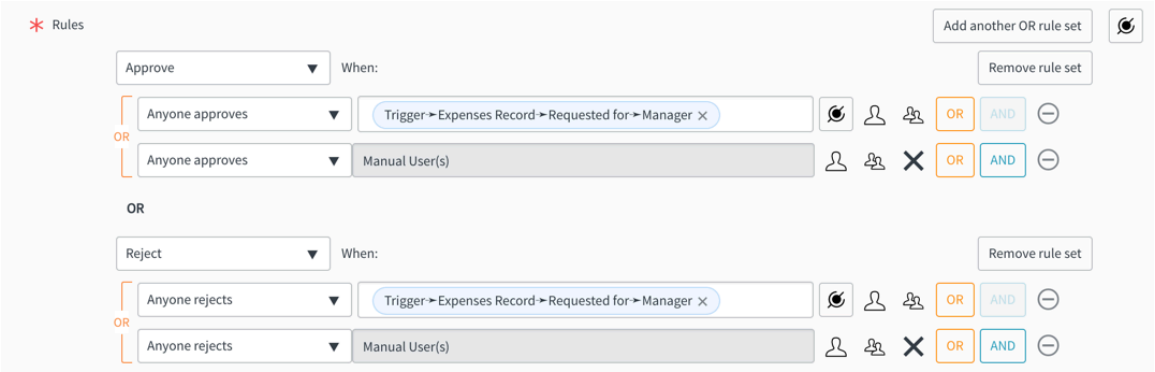


b. Define the rules in the Ask for Approval step.

- **[Approve]** when **[Anyone approves]** from the field **[Trigger->Expenses Record->Requested for->Manager]**, **[OR]**
- **[Anyone approves]** from the **[Manual User(s)]** list. Select Manual approvers to allow a manual approver to process an approval or rejection. A manual approver is a user manually added to the Approvers related list who can then approve the request. For example, you can manually add a subject matter expert to a task to approve the request. To learn more about adding manual approvers, see Generate approvals using the approvers related list.

Select **Add another OR rule set** to define rejection rules. When defining approvals, make sure to include rejection rules to avoid creating flows that remain in a waiting state if there are no matching approval rules.

- **[Reject]** when **[Anyone rejects]** from the field **[Trigger->Expenses Record->Requested for->Manager]**, **[OR]**
- **[Anyone rejects]** from the **[Manual User(s)]** list.

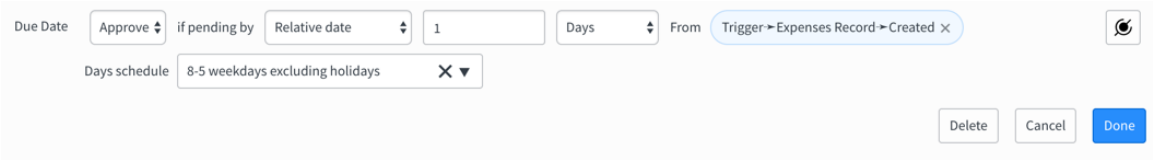


c. Define a due date to automatically approve, cancel, or reject an approval if the request is not approved or denied by the designated time.

Adding a due date ensures that the flow does not remain in a waiting state.

- **[Approve]** if pending by **[Relative date] [1] [Days]** from **[Trigger->Expenses Record->Created]**.
- Days schedule **[8-5 weekdays excluding holidays]**.

This due date automatically approves all requests that have not been approved or denied within one day from when the request was created.



11. Click **Save**.
12. Test the flow using a record with an amount below the designated limit.

    a. From the flow, click **Test**.
    The Test flow modal appears.

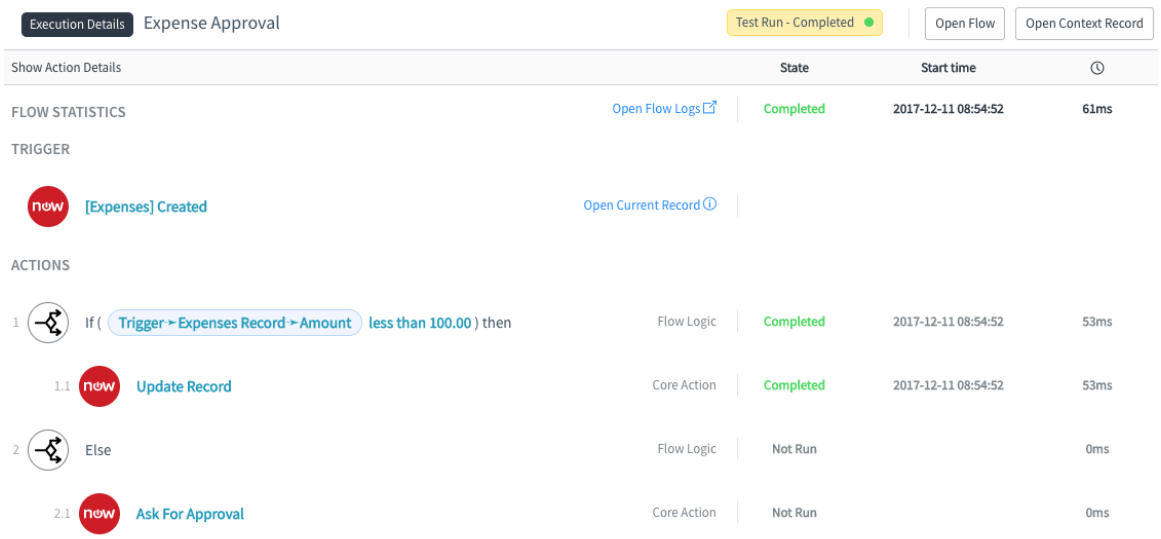
    b. In the **Record** field, select a record you created in earlier steps that has value in the **Amount** field under the 100.00 limit.
    This field is a reference to the table defined in the trigger.

    > **Note:**
    > Testing a flow bypasses the trigger conditions and immediately runs it. To test a flow with a record-based trigger, you must select a specific record to act as the trigger.

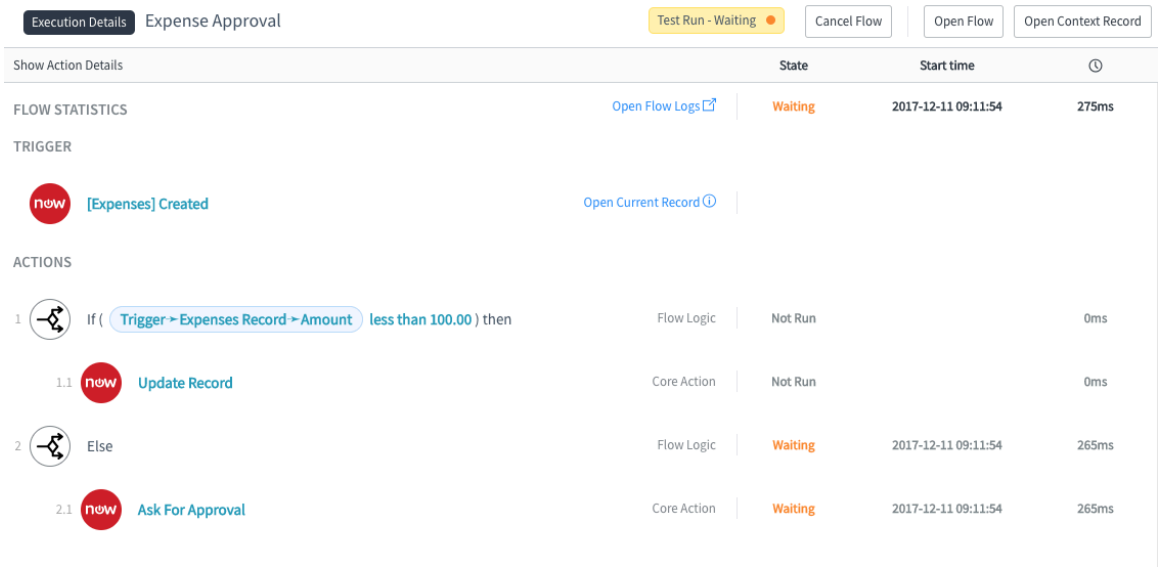
    c. Select **Run Test**.

    d. After the flow executes, click **Flow has been executed. To view the flow, click here**.
    The Execution Details open.

Because the amount is less than 100.00, the first condition is met and the request is approved.



13. Navigate back to the flow and run the test again using a record with an amount over the designated amount.
14. After the flow executes, open the flow Execution Details.
    Because the amount is over the designated limit, the request must be approved. Until a manager or a manual approver approves the request, the state is **Waiting**.

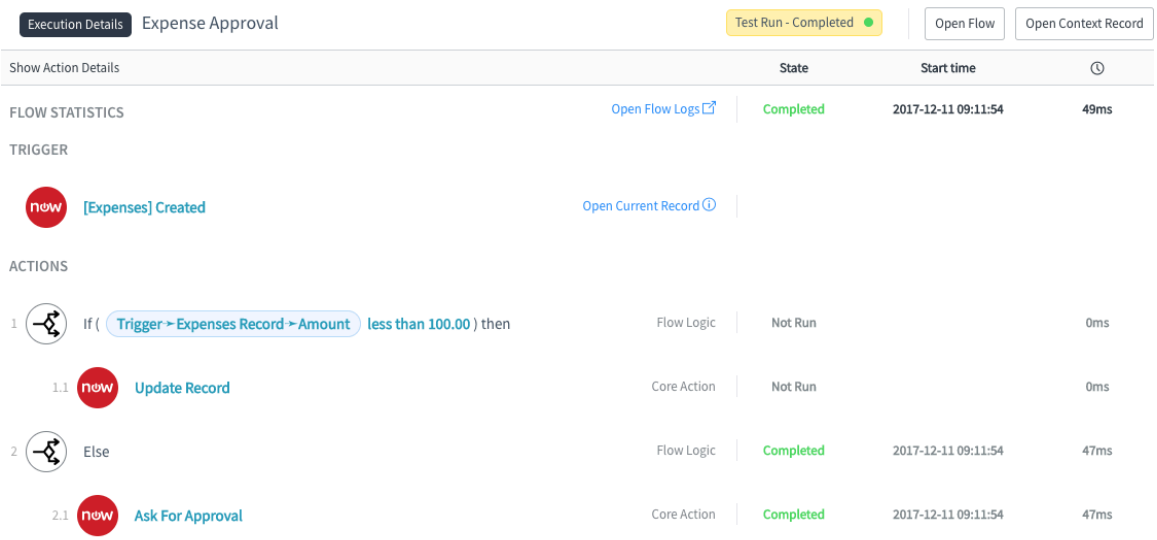


15. Approve the request.
In an active flow, a user from the Approvers list would approve or reject the request. However, because the flow is being tested, an admin can approve the flow.

a. Navigate to the test record.
The associated manager appears in the Approvers related list with **Requested** in the **State** field. Alternatively, you can edit the list to add manual approvers.

b. Change the value of the **State** field in the Approvers related list to **Approved**.

c. Navigate back to the flow Execution Details and refresh the browser.
Because the request is approved, the flow completes.



What to do next

Transform the Ask for Approval action into a reusable action using Workflow Studio. Actions enable flow designers to add complex actions to multiple flows with minimal configuration. See Getting started with actions.

Build your first flow in Workflow Studio

Step through an example of how to build, test, and activate a sample flow in Workflow Studio.

Before you begin

- Role required: admin, flow_designer, or delegated_developer
- Make sure to familiarize yourself with any features that your business uses to automate operations on the ServiceNow AI Platform, such as flows, subflows, and actions.

About this task

To help you get started with building your first flow in Workflow Studio, follow along with the steps below. The example flow will start, or trigger, every time a user on the instance creates a request for a Service Catalog item. When a request is created, our flow will automatically run the following actions:

- Check if the catalog item's price is greater than $1,000.
- If the price is greater than $1,000, notify the requester's manager to approve the request.
- Otherwise, if the price is less than or equal to $1,000, automatically approve and close the request.

Procedure

1. Use the filter navigator to go to **All** > **Self-Service** > **Service Catalog**.
2. From the Top requests box, select **Standard Laptop** and complete the record producer.
   The system produces a Requested Item record that has a price of $1100 or more (RITM0010001).
3. Use the filter navigator to go to **All** > **Process Automation** > **Flow Designer**.
4. On the Workflow Studio landing page's main header, select **New** > **Flow**.
5. In the Flow properties window, fill in the following fields:

Field	Action
Flow name	Enter `Approval flow for requested items`

   Then select **Submit** to open your flow in the Workflow Studio design environment.
6. Under the TRIGGER section, select **Add a trigger**.
7. In the trigger picker, either enter `Service Catalog` in the search field, or locate the *`Service Catalog`* trigger under the APPLICATION category.
   Then, select the *`Service Catalog`* trigger to add this trigger to your flow. Later, when we test this flow, we can simulate firing this trigger by creating a new Service Catalog Item Request record.
8. Click **Done** to finish adding the *`Service Catalog`* trigger to your flow.
9. Under the ACTIONS section, select **Add an Action, Flow Logic, or Subflow**.
   Then, select **Flow Logic** to open the flow logic picker.
10. In the flow logic picker, select **If**
11. Next to the Condition 1 input, click the pill picker icon ( ) to open the dot-walker.
    The dot-walker lets you access data from the trigger in your flow. Later, when you add actions to your flow, you can also use the dot-walker to access data from those actions. You can use the

dot-walker to drill down into data that references other records in order to get the proper field placeholder value that you want to drop as a data pill in an action's input.

12. Navigate, or dot-walk, to **Trigger - Service Catalog** > **Requested Item Record** > **Price** and select **Price** to add this data pill to the Condition 1 input.
13. In the condition builder's next field, choose **greater than**, and then enter `1000` in the final field. Now, you've successfully set up a condition that will check if the price of the catalog item that a user requests is greater than $1,000.
14. Click **Done** to finish adding the *If* flow logic to your flow.
15. Under your *If* flow logic condition, select **Action**.
16. In the action picker, either enter `Ask For Approval` in the search field, or locate the *Ask For Approval* action by selecting **ServiceNow Core** > **Ask For Approval** under the Default subcategory.
17. Add the following values for the Ask For Approval action's inputs:

Input	Action
Record	Click the pill picker icon ( ) to open the dot-walker. Then, navigate, or dot-walk, to **Trigger - Service Catalog** > **Requested Item Record** and select **Requested Item Record** to add this data pill to the input.
Table	Leave as `Requested Item [sc_req_item]`.
Approval Field	Leave as `Approval`.
Journal Field	Leave as `Approval history`.
Rules	Leave the first field as `Approve`. Under When, select **Anyone approves** from the list. Then, click the pill picker icon ( ) to open to dot-walker. Navigate, or dot-walk, to **Trigger - Service Catalog** > **Requested Item Record** > **Opened by** > **Manager** and select **Manager** to add this data pill to the field.
Due Date	Leave as `None`.

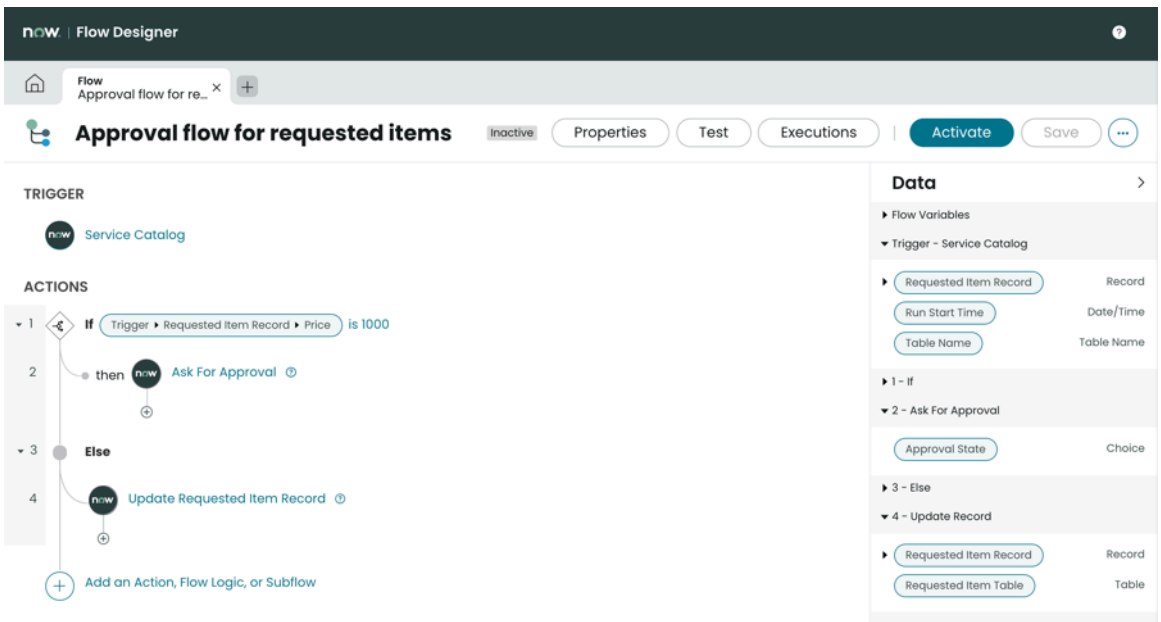
Now, you've successfully set up a conditional action that will automatically ask for approval from the requester's manager for any catalog item they request that has a price greater than $1,000.

18. Click **Done** to finish adding the conditional *Ask for Approval* action within the *If* flow logic of your flow.
19. Select **Add an Action, Flow Logic, or Subflow**.
Then, select **Flow Logic** and choose *Else* from the flow logic picker.
20. Under your *Else* flow logic, select the plus icon ( ) and then select **Action**.
From the action picker, select the **Update Record** action.
21. Add the following values for the Update Record action's inputs:

Input	Action
Record	Click the pill picker icon ( ) to open the dot-walker. Then, navigate, or dot-walk, to **Trigger - Service Catalog** > **Requested Item Record** and select **Requested Item Record** to add this data pill to the input.
Table	Leave as `Requested Item [sc_req_item]`.
Fields	Select **+ Add field value**. Then, select the **Approval** field and choose **Approved** as the field's value. Next, select the **State** field and choose **Closed Complete** as the field's value. Finally, select the **Close notes** field and enter `Request automatically approved, as requested item's value is less than $1,000` for the field's value.

Now, you've successfully set up a conditional action that automatically approves and closes the catalog item request if the requested item's price is less than or equal to $1,000.

22. To finish adding the conditional *`Update Record`* action within the *`Else`* flow logic of your flow, click **Done**.
23. In the main header, click **Save** to save the changes you've made to your flow.
Your flow should look similar to the following example:



24. To test your flow and see if it triggers and runs properly without any errors, go to the main header and click **Test**.
25. In the Test Flow window, select a Requested Item record that has a price greater than $1000, then select **Run Test**.

Example
For example, select the requested item record that you previously created from the service catalog (RITM0010001).

You can select the Create new record icon (+) to create a new requested item.

26. Select **Your test has finished running. View the flow execution details.**
27. On the Execution Details page, see the values that populated for the trigger and each automated action in your flow.
    For more information, see Flow execution details.
28. Navigate back to your flow.
29. In the main header, click **Activate** so that your flow's trigger fires whenever a user on your instance creates a new Service Catalog Item Request record.

What to do next

Your flow is now active and will run whenever it's triggered. Next, you can manage your flow every time it runs by viewing your flow's execution history. To view this history, open your flow in the Workflow Studio design environment and click **Executions** in the main header. The resulting page shows you the state of completion for each flow execution as well as how long it took for each flow execution to run, or its runtime. To troubleshoot a flow execution for errors, select an execution from this list to open the Flow execution details.

Build a flow from a template in App Engine Studio

Step through an example of how to build, test, and activate a flow using a flow template in App Engine Studio.

Before you begin

Create an application in App Engine Studio. Once your application is built, you can use flow templates to create flows. For more information, see Building applications in App Engine Studio .

Role required: admin, flow_designer, or delegated_developer

About this task

To help you get started with building a flow from a flow template, follow along with the steps below. The example flow will start, or trigger, every time a user on the instance creates a request for a Service Catalog item. When a request is created, our flow will automatically run the following actions:

- Check if the catalog item's price is greater than $1,000.
- If the price is greater than $1,000, notify the requester's manager to approve the request.
- Otherwise, if the price is less than or equal to $1,000, automatically approve and close the request.

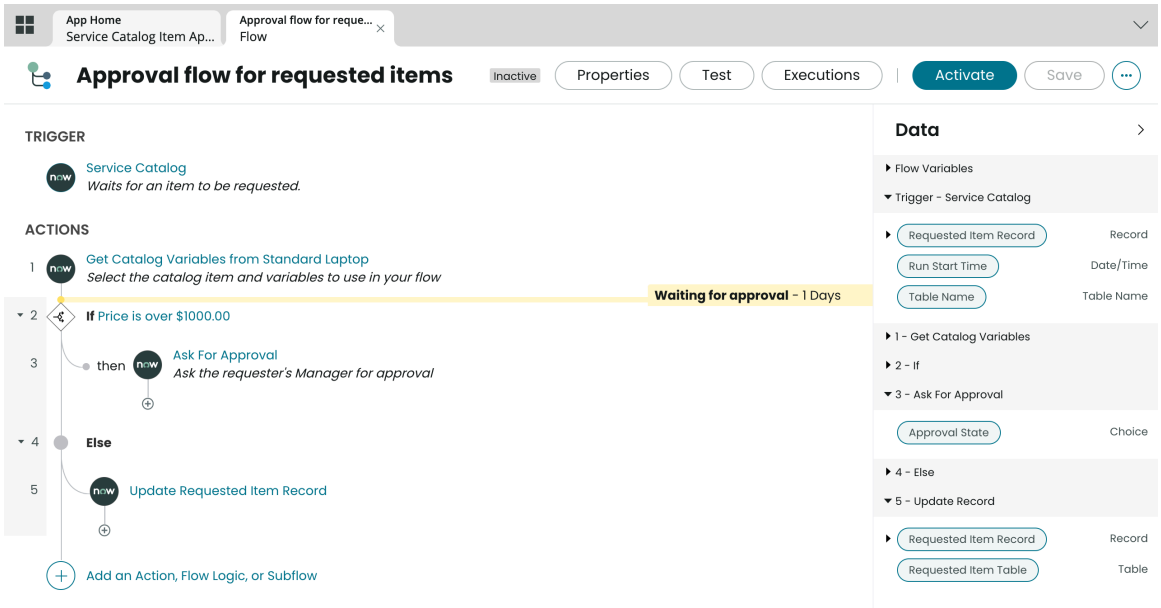
Procedure

1. Navigate to **All** > **App Engine Studio** > **App Engine Studio**.
2. From the My Apps page, open your application.
3. In your application, next to Automation, click the add icon (⊕).
4. From the gallery of automation templates, select **Create an approval for a requested catalog item**, then click **Begin**.
5. From the **Template catalog item** list, select the **Standard Laptop**.
6. In the **Ask for approval if the catalog item's price is greater than** field, enter `1000`, then click **Done**.
7. In the **Name** field, enter `Approval flow for requested items`.

8. In the **Description** field, enter `Approval flow for requested items`, then click **Continue**.
9. Once your flow is created, click **Edit this flow**.
10. On the flow page, click the **If not approved** step, then click **Delete** to remove the step.
11. At the bottom of the flow, click the **Log** step, then click **Delete** to remove the step.
12. In the same way, delete the last two log actions at the bottom of the flow.
13. Select **Add an Action, Flow Logic, or Subflow**.
    Then, select **Flow Logic** and choose *Else* from the flow logic picker.
14. Under your *Else* flow logic, select the plus icon (⊕) and then select **Action**.
    From the action picker, select the **Update Record** action.
15. Add the following values for the Update Record action's inputs:

Input	Action
Record	Click the pill picker icon ( ) to open the dot-walker. Then, navigate, or dot-walk, to **Trigger - Service Catalog** > **Requested Item Record** and select **Requested Item Record** to add this data pill to the input.
Table	Leave as `Requested Item [sc_req_item]`.
Fields	Select **+ Add field value**. Then, select the **Approval** field and choose **Approved** as the field's value. Next, select the **State** field and choose **Closed Complete** as the field's value. Finally, select the **Close notes** field and enter `Request automatically approved, as requested item's value is less than $1,000` for the field's value.

    Now, you've successfully set up a conditional action that automatically approves and closes the catalog item request if the requested item's price is less than or equal to $1,000.
16. To finish adding the conditional *Update Record* action within the *Else* flow logic of your flow, click **Done**.
17. In the main header, click **Save** to save the changes you've made to your flow.
    Your flow should look similar to the following example:



18. To test your flow and see if it triggers and runs properly without any errors, go to the main header and click **Test**.
19. In the Test Flow window, select a Requested Item record that has a price greater than $1000, then select **Run Test**.

    Example
    For example, select the requested item record that you previously created from the service catalog (RITM0010001).

    You can select the Create new record icon (+) to create a new requested item.
20. Select **Your test has finished running. View the flow execution details.**
21. On the Execution Details page, see the values that populated for the trigger and each automated action in your flow.
    For more information, see Flow execution details.
22. Navigate back to your flow.
23. In the main header, click **Activate** so that your flow's trigger fires whenever a user on your instance creates a new Service Catalog Item Request record.

What to do next

Your flow is now active and will run whenever it's triggered. Next, you can manage your flow every time it runs by viewing your flow's execution history. To view this history, click **Executions** in the main header. The resulting page shows you the state of completion for each flow execution as well as how long it took for each flow execution to run, or its runtime. To troubleshoot a flow execution for errors, select an execution from this list to open the Flow execution details.

Use the Workflow Studio help panel

Browse topics in the side help panel to learn more about building flows and actions, working with data and spokes, and stepping through guided tours in Workflow Studio.

Before you begin

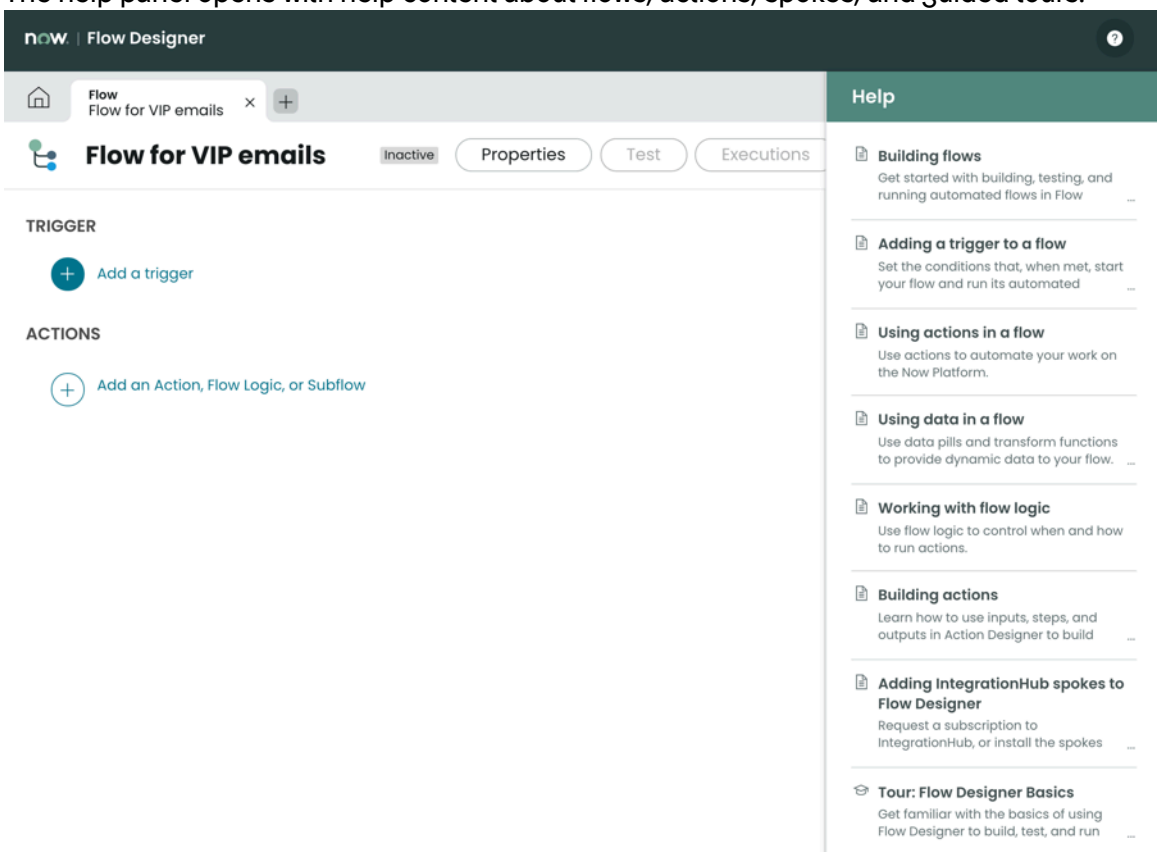
Role required: admin, flow_designer, or delegated_developer

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. In the Workflow Studio landing page's main header, select the question mark icon ( ) to open the help panel.
   You will see help topics that you can select and browse for more information about building flows and actions, working with data, and setting up Integration Hub spokes.
3. To close the help panel, select the question mark icon ( ) again.
4. Open any flow from the list of flows to go to the Workflow Studio design environment.
5. Next to the name for any action in the flow, select the Open help panel icon ( ).
   The help panel opens with information about the action and how to use the action's inputs and outputs in your flow. If no help content is available for your selected action, then the panel displays `No help content found for:` and the action's name.
6. You can also access guided tours from the help panel by selecting the question mark icon ( ) while in the Workflow Studio design environment.
   After opening the help panel, locate a help card beginning with `Tour:` and select that card to start the tour

Result

The help panel opens with help content about flows, actions, spokes, and guided tours.



Flow action numbering

The action outline displays a whole number besides each action and flow logic block in a flow. You can update flows containing legacy action numbering from within Workflow Studio.

Current action numbering

The current number sequence increments each item in the action outline by a whole value of one. For example, if a flow logic block is step 2 in the action outline, then the actions within the flow logic block are steps 3, 4, and 5. Inline scripts reference the whole number value of actions and flow logic.

Legacy action numbering

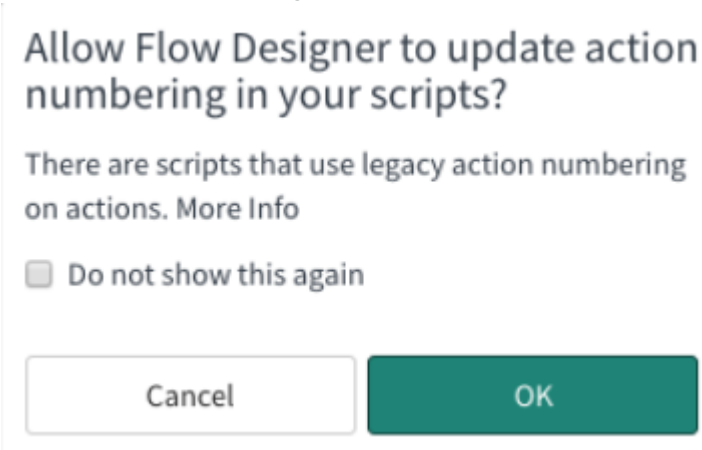
The legacy number sequence increments each item within a flow logic block by a decimal value of 0.1. For example, if a flow logic block is step 2 in the action outline, then the actions within the flow logic block are steps 2.1, 2.2, and 2.3. Inline scripts reference the decimal values of actions and flow logic.

Important:
Inline scripts produce an error when they refer to actions using legacy action numbering. Update all inline script references to the use the new action numbering.

Automatic action numbering updates

Workflow Studio automatically updates the action numbering of all flows during upgrade. Whenever you open a flow that contains inline scripts, Workflow Studio checks the script for references to legacy action numbering. If the script contains legacy references, it displays a prompt to update the action numbering.

Prompt to automatically update action numbers in inline scripts



The system attempts to match the actions referred to by the legacy action numbering to actions with the new numbering. Review the inline script changes to ensure that your inline scripts refer to the correct actions.

Note:
If an upgraded flow does not have the correct numbering, move the actions and flow logic to the correct sequence.

Flow Assist

Use Flow Assist to generate flows from text prompts and to generate recommendations for the next step of a flow. Flow Assist is part of the Now Assist for Creator application.

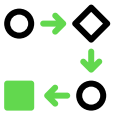
Get started

Explore

Learn more about what Flow Assist supports

Configure

Enable Flow Assist for an instance



Create flow outlines with generative AI

Reference

Learn about the roles required to use Flow Assist

Important:
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. Please check for availability updates in future releases.

Important:
Some Now Assist products/features are currently available only for customers in some regions. 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. 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 Flow Assist

Use Flow Assist to generate flows from text prompts and to generate recommendations for the next step of a flow. Flow Assist is part of the Now Assist for Creator application.

Overview

Flow Assist offers generative AI capabilities to flow authors.

- The Flow generation skill lets flow authors create or edit a flow from text prompts. See Flow generation for more information about this skill.
- The Flow recommendation skill provides flow authors with recommendations for the next step in their flow. See Flow recommendations for more information about this skill.

Supported versions

Flow Assist is supported on Vancouver Patch 2 and later releases.

Supported user interfaces

The Now Assist for Creator application supports Flow Assist skills in these user interfaces.

Flow Assist skills supported by interface

Interface	Skill supported
Workflow Studio properties for a new flow or subflow	Flow generation
Workflow Studio design environment	Flow recommendations

Flow generation

Create multi-step flows with generative AI. Generate appropriate data pill values for supported flow triggers and action inputs.

Activation

Flow generation is a skill installed with the Now Assist for Creator (sn_now_creator) application. You can install this application from the ServiceNow Store website.

Benefits

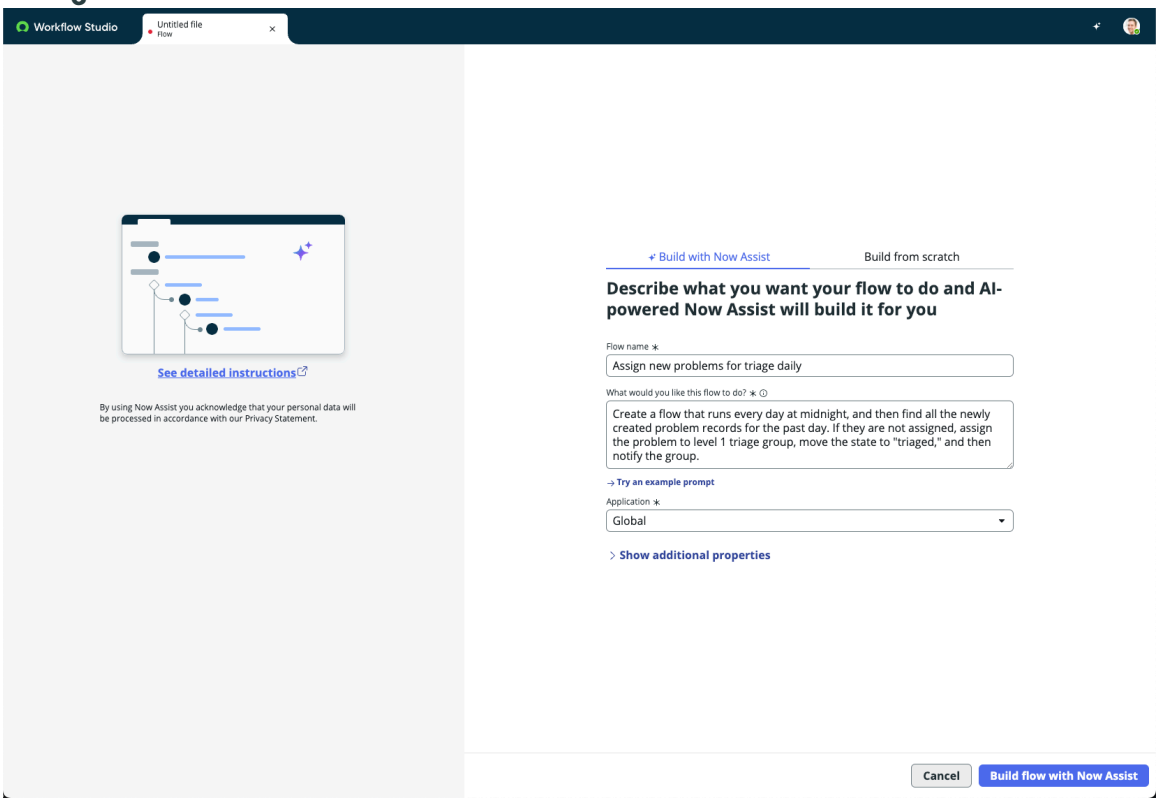
Enable flow generation to gain these benefits.

- Build a multi-step flow from a flow description in Workflow Studio. Supported triggers and actions are configured with appropriate data pill values.
- See the text directions used to generate each item in your flow.
- Use standard flow design patterns.

Supported user interfaces

Access flow generation skills from the Workflow Studio user interface.

Flow generation from the Workflow Studio user interface



Supported trigger types

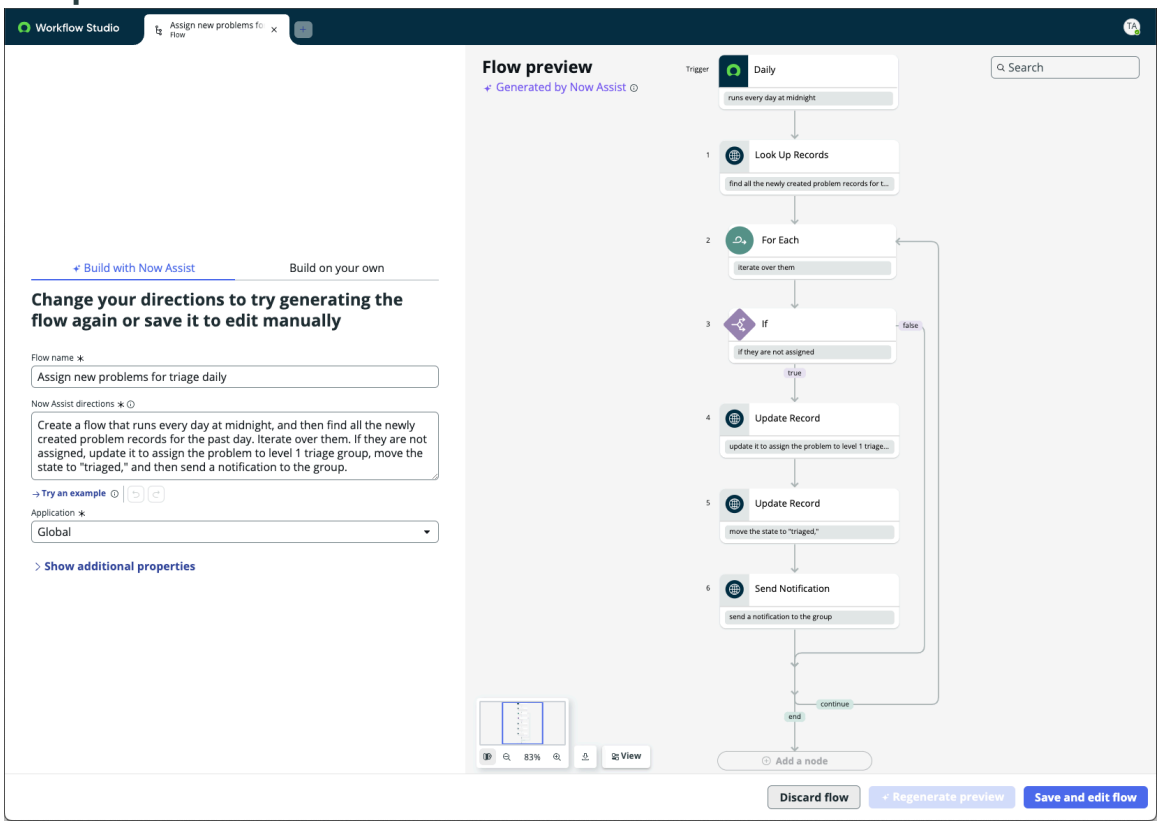
Flow generation supports a limited number of trigger types. You can use text directions to specify these trigger types.

- Scheduled triggers
- Record triggers
  - Created
  - Created or Updated
  - Updated
- Service Level Agreements triggers
- Inbound email triggers
- Service Catalog triggers

Annotations display directions used

Beginning with the Xanadu Patch 1 release, the flow generation skill displays flow annotations below each item added to the flow. The flow annotations contain the text directions that the flow generation skill used to generate the item. You can use these flow annotations to build better text directions and also receive feedback about how your directions map to specific actions, flow logic, and subflows.

Flow preview with flow annotations



This example illustrates how the flow generation skill mapped specific text directions to flow items.

Sample mappings of directions to flow items

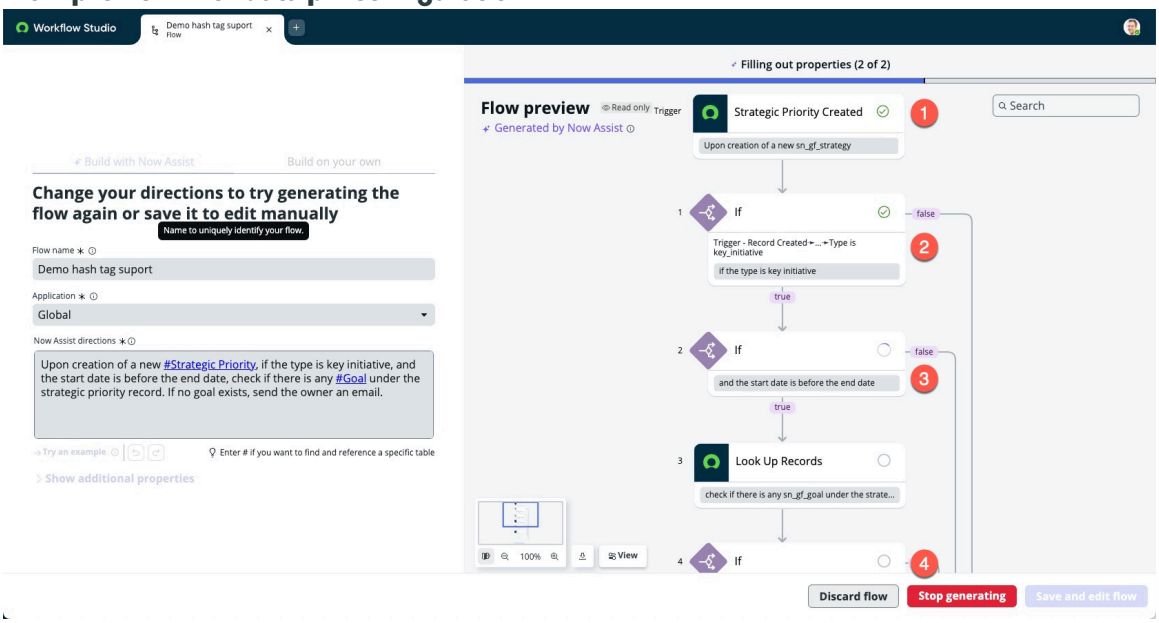
Directions	Flow item generated
`runs every day at midnight`	Daily trigger
`find all the newly created problem records for the past day`	Look Up Records action
`iterate over them`	For Each flow logic
`If they are not assigned`	If flow logic
`update it to assign the problem to level 1 triage group`	Update Record action
`move the state to "triaged,"`	Update Record action
`send a notification to the group`	Send Notification action

Data pill configuration for supported items

Beginning with the Xanadu Patch 3 release, the flow generation skill generates appropriate data pill values for supported triggers, actions, and flow logic. The flow generation skill updates the flow preview with data pill values as it generates them. While generating a flow preview, the system displays a check mark icon next to each item that contains generated data pill values. You can see the data pill values generated between the name of the flow component and the flow annotations containing your original text directions. The flow preview displays an animated

working icon next to items that are waiting for data pill values to be generated. While the flow preview is generating data pill values, the system displays an option to stop generating the flow preview. If you stop generating the flow preview, you must either manually save and edit the flow, or edit your directions to generate another flow preview.

Flow preview with data pill configuration



1. Completed icon

The system displays a green check mark icon on the cards of flow components for which it has generated data pill values. These status icons are only visible while the system is generating the flow preview.

2. Generated data pill values

The system displays a preview of data pill values in between the flow component name and the text directions used to generate the flow component. The system displays a preview of long data pill values. You can select a flow component to see the full data pill configuration in the properties pane.

3. Working icon

The system displays an animated working icon on the current card for which it is generating data pill values. The system displays a static working icon for the flow components that have not yet generated any data pill values. These status icons are only visible while the system is generating the flow preview.

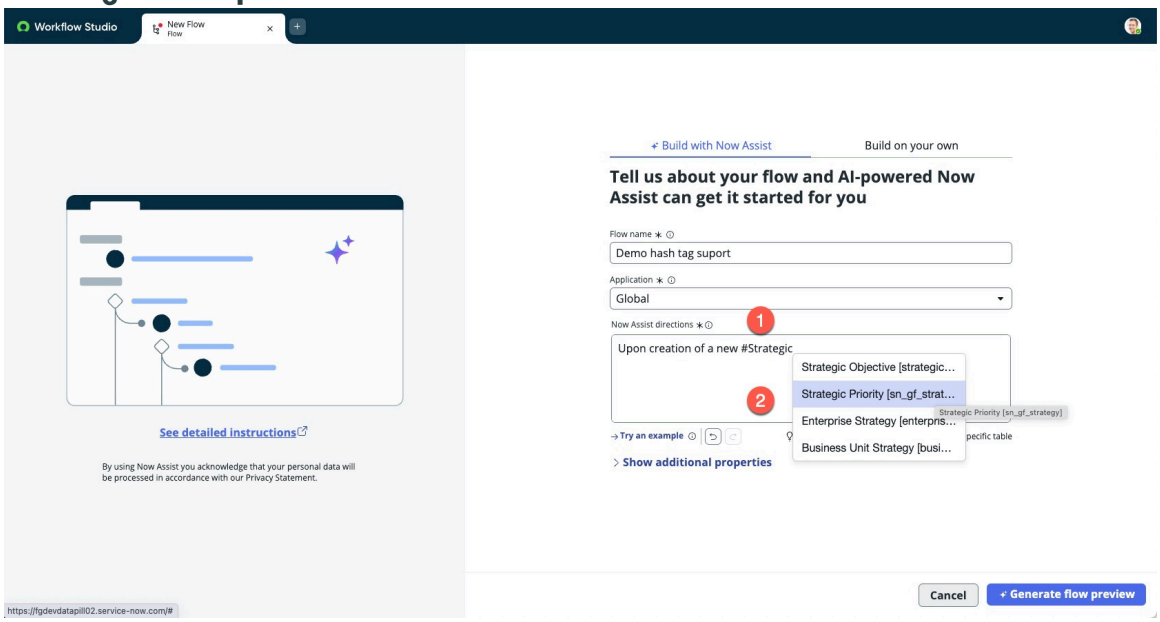
4. Stop generating button

The system displays the Stop generating button while it is generating data pill values. You can stop flow generation to either manually save and edit the flow or to update the Now Assist directions used to generate your flow preview.

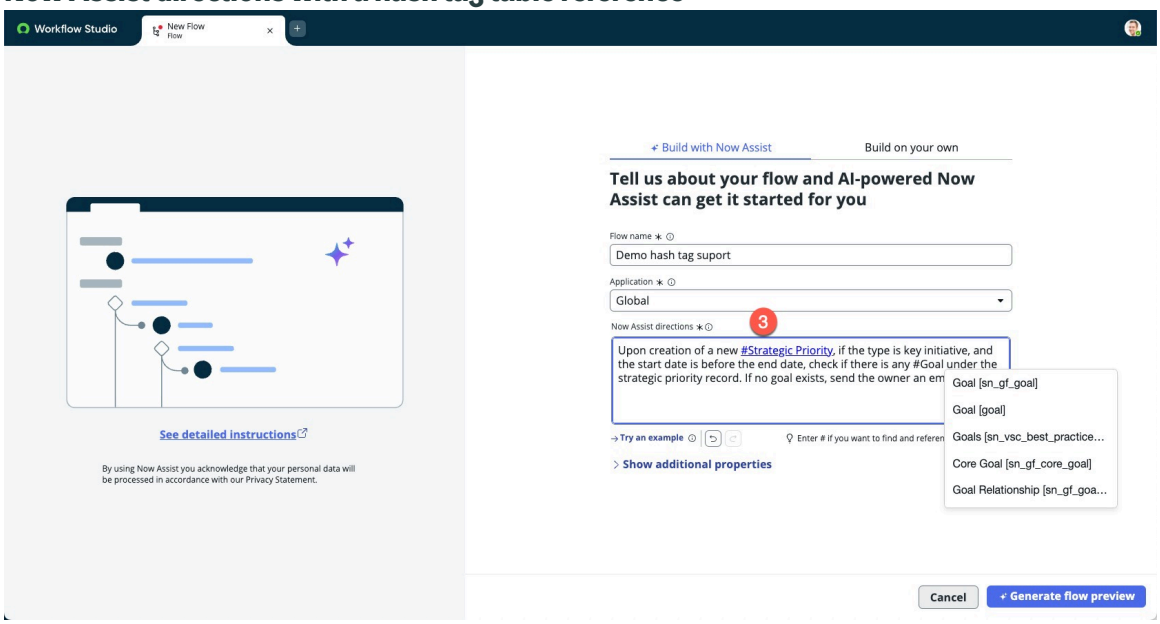
Hash tag support for table name references

Beginning with the Xanadu Patch 3 release, you can refer to a specific table in your Now Assist directions by typing a hash tag character and then typing either the table label or table name. As you type, autocomplete displays tables that match your entry. Add hash tag references to your directions when you want to ensure that flow generation selects the correct table for triggers or record-based actions. Hash tag references are particularly useful to help the LLM distinguish between tables that have similar or even identical labels such as the User [imp_user] and User [sys_user] tables.

Hash tag autocomplete



Now Assist directions with a hash tag table reference



1. Hash tag text

The system recognizes hash tags as a reference to a table label or table name. You can type either a table label such as Strategic Priority or a table name such as sn_gf_strategy.

2. Autocomplete table name suggestions

The system displays table name suggestions that match your hash tag text. You can select an autocomplete suggestion to use data from that specific table in your Now Assist directions.

3. Hash tag link to table

The system displays underlined link text to the table you selected.

Retrieval Augmented Generation (RAG) support

Flow generation uses Retrieval Augmented Generation (RAG) to include the names of common and recently published actions and subflows available on your instance. Workflow Studio updates the list of common and recently published actions and subflows every hour to make them available to flow generation. You can list published actions and subflows by name in your flow generation directions.

Example prompts

Here are some examples that you can use to create flows and subflows.

Example flow prompt 1: scheduled trigger

You can use this prompt to create a flow with a scheduled trigger.

```
Create a flow that runs every day at midnight, and then
find all the newly created problem records for the past
day. Iterate over them. If they are not assigned, update
it to assign the problem to level 1 triage group, move
the state to "triaged," and then send a notification to
the group.
```

Example flow prompt 2: Service Level Agreement (SLA) Task trigger

You can use this prompt to create a flow with a Service Level Agreement (SLA) Task trigger.

```
Create a flow with SLA trigger. Wait for 50% of the SLA,
then send a notification. Next, wait for 75% of the SLA
then send another notification. After 100% of the SLA is
complete, do an if statement to determine if the SLA is
breached or was completed.
```

Example flow prompt 3: Record trigger

You can use this prompt to create a flow with a record trigger.

```
Whenever a change request is created or updated where
model is unauthorized demo, do the following in parallel.
First, Apply change approval policy. If approvals are
approved or skipped, update change request record as
approved. If not, updated change request record as
rejected. Evaluate the model once again. If rejected,
send email. Second, Wait until active is false, disregard
change request approvals and evaluate the change model.
```

Example subflow prompt 4: Record input

You can use this prompt to create a subflow with a record input.

```
Create a subflow that logs the name of the problem input,
and then check if the last updated by person is the same
as the assigned to. Output the result from the subflow
output.
```

Example subflow prompt 5: Approval input

You can use this prompt to create a subflow for an approval.

```
Create a subflow to create a flow launcher job using
the given job config sys id and workload generator
parameters. If the job sys id is not empty, assign it
as the job exec id subflow output. Subsequently, start
```

```
the flow launcher for the created job sysid and assign
subflow outputs.
```

Example subflow prompt 6: Catalog tasks

You can use this prompt to create a subflow that creates catalog tasks.

```
Create a subflow that for every user with an assigned
laptop it sends an email stating that their operating
system has to be updated immediately and sends an SMS
with the instructions to do it.
```

These examples illustrate using hash tags to refer to specific tables.

Example flow 7: Strategic Priority [sn_gf_strategy] and Goal [sn_gf_goal]

```
Upon creation of a new #Strategic Priority, if the type
is key initiative, and the start date is before the end
date, check if there is any #Goal under the strategic
priority record. If no goal exists, send the owner an
email.
```

Example 8: Transfer Order [alm_transfer_order]

```
Once a #Transfer Order is updated to the stage shipment
preparation, check that there is a value in the to
stockroom field. If it is not empty, create a new
transfer order line and link it to the triggering
transfer order record.
```

Example 9: Assessment Metric type [asmt_metric_type]

```
Everyday at 7pm, check if there are new #Assessment
Metrics Type records that are live feed. For each of
them, if the pagination setting is category, then add the
assessor role to the assessment manager user.
```

Example 10: Test Suite [sys_atf_test_suite] and Test [sys_atf_test]

```
When a new #Test Suite record is created or updated,
check if there are any child #Test records within it. If
there are none, then set the Active field to false.
```

Placeholder steps

Flow generation inserts a placeholder step when it can't match part of your request to an available action or subflow. Placeholder steps don't perform any operations. They're empty steps that only display an annotation. Flow authors can use the placeholder text to select an appropriate replacement action or subflow.

Important:
You can't activate a flow that contains placeholder steps. You must either delete or replace each placeholder step with an action or subflow.

General guidelines

Follow these general guidelines when writing Now Assist directions.

Always describe the trigger first

Describe the flow trigger and its data conditions first. After the trigger, describe the actions and flow logic in the same order that you want them to be in the flow.

Avoid spelling errors

Avoid misspelling the names of actions, flow logic, or tables. Consider using hash tags to avoid making mistakes with table names.

Be precise and descriptive in your request

Make sure that your request is precise and descriptive. Describe the flow trigger, record data, actions, and flow logic in as much detail as you can.

Be succinct and direct in your request

Start by specifying whether you want to generate a flow or a subflow. For example, use the phrase, "`Create a flow that`" to generate a flow. Describe each step the flow in order.

Refer to actions, flow logic, and tables by name

Use action, flow logic, and table names as part of your directions. The closer your directions are to the actual names, the easier it is for the LLM to recognize them. For example, use the text `for each` or `do the following in parallel` to refer to those specific flow logic options. For table names, consider using hash tags.

Review the generated flow outline and input values

Review each action, flow logic, and subflow in the generated flow outline. Review the generated inputs values to confirm that they contain relevant data references.

Use hash tags to refer to data in a specific table

Use a hash tag to select a specific table name. Hash tags are particularly useful to distinguish between tables that have identical or similar display names such as the User [sys_user] and User [imp_user] tables.

Use numbers to distinguish the branches of do the following in parallel flow logic

Add a number to each parallel branch. For example, the directions, "`When a P1 incident is created, do the following in parallel: 1. Log its short description and 2. Look up the user assigned to it and send an email`," makes it clear that there are two branches.

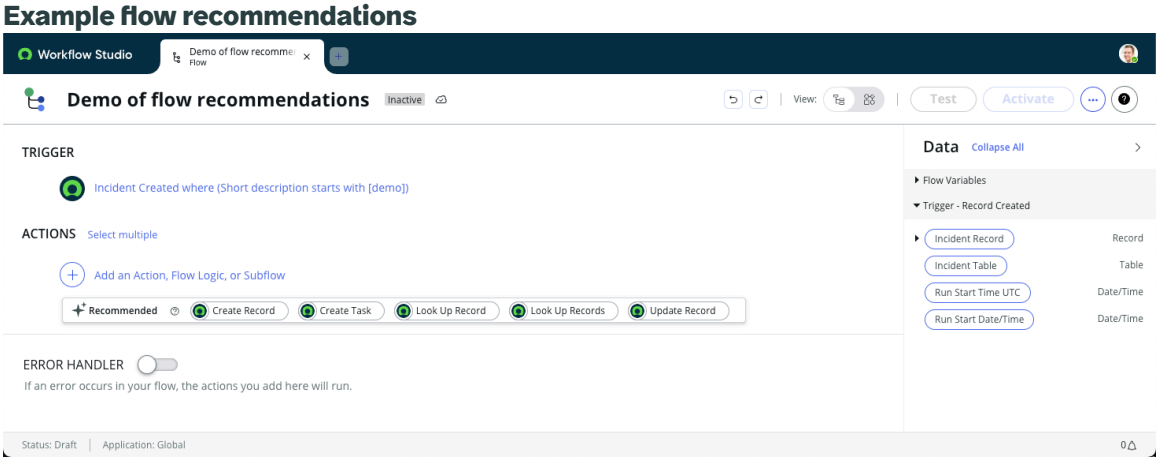
Use quotation marks to set exact values

Enclose exact data values in quotations marks to help the LLM distinguish between operation names and data values. For example, the directions, "`Log the value, 'incident reopened'`" make it clear that the text "incident reopened" is a data value.

Flow recommendations

Select the next component in your flow from a list of AI-generated recommendations. The system generates recommendations based on the current position in the flow and the flow component names listed before.

Example flow recommendations



The model uses the name of the flow components that come before to generate one to five recommendations for the next step of the flow. If there are no recommendations listed, then there are no flow components that meet the required relevance threshold.

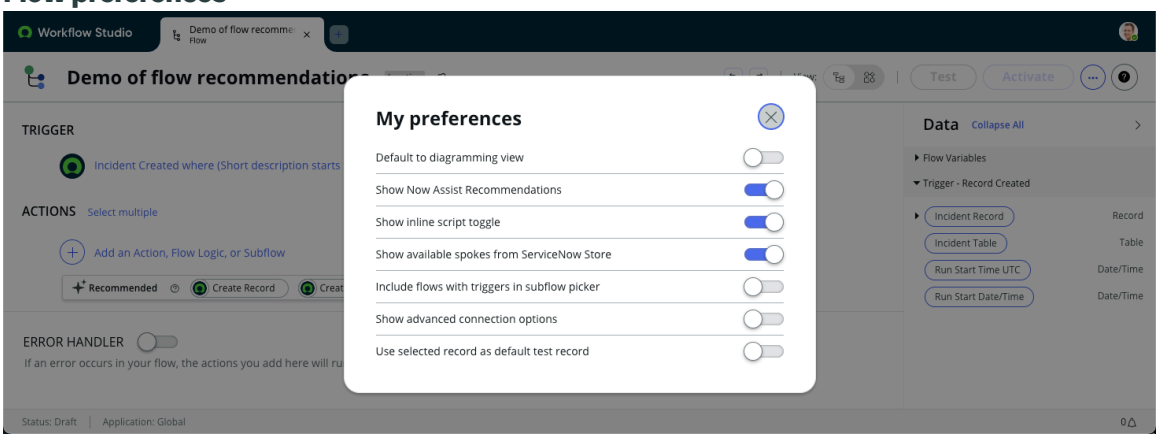
The system can only recommend actions, flow logic, and subflows that are available from ServiceNow. Recommendations can't include user-generated flow components such as custom actions, nor can recommendations include actions from ServiceNow Store spokes.

Generative AI model training

This Generative AI large language model was pre-trained with internal ServiceNow flows to learn flow creation patterns. The goal was to understand what flow components are most relevant for a certain position in a flow given the content before.

Flow preference

Flow preferences



By default, Workflow Studio shows flow recommendations as you build a flow. You can hide these recommendations on a flow by flow basis by turning off the Show recommendations flow preference. See User preferences for flows for more information.

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.

Configuring Flow Assist

Enable Flow Assist skills in the Now Assist for Creator application so that you can get started with building flows faster.

Install Flow Assist

Install the Now Assist for Creator application to add the Flow Assist generative AI capability.

Before you begin

- Role required: admin
- Review the Now Assist for Creator application listing in the ServiceNow Store for information on dependencies, licensing or subscription requirements, and release compatibility.

- Upgrade to Washington DC Patch 1 or later. For more information about this release, see Available patches and hotfixes.
- Enable Next Experience. For information about activating Next Experience, see Considerations for activating Next Experience.

Procedure

1. Navigate to the Now Assist for Creator application on the ServiceNow Store .

   Important:
   Now Assist for Creator requires a separate subscription.

2. From the Now Assist for Creator application page, select **Request App**.
3. After approval has been granted, on your instance, navigate to **All** > **System Applications** > **All Available Applications**.
4. Find the Now Assist for Creator application (sn_now_creator) using the filter criteria and search bar.
5. Select **Install**.

What to do next

Configure Now Assist for Creator to turn on the Flow Assist skills. Grant the now.assist.creator role to each user you want to use Flow Assist skills.

Turn on the flow generation skill

Turn on the flow generation skill to use generative AI to create and edit flows.

Before you begin

- Install the Now Assist for Creator application
- Role required: admin

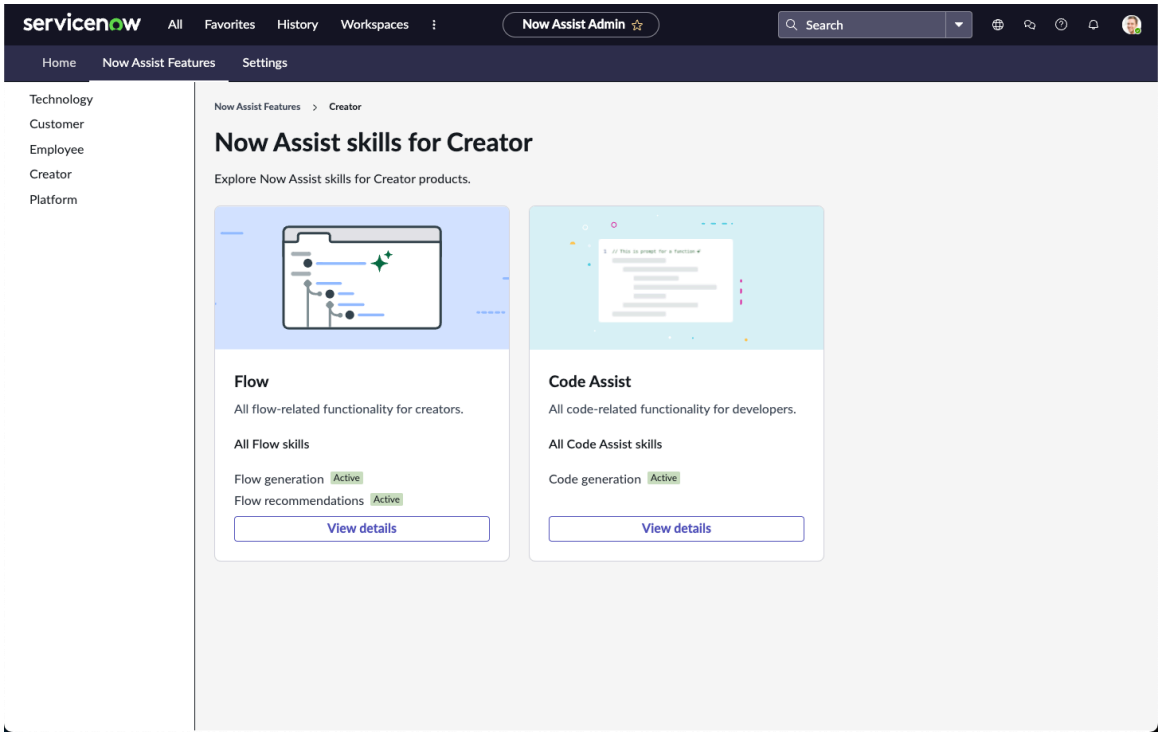
About this task

Important:
The Flow generation skill requires a separate subscription to Now Assist for Creator.

Procedure

1. Navigate to **All** > *Now Assist Admin* > **Features**.
2. In the workflow list, select **Creator**.
3. In the Flow Assist card, select **View details**.
4. Select **All available Flow Assist skills** > **Flow generation** > **Turn on skill**.

5. In the Flow Assist card, verify that the **Flow generation** skill is activated under **All Flow Assist skills**.



What to do next

Grant the now.assist.creator and flow_designer roles to each user who will use the flow generation skill.

Turn on the flow recommendations skill

Turn on the flow recommendations skill to get recommendations for the next step of your flow with generative AI.

Before you begin

- Install the Now Assist for Creator application
- Role required: admin

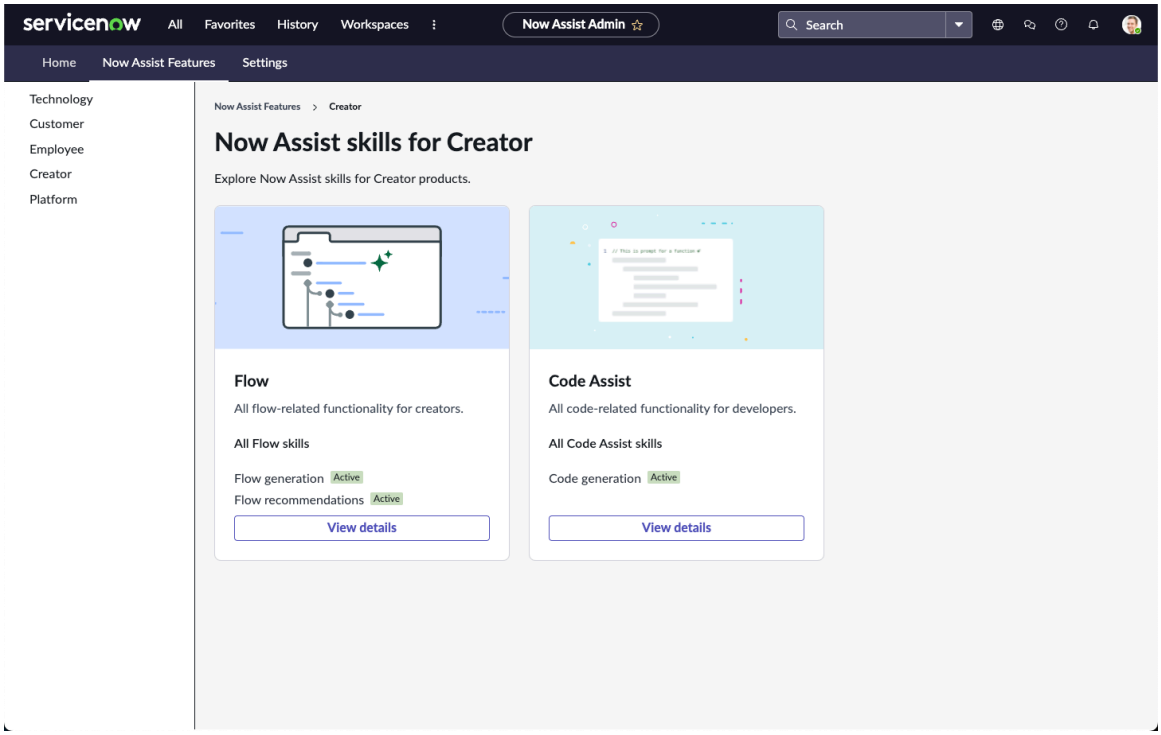
About this task

Important:
The Flow recommendations skill requires a separate subscription to Now Assist for Creator.

Procedure

1. Navigate to **All** > *Now Assist Admin* > **Features**.
2. In the workflow list, select **Creator**.
3. In the Flow card, select **View details**.
4. Select **All available Flow skills** > **Flow recommendations** > **Turn on skill**.

5. In the Flow card, verify that the **Flow recommendations** skill is activated under **All Flow skills**.



What to do next

Grant the now.assist.creator role to each user who will use the Flow recommendations skill.

Flow Assist reference

Reference topics provide additional information about configuration properties, roles, and more.

Flow Assist roles

The following roles are installed for use with the Now Assist for Creator flow generation and flow recommendations skills.

You can grant users entitlement to the applications that you purchase on the ServiceNow AI Platform by allocating subscriptions in Subscription Management. You allocate subscriptions by adding one or more groups with measured roles to a product subscription.

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]

This role grants users access to both the flow generation and flow recommendations skills of Flow Assist.

Groups

This role is assigned to no groups by default.

Contains Roles

This role contains no roles.

Elevated

This role isn't an elevated role.

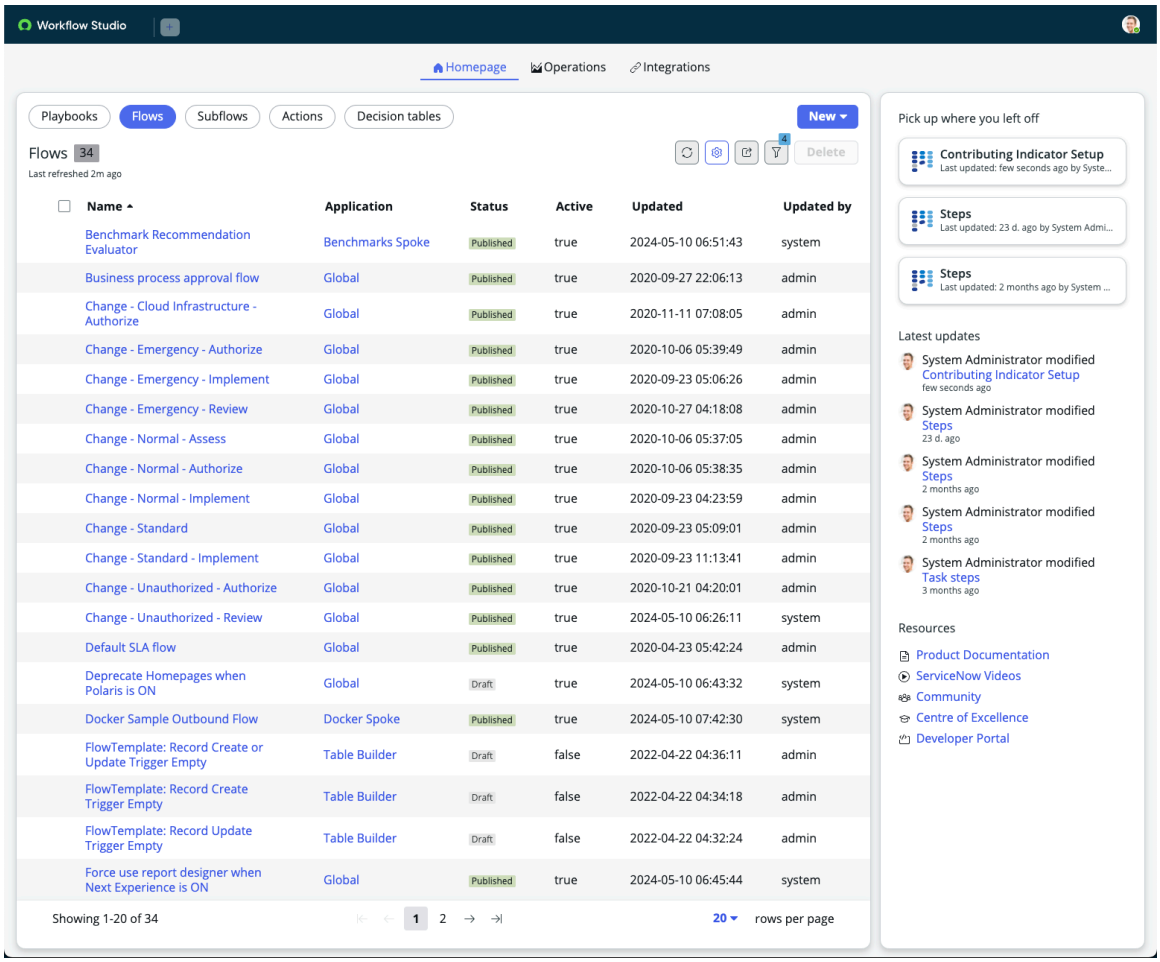
Special considerations

None

Workflow Studio Landing page

Access or create actions, flows, subflows, or their execution details.

The Workflow Studio landing page includes several sections and widgets to guide your work.



Available work by component type

This section displays a list of records for each component type of work.

Flows

Lists the available flows that you can work on. Selecting a flow opens the flow in a new tab. By default, this list displays the flow name, internal name, application scope name, publication status, active state, and update information. You can use the standard list controls to filter or personalize this list.

Subflows

Lists the available subflows that you can work on. Selecting a subflow opens the subflow in a new tab. By default, this list displays the subflow name, internal name, application scope name, publication status, active state, and update information. You can use the standard list controls to filter or personalize this list.

Actions

Lists the available actions that you can work on. Selecting an action opens the action in a new tab. By default, this list displays the action name, internal name,

application scope name, publication status, active state, and update information. You can use the standard list controls to filter or personalize this list.

Executions

Lists the execution details of the actions, flows, and subflows run. By default, this list displays the created date, name of the action, flow, or subflow run, run state, runtime duration, created by, and sys_id information. You can use the standard list controls to filter or personalize this list.

Connections

Lists the available connection records that you can view or edit for Integration Hub spokes.

All Work filter

Lists filter options for available work. Choose between **Created By Me** or **All Work**. The Created By Me option only displays Workflow Studio components or execution details that you created. The All Work option shows all Workflow Studio components or execution details that you have permission to view or edit.

Create New

This section includes links to create a flow, subflow, or action. The available choices depend on your user role, developer permissions, and feature access settings. You can only create components that you have permission to create. See User access to Workflow Studio flows for information about giving or limiting access to Workflow Studio components.

Pick up where you left off

This section lists the Workflow Studio components that you last created or edited. Select a component to continue working on it.

Execution reports

This section shows two reports.

Today's flow executions

This report lists the execution details created today.

Today's most executed flows

This report lists the most run flow or subflow during the last 24 hours.

Recent Activity

This section shows an activity stream of Workflow Studio component creations and updates. The activity stream shows the user, the Workflow Studio component worked on, and how long ago the activity happened.

Resources

This section includes links to product documentation, support videos, community discussion, Integration Hub spokes, and developer courses.

Help Panel

This section is available when you select the question mark icon () to open the Workflow Studio help panel. You can browse topics and guided tours that can help you get started building flows and actions. For more information, see Use the Workflow Studio help panel.

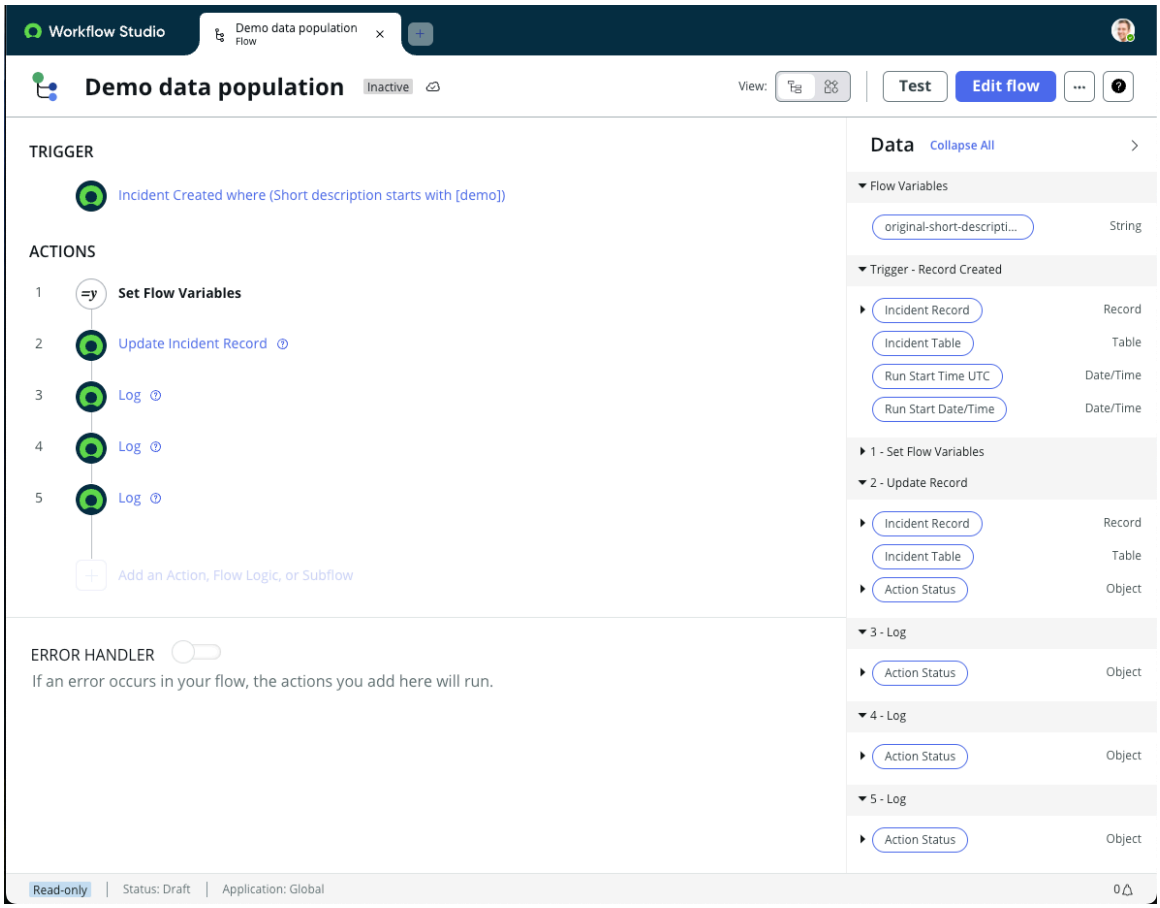
Workflow Studio data

Each time you add an action to a flow, Workflow Studio adds a data pill to store its results. The data pill name indicates its sequence in the flow and its data type.

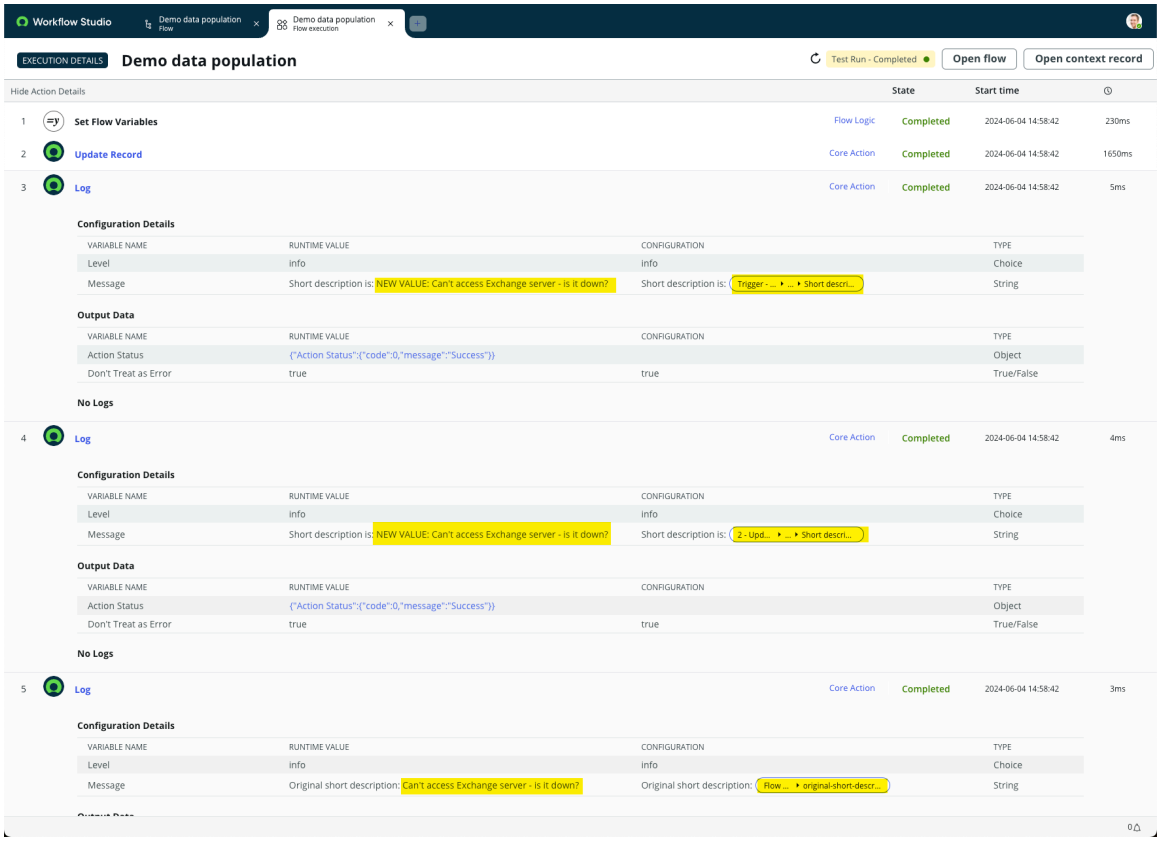
Flow designers use action result data pills to provide input for other flows, actions, or subflows. Flow designers can use the sequence value in the data pill name to ensure that they select the correct data pill as an input value. When a flow runs an action, it generates the data pill runtime value as it is used. For example, if a data pill for **[Trigger->Incident record]** gets populated with incident record values at the start of a flow, and then the incident record values are updated, the data pill reflects the updated real-time values for the rest of the flow.

Data pill population

Workflow Studio populates data pill values when the action, flow logic, or subflow that produces the data pill finishes running. If another action, flow logic, or subflow runs and changes the data pill value, then the new value is used for the rest of the flow. For example, suppose that you have a flow triggered by the creation of an incident record that performs the following actions.



1. **Set Flow Variables** flow logic to store the value of the **[Trigger->Incident Record->Short description]** in the **original-short-description** flow variable.
2. **Update [Incident] Record** action to add a text string to **[Trigger->Incident Record->Short description]**.
3. **Log** action to store the value of the **[Trigger->Incident Record->Short description]** data pill.
4. **Log** action to store the value of the **[2->Incident Record->Short description]** data pill.
5. **Log** action to store the value of the **original-short-description** flow variable.



When you test this flow and view it's execution details, you can see that the runtime values reflect the actions and flow logic that were run. Action-1 stores the original incident short description in a flow variable called **original-short-description**. Action-2 changes the value of the **[Trigger->Incident Record->Short description]** data pill to add the prefix NEW VALUE: to the front of the string. Any other actions that use this data pill will use this new value. Action-3 logs the current value of the **[Trigger->Incident Record->Short description]** data pill, which has been updated by Action-2. Action-4 logs the value of the **[2->Incident Record->Short description]** data pill, which has the same value as the **[Trigger->Incident Record->Short description]** data pill. Both data pills store the incident short description as set by Action-2. Action-5 logs the value of the **original-short-description** flow variable set by Action-1.

Important:
Prior to Washington DC release, Workflow Studio populated all data pill values as soon as the data became available regardless of where the data pill was located in the flow sequence. In this example, the Action-3 data pill for **[Trigger->Incident Record->Short description]** would be set when the flow starts rather than using the updated the data pill value generated by Action-2. If the older behavior is desired, you can use flow variables to store data pill values as they are generated. For example, Action-1 stores the **[Trigger->Incident Record->Short description]**data pill in a flow variable called **original-short-description** before the data pill value is changed.

Data security and HTML sanitization

Workflow Studio protects against cross-site scripting and code injection by evaluating all string data for HTML markup. The system only preserves HTML markup that is present in its inclusion list. All other HTML markup is removed from string data.

The inclusion list supports these HTML elements and attributes, which cannot be modified.

HTML inclusion list

HTML element	Included Attributes
a	class, href, target, title
abbr	class, title
address	class
area	alt, class, coords, href, shape
article	class
aside	class
audio	autoplay, class, controls, loop, preload, src
b	class
bdi	class, dir
bdo	class, dir
big	class
blockquote	cite, class
br	class
caption	class
center	class
cite	class
code	class
col	align, class, span, valign, width
colgroup	align, class, span, valign, width
dd	class
del	class, datetime
details	class, open
div	class
dl	class
dt	class
em	class
emp	class
font	class, color, face, size
footer	class
h1	class
h2	class
h3	class
h4	class
h5	class

HTML inclusion list (continued)

HTML element	Included Attributes
h6	class
header	class
hr	class
html	
i	class
img	alt, class, height, src, title, width
input	aria-label, class, type, value
ins	class, datetime
li	class
mark	class
nav	class
ol	class
p	class
pre	class
s	class
section	class
small	class
span	class
sub	class
sup	class
svg	class
strong	class
style	
table	align, border, class, valign, width
tag	class
tbody	align, class, valign
td	align, class, colspan, rowspan, valign, width
tfoot	align, class, valign
th	align, class, colspan, rowspan, valign, width
thead	align, class, valign
tr	align, class, rowspan, valign
tt	class
u	class
ul	class

HTML inclusion list (continued)

HTML element	Included Attributes
video	autoplay, class, controls, height, loop, preload, src, width

Flow diagramming view

Create and view flows as diagrams. See the paths a flow can follow and the connections between elements.

Activation

Install Flow Diagramming from the ServiceNow Store website.

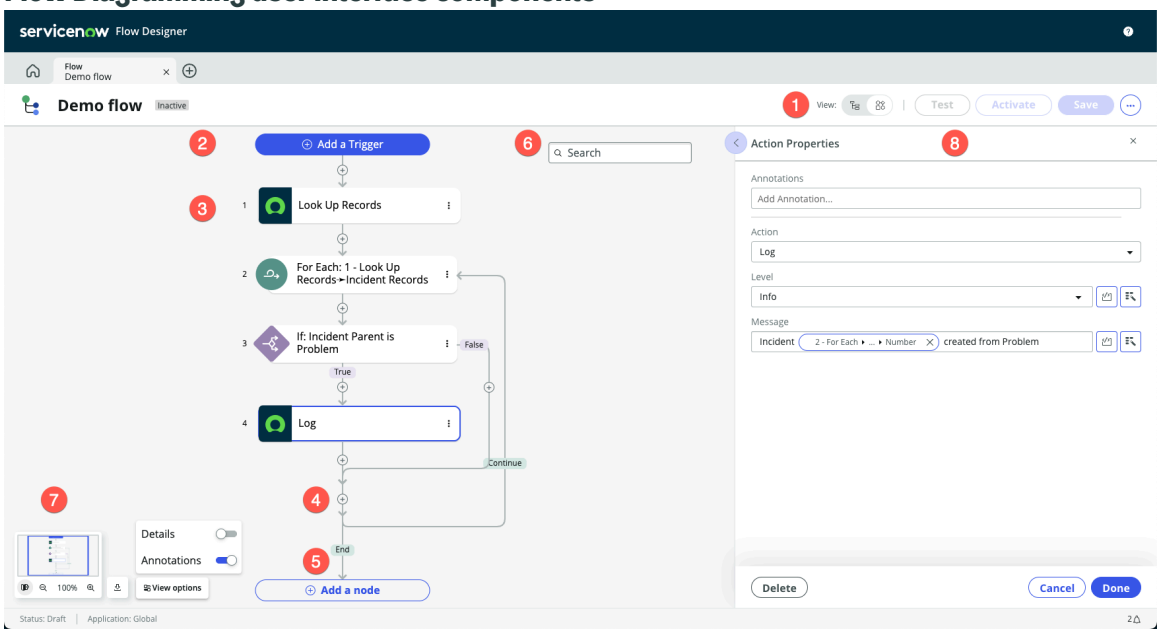
Benefits

Enable the flow diagramming view of a flow to gain these benefits.

- Add and edit Workflow Studio components within specific paths of a flow.
- See the branches and paths a flow can follow.
- See the relationships between Workflow Studio components.

Flow Diagramming components

Flow Diagramming user interface components



The Flow diagramming view consists of these components.

1. View selector

Switch between the flow diagramming view and the flow description view.

2. Add a trigger

Select and configure a flow trigger. Flow diagramming view does not support some trigger types.

3. Flow nodes

View and configure a flow component as a node. Each node displays these elements.

- The sequence of the flow component in the flow
- The icon representing the spoke or component type
- The name of the flow component
- The menu of available options for the flow component
- The paths available from this node

Note:
The flow diagramming view only displays nodes for supported flow components.

4. Plus icons

Select and configure an action, flow logic, or subflow to insert along a specific path of the flow.

Note:
The flow diagramming view only displays options for supported flow components.

5. Add a node

Select and configure an action, flow logic, or subflow at the end of the flow.

Note:
The flow diagramming view only displays options for supported flow components.

6. Search nodes

Find all nodes that match your search criteria.

7. Diagram view controls

Select the current focus of the view, set the zoom level, download the diagram, and select view options.

- Preview and zoom controls
- Download diagram as an image
- View options

8. Node properties

Configure the properties of the currently selected node.

Preview and zoom controls

Set the current focus of the view by selecting a region of the thumbnail image. Zoom in and out to see specific portions of the flow structure.

Download diagram as an image

Download the current flow as an image in the Portable Network Graphic (PNG) format. Opens a dialog window to choose the location to save the file. The image only shows the nodes, connection lines, and order numbering. If the image is too big to fit into one file, the system creates multiple images for each section of the flow.

View options

Set how nodes appear in the diagramming view.

Details

Show or hide configuration details in a box beneath each node. The configuration details include conditions and data pill paths.

Annotations

Show or hide the annotations available for each node. Annotations appear as italic text beneath each node.

Supported flow components

The flow diagramming view supports a limited selection of flow components. Workflow Studio disables the flow diagramming view when a flow contains unsupported flow components.

Annotations

The flow diagramming view supports adding and editing annotations to actions, flow logic, and subflows.

Data Stream Actions

The flow diagramming view supports adding and editing data stream actions.

Flow logic

The flow diagramming view only displays flows containing these flow logic types.

- Call a Workflow
- Do the following in parallel
- Do the following until
- Dynamic Flows
- Else If
- End Flow
- Exit Loop
- For Each
- Get Flow Outputs
- If
- Make a decision
- Placeholder
- Set Flow Variables
- Skip Iteration
- Try
- Wait for a duration of time

Stages

The flow diagramming view displays the stages available to a flow.

Subflows

The flow diagramming view supports adding and configuring subflows.

Triggers

The flow diagramming view only displays flows with these trigger types.

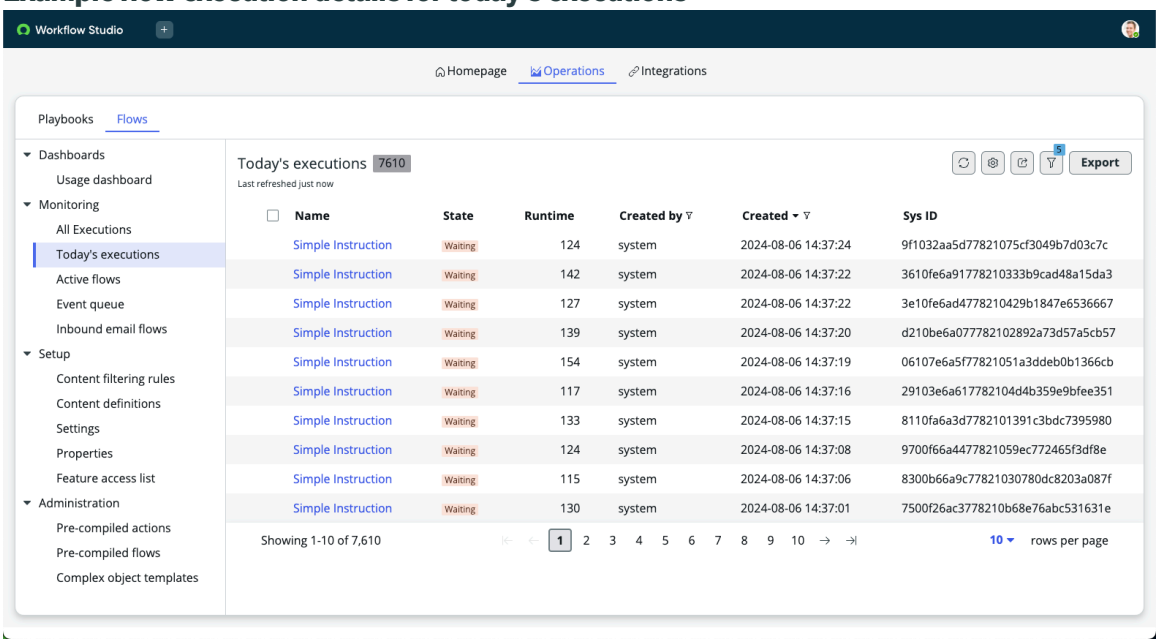
- Record triggers
- Date triggers
- Inbound email
- Kafka Message
- MetricBase
- REST API - Asynchronous
- Service Catalog
- SLA Task

Flow execution details

View run-time information about an action or flow directly from the design environment, such as the current state, items run, and values produced. Open related records from embedded ServiceNow AI Platform editors or in a new tab.

Each time you test a flow, the system generates information about the configuration and runtime values produced as flow execution details. You can view flow execution details from the **Operations** tab of Workflow Studio. Select a flow execution to open its associated execution details page.

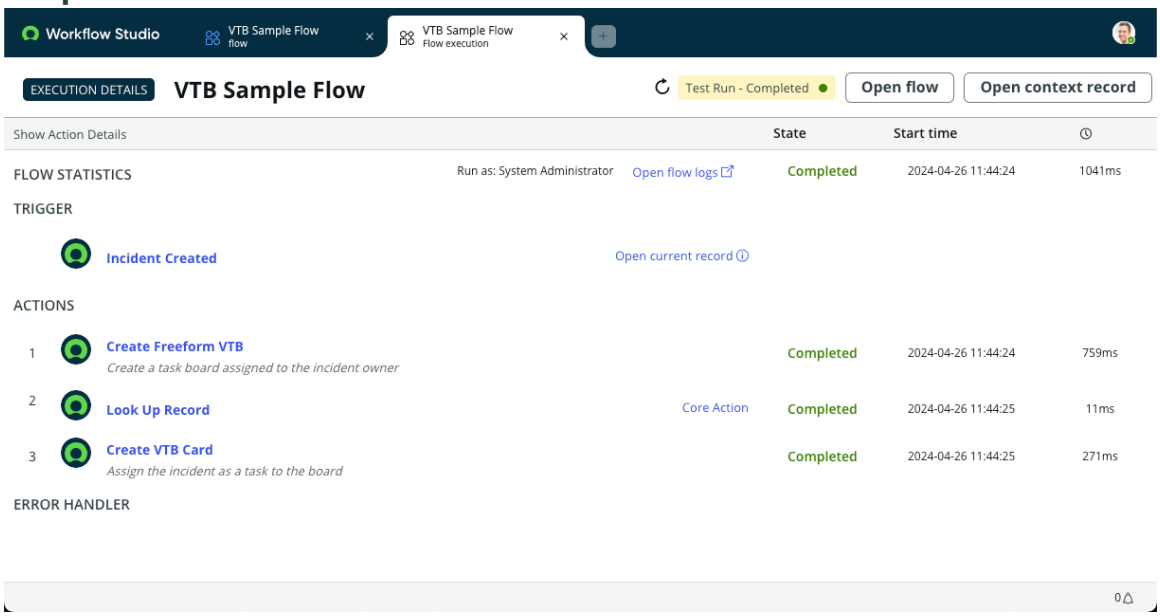
Example flow execution details for today's executions



Each execution details page displays runtime information about the flow.

- Name of the flow
- Refresh flow data
- Flow state
- Related record options
- Flow Statistics

Sample execution details



Reporting level

The reporting level determines what execution details a flow, subflow, or action generates during normal operations. By default, Workflow Studio doesn't generate any execution details during normal operations. Workflow Studio generates full flow execution details when you test an individual flow, subflow, or action. When your instance generates and stores more execution details, reporting has more performance impact on your instance.

Each reporting level generates progressively more flow execution details.

Off

The system doesn't generate flow execution details. The system only generates execution details when you run a test.

> **Note:**
> Testing an action or flow generates execution details at the Trace level.

Basic: Runtime states and durations only

The system generates runtime execution details for each flow, subflow, and action run. You can see the runtime state and duration for these basic items. You can also see configuration and runtime values for flow triggers, subflow inputs, and subflow outputs.

Full: Action configuration and runtime values (for debugging only)

The system generates configuration and runtime execution details for each flow, subflow, and action run. You can see the runtime state, duration, input values, and output values for all items. For custom actions, you can also see the runtime state, duration, input values, and output values of its steps. You can also see the configuration values for flow triggers, subflows, actions, and steps that are part of a custom action.

> **Important:**
> Only users with the fd_read_operations_all role can see configuration and runtime information such as record values in the flow execution details. Users without this role will only see basic details about the state and duration.

Trace: All values (for testing and Support only)

The system generates configuration and runtime execution details for each flow, subflow, action, and step run. You can see the runtime state, duration, input values, and output values for all items. You can also see the configuration values for flow triggers, subflows, actions, and steps.

Important:
Only users with the fd_read_operations_all role can see configuration and runtime information such as record values in the flow execution details. Users without this role will only see basic details about the state and duration. Testing an action or flow generates execution details at the Trace level.

If a flow runs while reporting is off, past execution details are never available for the flow, even if the reporting level later changes. If a flow runs while reporting is on, execution details are available for that flow execution, even if the reporting level changes. The reporting level has no effect on context and log records.

You can configure the default reporting level the system uses to generate execution details each time a flow is run. For more information, see Activate flow reporting.

Refresh flow data

Update flow runtime data as needed. Set a flow preference to refresh flow data automatically when you run a test. See User preferences for flows.

Flow state

All active flows are in one of these states.

Completed

The flow successfully ran all actions. The flow statistics display configuration and runtime details for each action.

Cancelled

The flow was either stopped because the flow's run time duration exceeded the flow time out value, or someone manually cancelled the flow.

Waiting

The flow paused on an action that is waiting for some condition to be met before continuing. The flow statistics display configuration and runtime details for completed actions as well as any actions waiting for a condition to be met. Flows in the Waiting state display a **Cancel Flow** UI action in the header.

Paused

The flow paused to enable higher priority flows to run or to enable a graceful node shutdown. The flow statistics display configuration and runtime details for completed actions.

Presumed Interrupted

The flow has been running for more than 15 minutes and no longer has a valid transaction ID for the current node. Alternatively, the flow has been running for more than eight hours on another node. The flow may have stopped because of an unexpected interruption such as its transaction being terminated or its node being restarted.

Error

The flow stopped with an error. The flow statistics display configuration and runtime details for completed actions and configuration details for the action that produced the error. Flows in the Error state display a **Go to error** UI action in the header.

Note:
When an action results in an error, the flow stops executing at that point and results in an error state.

Related record options

From the Execution details page, you can access records related to the current flow.

Open Flow

Use this option to make configuration changes and publish a new instance of the flow. Changing the flow configuration doesn't change any currently active flow.

Open Context Record

Use this option to view the flow state, run duration, and related log entries from a standard form view. This option opens the context record in a new tab.

Open Flow Logs

Use this link to view detailed log information about each action. This link opens the log entries list in a new tab.

Open Current Record

For flows that have a record-based trigger, use this link to view the triggering record in a pop-up window.

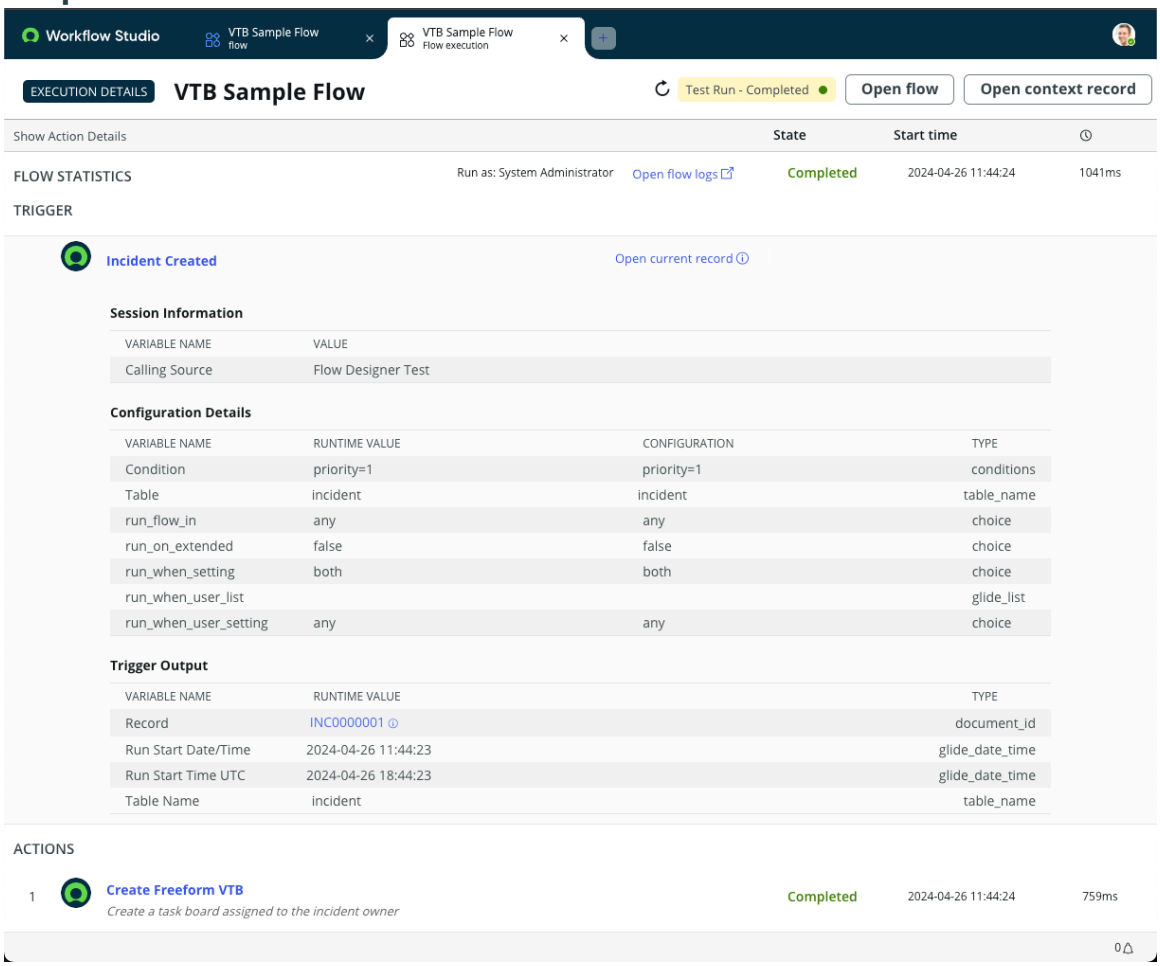
Open Action

Use this link to make configuration changes and publish a new instance of the action. Changing the action configuration doesn't change any currently active flow. This link is unavailable for the core actions provided by ServiceNow.

Flow statistics

Use flow statistics to see configuration details and runtime values for each flow component. Selecting a trigger or action expands the row and displays configuration and runtime details about it.

Sample flow statistics



The following types of execution details are available.

Calling source

View the calling source that started a flow, subflow, or action.

Run as

Identify whether the flow was **Run as** the system or the user who triggered the flow.

Run with roles

Identify the roles granted to the user who triggered the flow.

Integration Metadata

View transaction data such as connection and credential used, MID Server used, target host, and payload size. Integration Metadata is only displayed for integration steps and requires a separate Integration Hub subscription. For more information, see Integration steps.

Configuration Details

View the list of input variables to identify any configuration errors with the action. Each variable has its own row displaying its name, data type, configuration settings, and runtime value. The configuration settings display dynamic values as pills. The runtime values display generated records as a link.

> **Note:**
> Variables that have transform functions only display one runtime value, which is the result of all transformations.

Output Data

View the list of output variables to identify any configuration errors with the action.

Logs

Use the log entries to identify potential processing or performance issues. Each log entry has its own row displaying the creation date, log level, and log message. If the action doesn't generate any logs, the statistics displays the string `No Logs`.

> **Note:**
> Logs display time in UTC format because logs must be saved as strings so that the instance can share the log data between its multiple nodes. Because each node can reside in a different time zone, the UTC format is used as a common format to preserve correct time values.

Steps

Use the list of steps to identify any configuration errors with the action. Each step has its own row displaying the variable name, data type, configuration settings, and runtime value. Core actions don't display steps because you can't change their configuration.

Change the *`com.snc.process_flow.reporting.serialized.val_size_limit`* system property to truncate runtime values in the flow execution details step configuration. To learn more, see Workflow Studio flow system properties.

Start time

View the local time when an action started.

Run duration

Use the run duration to identify potential processing or performance issues. The run duration is measured in milliseconds. As of the Washington DC release, the run duration lists the total time to execute an item. The total time includes these items.

- Time in the event queue
- Time in the ECC queue
- Time processing the event
- Time running in the flow engine
- Time communicating with a MID Server

Retry Info

Use the retry info section to view details about the retry policy. Details include the type of retry strategy, elapsed time, and the next scheduled retry request. The Retry Info section appears only when the retry policy is enabled in the step. For more information, see Retry policy.

Calling source

The calling source lists what started a flow, subflow, or action.

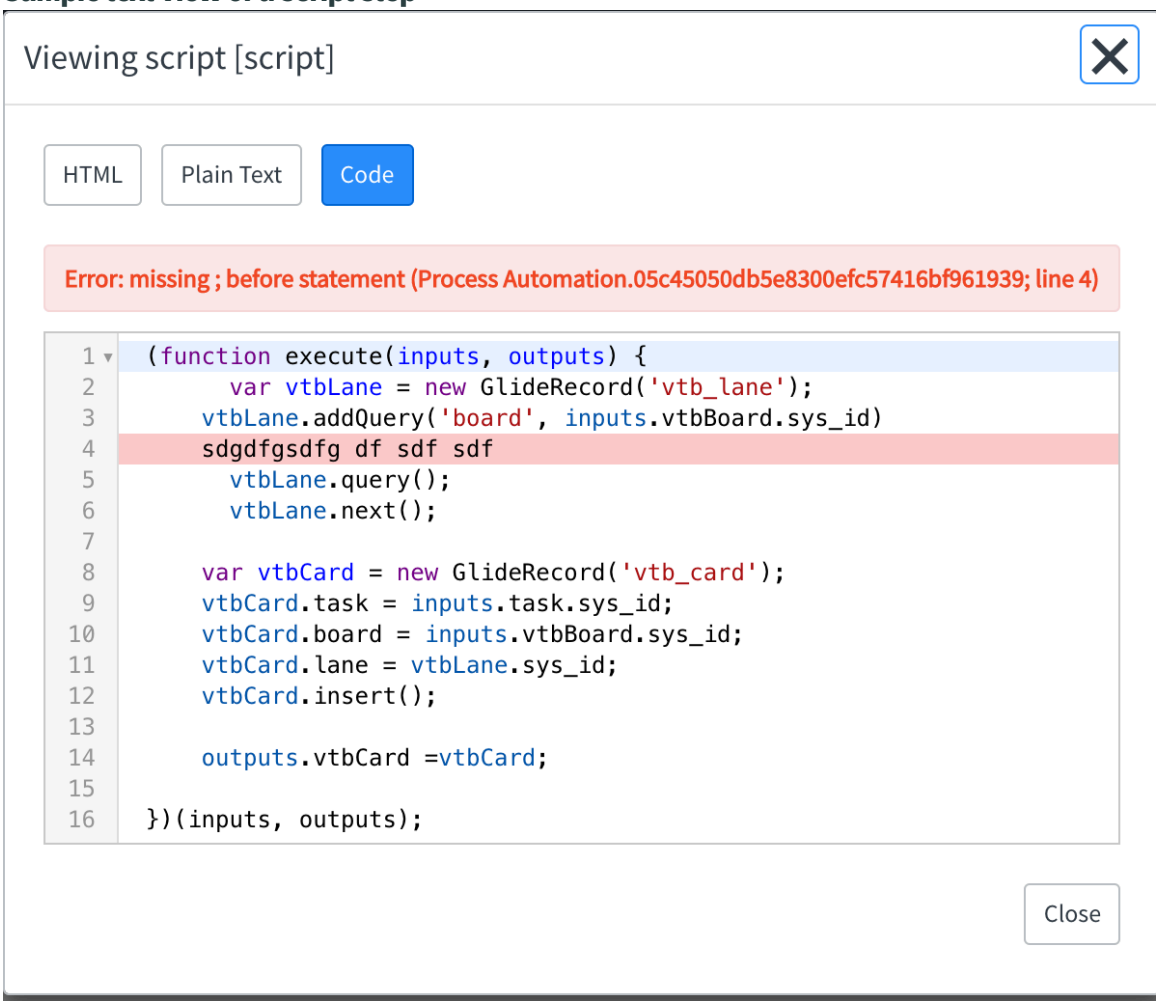
Source	Description
Workflow Studio Test	The flow started because someone selected the **Test** option from the Workflow Studio interface. The flow trigger conditions were ignored.

Source	Description
CRUD Trigger	The flow started when the record-based trigger conditions were met.
Date Trigger	The flow started when the schedule-based trigger conditions were met.
Metric Trigger	The flow started when the MetricBase trigger conditions of a MetricBase were met.
Service Catalog Trigger	The flow started when a Service Catalog item was requested.
Script	The flow started from a method call in a script, such as a business rule.
Background Script	The flow started from a method call in the **Scripts - Background** module.

Embedded text viewer

Workflow Studio displays large text-based configuration and runtime output records, such as email output, XML payloads, or script steps using an embedded text viewer. The embedded text viewer can format text as HTML, plain text, or color-coded JavaScript. For script steps, the text viewer highlights code lines containing errors.

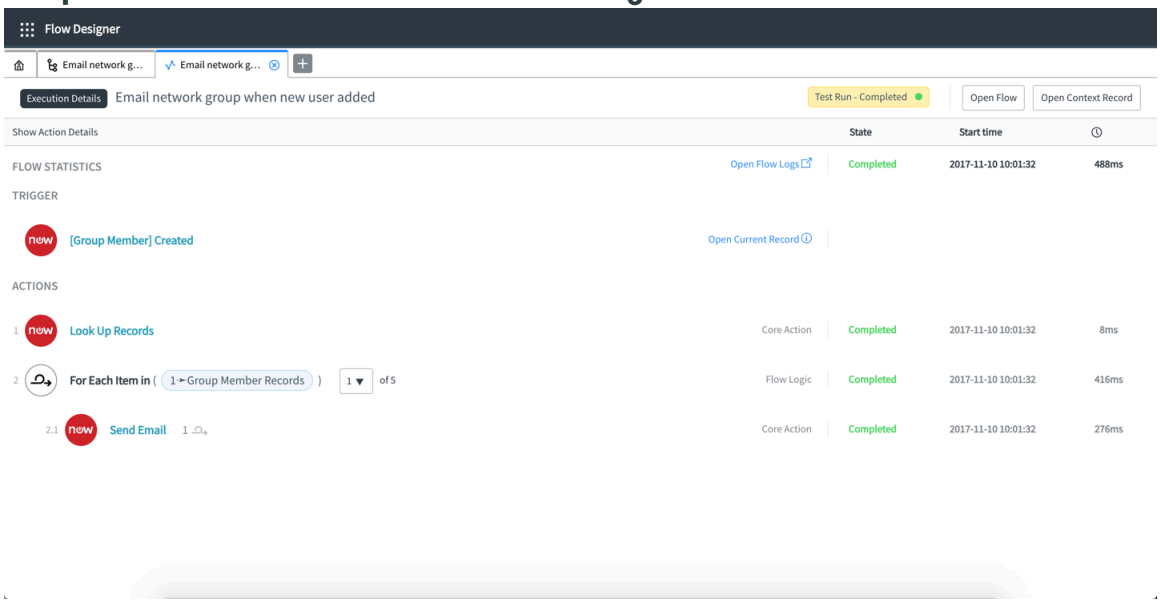
Sample text view of a script step



Viewing results for each item in flow logic

Workflow Studio displays a selector control to view the configuration and runtime results for each item processed by flow logic. Select a record number to see its configuration and runtime details.

Sample flow statistics for each item in the flow logic



Subflow execution details

Process analysts can view subflow execution details from multiple locations.

Parent flow

A parent flow lists the flow execution details of each subflow that it calls as inline elements. Expand the subflow step to see the subflow execution details.

Subflow

The system generates flow execution details for each subflow run. View subflow execution details directly from the list of flow executions.

User role support

You can control access to flow execution details by granting user roles. For more information about available Workflow Studio user roles, see User access to Workflow Studio flows.

Workflow Studio roles for execution details

Role title [name]	Description	Contains Roles
flow_operator	Enables you to view flow execution details, dashboards, and logs. Administrators can grant this role to users that want to be able to view execution results but not create, change, or test them.	none
fd_read_operations	Enables you to view basic flow and action execution details. When reporting is enabled, users with this role can only see basic execution details such as the runtime state and duration. If the reporting level generates additional details, users with this role cannot see them.	none

Workflow Studio roles for execution details (continued)

Role title [name]	Description	Contains Roles
	Administrators can grant this role to users that only need to view basic execution results but not create, change, or test flows and actions. **Note:** Read only roles are incompatible with roles that provide write access. Avoid granting the same user both a read only and a write access role.	
fd_read_operations_all	Enables you to view all generated flow and action execution details. When reporting is enabled, users with this role can view all available execution details. The user can only see as much detail as defined by the reporting level system property. Administrators can grant this role to users that need to view all flow results but not create, change, or test flows and actions. **Note:** Read only roles are incompatible with roles that provide write access. Avoid granting the same user both a read only and a write access role.	fd_read_operations

Run with roles support

When a flow runs with one or more privileged roles, a user must also have these privileged roles to see the flow execution details. For example, if a flow runs as System, then a user must have the admin role to view its flow execution details. In addition, if a flow runs with a role that can access an encryption context, then a user must also have that role in order to view its flow execution details. For more information about encryption contexts and roles, see Column Level Encryption.

Flow roles

Create flows and subflows that run with specific roles. Assigning roles enables you to create user-initiated flows that run with their own roles rather than the user's roles.

Role selection

A flow runs as either the system user or as the user who initiates the session. You can only assign roles to flows that run as the user who initiates the session. When the flow runs as the system user, it runs with the system role, and individual role selection isn't available. For more information, see Create a flow in Workflow Studio.

You can assign multiple roles to a flow. Selecting new roles replaces the flow's original roles. If roles aren't selected, the flow runs with the roles of the user who initiates the session.

The roles you can select for a flow depend on the roles you have and the application scope of the flow. Assign any roles you that have access to in a particular scope, except high-security roles. You can't assign the following roles to a flow:

- admin
- security_admin
- application-specific admin roles, such as an application admin role for Human Resources.

Modified and copied flows

Other users can modify and copy your flow. To modify a flow, a user must have the same roles as the flow. Users missing any of the roles assigned to the flow, sees the flow as read-only.

When you copy a flow, the assigned roles are removed. The copied flow runs with either the system role or the roles of the user who initiated the session.

Missing roles

Sometimes a flow refers to a role that is not on the instance. The missing role may have been removed or may not exist on the instance. Either situation can occur when moving a flow between instances. When a role is unavailable, the **Run with role(s)** field displays the sys_id of the role instead of its name. While the role is missing, you cannot save changes to the flow. To save flow changes, either remove the role from the flow or add it to the instance.

Flow roles in execution details

You can see the "Run with" roles for a flow by viewing the flow execution details. Use the **Run As** field to determine which user ran the flow. Only flows that ran as the initiating user can have roles assigned. These flows have a **Run with role(s)** field that displays the roles assigned to the flow.

Subflow roles

Flows and subflows each run with their own roles. Subflows don't inherit roles from a parent flow. When flow execution returns to a parent flow from a child flow, any special roles associated with the child flow are removed. The parent continues execution with its own roles.

Access control lists

Assigning a role to a flow doesn't guarantee that the flow can access a record or table. While roles are an important part of access control lists (ACLs), they are just one possible condition. If a flow cannot access the records you expect it to, review the record ACL rules for the table and fields. The ACL rules might require additional criteria to grant access. For more information, see access control list rules.

Domain separation and Workflow Studio

Domain separation is supported in Workflow Studio. Workflow Studio supports domain separation of business logic, which lets each tenant domain have its own flows, actions, and subflows. 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 **Basic** level
- Business logic: Processes can be created or modified per customer by the service provider. The use cases reflect proper use of the application by multiple service provider customers in a single instance.
- The owner of the instance needs to be able to configure MVP business logic and data parameters per tenant as expected for the specific application.

Use case: As an admin, I need the ability to make comments mandatory on close of a record for one tenant, but not for another.

How domain separation works in Workflow Studio

The system domain separates Workflow Studio content according to these rules.

Workflow Studio content inherits the domain of the user who creates them

Flows, actions, and subflows belong to the domain of the user who creates them. For example, when a service provider (SP) administrator in the TOP domain creates a flow, it belongs to the TOP domain.

> **Note:**
> The domain selected from the domain picker overrides the domain the user belongs to. For example, when an SP administrator in the TOP domain selects the ACME domain from the domain picker, any content created belongs to the ACME domain.

Workflow Studio content runs from the domain from which it is triggered or initiated

Flows, actions, and subflows run from the domain of the record or user who initiates them. For example, when a user from the child domain ACME triggers a flow belonging to the parent domain TOP, the flow runs in the context of the child domain ACME.

Domain assignment by trigger type

Trigger type	Domain assignment
API call	Domain of the user making API call
Email trigger	Domain of the email sender
Record trigger	Domain of the triggering record
Scheduled trigger	Domain of the flow
Service Catalog trigger	Domain of the requested item record

Workflow Studio only runs content accessible from the current domain context

The system can only run content to which the current domain context allows access. See Understanding domain separation to understand data separation and the domain hierarchy. For example, a user in the child domain ACME can trigger flows belonging to the parent domain TOP, but cannot trigger flows belonging to a sibling domain such as INITECH.

Workflow Studio runs record operations from the current user domain context. A read operation such as the Lookup Records action returns records based on the currently selected domain and its children. For example, if the currently selected

domain is the TOP domain, you will see records from the TOP domain and all its children such as the ACME and INITECH domains. If the currently selected domain is the ACME domain, you will see records from the ACME domain and its children, but you will not see records from the parent TOP domain.

> **Note:**
> Record operations use the data or process separation rules applied to the table the record belongs to. For example, suppose you have process-separated the Business Rule table. If you add a business rule to the TOP domain, the business rule will be accessible to record operations in child domains such as the ACME domain because process separation allows access to records from parent domains.

Flows that call another application such as a decision table or workflow also run from the current user domain context.

Workflow Studio runs all flows whose trigger conditions are met

A flow in one domain cannot override or prevent a flow from another domain from running. Workflow Studio runs any flow that is visible to the current user and whose trigger conditions have been met. For example, a flow belonging to the TOP domain that is triggered by the creation of an incident record runs anytime an incident is created, regardless of whether the incident is created in the ACME or INITECH child domains.

General guidelines

Follow these general guidelines when using domain separation with Workflow Studio.

Ensure that tenant flows, actions, and subflows are run properly for domains

Since tenants cannot override Workflow Studio content, a service provider (SP) administrator from the TOP domain must author and manage them to ensure they run properly for domains. While you can create domain-specific flows, users working from domains higher in the hierarchy may trigger multiple child domain flows. For example, a user working in the TOP domain can trigger flows in child domains such as ACME and INITECH.

> **Note:**
> Flow authors can see only Workflow Studio content available from their current domain and any parent domains in the hierarchy. Workflow Studio does not display content visible from Contains domains.

Provide a unique name for each flow, action, and subflow

Since all domains share Workflow Studio content, have an SP administrator in the TOP domain uniquely name each flow, action, and subflow to ensure that a flow intended for one domain does not duplicate the name of a flow in another domain. For example, add the domain to the flow name such as `Validate incidents - TOP`, `Validate incidents - ACME`, and `Validate incidents - INITECH`.

Ensure that flows and actions only contain artifacts from current or parent domains

Workflow Studio prevents the activation of any flow containing artifacts unavailable to the current or parent domains. For example, if you create a domain-specific flow that belongs to the ACME domain, it cannot contain actions or subflows belonging to the sibling domain INITECH.

Edit Workflow Studio content in the domain to which it belongs

Users in a parent domain can't see flows, actions, and subflows in a child domain. They must change to the domain to which they belong to edit them. For example,

an administrator in the TOP domain can't see flows from the ACME domain. The administrator must switch to the ACME domain to see and edit them.

Related topics

Domain separation for service providers

Generate group approvals for domain separated requests

Configure Workflow Studio to generate approvals for all members of a group or to restrict approvals to only group members who are visible from the domain of the request.

Before you begin

Role required: admin

About this task

By default, Workflow Studio generates approvals for all group members who can access the parent request regardless of domain visibility. This configuration allows requests from members of a child domain to generate approvals for members of a parent domain who are not otherwise visible from lower in the domain hierarchy. You can use this procedure to restrict the generation of approvals to only those group members who are visible from the domain of the parent request. For information about domain hierarchies, see Understanding domain separation. For more information about visibility and contains domains, see Visibility domains and contains domains.

Procedure

1. Add a system property.
2. For the system property name, enter `com.glide.hub.flow.approval.group_member.use_query_no_domain`.
3. For the system property type, select **true|false**.
4. For the system property value, enter one of these values.

Option	Description
true	Generate approvals for all members of the group who have access to the domain of the parent request. Select this option to exclude domain visibility from the approval generation query. For example, generate approvals for users who belong to domains higher in the do main hierarchy. This is the default value.
false	Generate approvals only for group members who are visible from the domain of the parent request. Select this option to include domain visibility in the approval generation query. For example, do not generate approvals for users who belong to domains higher in the domain hierarchy.

Note:
Workflow Studio only generates approvals for users who can access the domain of the request.

Result

Workflow Studio only generates approvals for group members who belong to the same domain or child domains of the parent request.

Architecture Overview

Understand how Workflow Studio works within the ServiceNow AI Platform to activate, trigger, and process flows and actions.

A flow consists of a trigger and one or more actions. The trigger specifies when to start the flow, which can be record-based, schedule-based, or application-based. Record-based triggers run a flow after a record has been created, updated, or deleted. The flow can use the triggering record as input for actions. Schedule-based triggers run a flow at the specified date and time. The flow can use the execution time as input for actions. Application triggers are added when the associated application is activated. For example, the MetricBase trigger is present when the MetricBase application is active.

Flow processing

Flow processing occurs in this sequence.

1. When the flow trigger conditions occur or an API directly calls the flow, the system creates an entry in the event queue to start the flow.
2. The scheduler processes the event and starts the flow in the background.
3. The system builds a process plan from the flow.
4. The system runs the process plan using the record that triggered the flow.
5. The system stores the execution details in a context record.

1. Process flow triggers and API calls

Each time trigger conditions are met or an API directly calls a flow, Workflow Studio creates an event entry. The system processes triggers after database operations. To learn more, see Execution order of scripts and engines. Typically, How business rules work and Workflow engine operation order that run synchronously run before a triggered flow.

2. Process events in the queue

Each flow event contains a reference to the flow to start and a reference to either the triggering record or the execution time. The system processes these events using Events where a scheduler periodically works through the current items in the event queue in the order in which they were added. Depending on what other events are in the queue, the system may not immediately start a flow. Flow designers should expect some lag time between when the trigger conditions occur and when the flow actually starts.

3. Build the process plan

When Workflow Studio pulls an event from the queue, it builds a process plan to actually run the flow. A process plan contains all the information necessary to execute a flow such as the sequence of published actions or subflows, the input values for each subflow or action, the action steps to run for each action, and the data provided by the trigger or subflow output.

Workflow Studio uses a just-in-time compilation scheme to ensure that process plans contain the latest changes to flows, subflows, and actions. If no changes are

detected, Workflow Studio uses a cached copy of the process plan. Otherwise, it builds a new process plan.

By automatically checking for updated flows, subflows, and actions with process plans, Workflow Studio enables you to apply changes from update sets and upgrades without having to edit current flows. If you move published actions to a target instance, every flow that uses the published action will automatically update the next time it is executed.

Warning:
If changing subflows or actions used in activated flows, do not change the inputs and outputs used in the subflow or action. Changing inputs and outputs may cause errors when the activated flow is next triggered because it has not been configured to use the new inputs and outputs. Any currently running flows are unaffected by changes to inputs or outputs as the flow uses the compiled subflows and actions from the process plan.

4. Run the process plan

Workflow Studio runs the process plan as the user specified in the flow properties and runs it within the flow application scope.

When running a flow with a record-based trigger, Workflow Studio stores the triggering record in memory as an instance that is represented in the interface as a data pill.

Each time you add an action to a flow, Flow Designer adds a data pill to store its results. The data pill name indicates its sequence in the flow and its data type. Flow designers use action result data pills to provide input for other flows, actions, or subflows. Flow designers can use the sequence value in the data pill name to ensure that they select the correct data pill as an input value. When a flow runs an action, it generates the data pill runtime value as it is used.

5. Store flow execution details

Workflow Studio stores flow execution details in a flow context record, which contains this information.

- Flow outcome state
- Flow runtime duration
- Flow log messages
- Flow configuration and runtime values

Each time a flow runs, Workflow Studio adds an entry to the **Flow Executions** list. Each entry has its own context record and matching execution details page.

Note:
A flow execution context runs in a single thread. However, there may be times when you want to run flows within separate contexts even though this may consume more of your instance's resources. To run subflows in separate flow contexts within the same flow, see Dynamic flows.

A flow can have one of these outcome states.

State	Description
Complete	The flow completed successfully.
In Progress	The flow is running. By default, a transaction quota rule prevents flows from running longer than an hour.
Waiting	The flow is waiting for another event to occur. For example, a user must update a task or approval, or a record must reach a specific state. When in the waiting state, the flow is quiesced and serialized into a context record.
Canceled	The flow was canceled by a user.
Error	The flow encountered an error and has stopped running. For example, an action is missing an input value, or a quota transaction rule has stopped the flow.

Flow, subflow, and action life cycle

Workflow Studio uses the flow or action status to describe the current state of configuration changes.

Flow and subflow status and activation state

The **Status** field indicates whether there is a process plan associated with the flow or subflow.

Flow status	Description
Modified	Indicates that there are unsaved changes to a flow or subflow. Modified flows or subflows have not been saved.
Draft	Indicates that there are saved changes to a flow or subflow, which have not been stored in a process plan. Draft flows have been saved but not activated. Draft subflows have been saved but not published.
Published	Indicates that there is a stored process plan for the flow or subflow. Published flows have either been activated or deactivated.

The **Active** field indicates whether the system runs a flow or subflow.

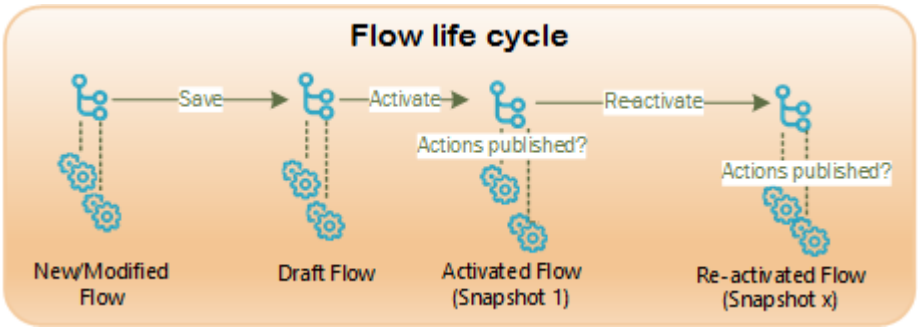
Active	Description
True	Indicates that the flow or subflow is active and runs when triggered or called. The flow has been activated or the subflow has been published. Active flows run when the trigger conditions are met or when called. Active subflows run when called.
False	Indicates that the flow is inactive and does not run when triggered or called. An inactive flow has either never been activated or has been deactivated. An inactive subflow has never been published.

When working with flows, you can:

- **Save** a flow: Creates a draft of the flow.
- **Activate** a flow: Enables the flow trigger and transform the flow into a process plan.
- **Deactivate** a flow: Disables the flow trigger and prevents new flow executions. Currently running flows continue to run.

When working with subflows, you can:

- **Save** a subflow: Creates a draft of the subflow. If the subflow is modified after being published, the subflow moves into a draft state. Any active flows that use the subflow only run the published subflow.
- **Publish** a subflow: Enables you to activate a flow containing the subflow. Publishing adds the subflow to the list of available subflows in a flow.



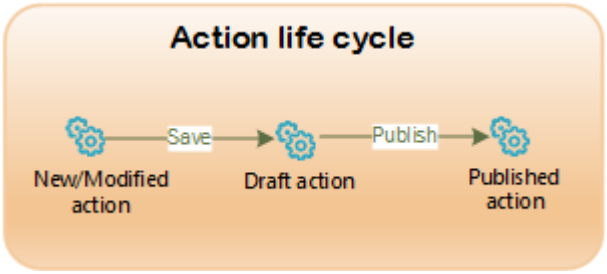
Action status

The Workflow Studio interface does not display the configuration status of actions. To view action status, navigate to the Action Types table [sys_hub_action_type_definition] and display the **Draft state** field.

Action Draft status	Description
Draft	Indicates that there are changes to an action that have not been published. Draft actions are only available to flows when the **Show draft actions** option is enabled. You cannot activate a flow containing draft actions.
Published	Indicates that the action has been published. Published actions are available to all flows and allow flows to be activated.

When working with actions, you can:

- **Save** an action: Creates a draft of the action that is only available to flows when **Show draft actions** is enabled. If the action is modified after being published, the action moves into a draft state. Any active flows that use the action only run the published action.
- **Publish** an action: Enables you to activate a flow containing the action. Publishing adds the action to the list of available actions in a flow. Only actions in a published state run during flow execution.



Application development

When designing an action or a flow, use these general guidelines.

Use standard ServiceNow AI Platform application development capabilities to create, manage, protect, and deploy Workflow Studio content. Flow and action designers typically perform the following application development tasks:

- Create a custom application to store flows and actions.
- Set application permissions to share or restrict access to application data.
- Grant application developers access to Workflow Studio.
- Publish custom applications to the application repository to deploy flows and actions on other instances.

Collision avoidance

Workflow Studio supports collision avoidance. Collision avoidance prevents a user from modifying an object that is being modified in a different update set. For example, User A is editing a flow in a particular update set. User B, who is working in a different update set, attempts to open the same flow. In this situation, the system detects a collision and alerts User B. User B can then choose to either **Cancel** or **Continue**. Selecting **Cancel** takes User B back to the Workflow Studio homepage. Selecting **Continue** opens the flow in read-only mode.

For collision avoidance to work, both users must be in the same application scope, and it must be an application scope other than global. Additionally, the application being modified must be linked to source control. For more information, see Collision avoidance.

Security

Control access to Workflow Studio processes and records.

- Administrators can grant users access to Workflow Studio flows by creating an application and assigning users as developers with the Workflow Studio delegated development permission. Delegated development allows administrators to control whether flow designers can access features normally restricted to admin users such as assigning user roles, creating access controls, or creating scripts. For more information, see Developer permissions.
- Administrators can grant access to Workflow Studio flows by directly assigning users the flow_designer user role, which includes the role to view flow execution details.

> **Warning:**
> Directly granting a user the flow_designer role is equivalent to giving the user the admin role, because Workflow Studio can run flows as the System user, which has access to all tables and all database operations.

- Flow and action designers can use standard Application access settings to manage how their content interacts with other applications.

Action limit

By default, flows can have no more than 50 actions. To change the default behavior, increase the value of the *sn_flow_designer.max_actions* system property. However, consider the performance impact that a large flow may have on your instance.

Trigger options for record updates

Flow designers can specify how often a flow can update a particular record with the **Run Trigger** option. Use the **Once** option when you want a flow to run only once. The first time a record is updated, the flow runs, but any further record updates do not trigger the flow. Use the **Always** option when you want the flow to run every time a record is updated and there is not already an active flow running for it. For example, you might set a flow that assigns an incident record to run only once, and set a flow that notifies the incident watch list to always run. The **Run Trigger** field is only available for these trigger types.

- Created or Updated
- Updated

Direct recursion prevention and indirect recursion limit

To prevent instance outages and consumption of system resources, Workflow Studio ignores any request to start a flow or subflow that is the result of direct recursion. Direct recursion occurs under these conditions.

- An action calls the same flow that it is part of. For example, a script step makes an API call to a flow.
- An action or subflow produces a result matching the flow trigger. For example, a flow that runs when an incident record is updated contains an update record action that updates an incident record.

Workflow Studio also limits the number of times a flow can be started from indirect recursion. Indirect recursion occurs under these conditions.

- The same flow is called multiple times in a chain of subflow calls. For example, if subflow A calls subflow B, and subflow B calls subflow A, then calling either subflow produces indirect recursion.
- The same flow is triggered multiple times in a chain of subflows. For example, suppose that there are two flows triggered by record creation. Suppose that creating record A triggers flow A and also creates record B. Furthermore, creating record B triggers flow B and creates record A. Creating either record type produces indirect recursion.

By default, the system stops triggering flow runs after the run count reaches the indirect recursion limit of three runs. Administrators can change the limit by setting the system property *com.glide.hub.flow_engine.indirect_recursion_limit* to an integer value equal to or greater than one. The system ignores any property value less than one and instead uses a limit of one. Consider the performance impact that increasing the indirect recursion limit may have on your instance.

Note:
By default, a transaction quota rule prevents flows from running longer than an hour.

Flow and action testing

Testing a flow bypasses the trigger conditions and immediately runs it. Testing a flow with a record-based trigger requires selecting a specific record to act as the trigger. Flow designers

should generate appropriate sample records prior to testing. For more information about testing a flow, see Test a flow.

During the design phase, you can test unpublished actions by setting **Show draft actions** on the flow. If testing with draft actions, use these guidelines.

- Design flows and actions on a non-production instance. Only deploy active, working flows to your production instance.
- Leave **Show draft actions** set to true until your draft action is in a final state. Once final, publish each action, set **Show draft actions** to false, and activate the flow.

> **Warning:**
> Disabling **Show draft actions** before publishing your actions removes all draft actions from your flow.

- Any change you make to an active flow or published action causes it to return to a draft state. If the flow is triggered, the system only runs the activated flow and published actions, and the flow execution details only display what was run. When there is a draft of an active flow, the trigger and actions listed in the flow execution details may be different than those listed in the draft flow.

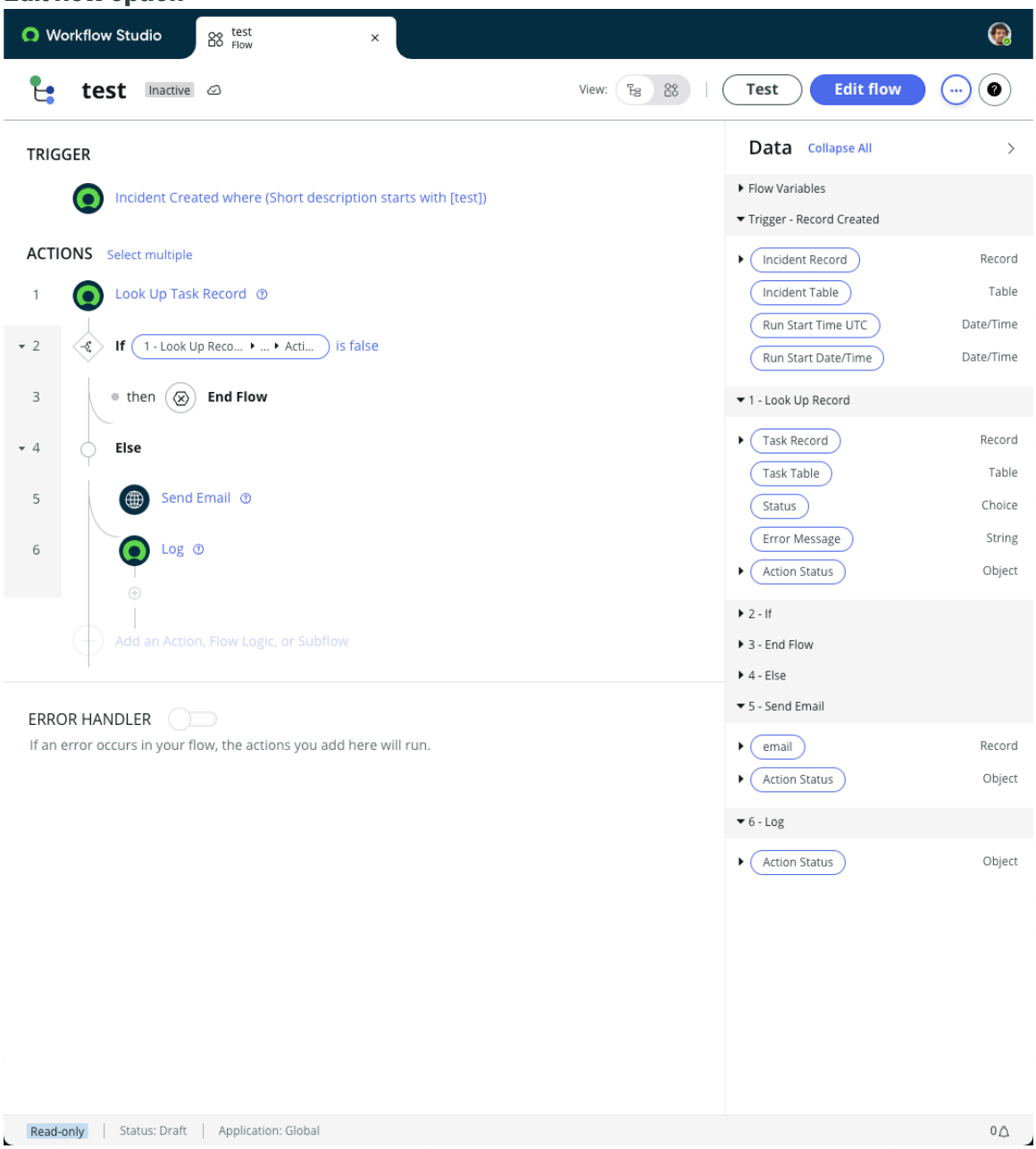
Default read-only flows

Open existing flows in a read-only state to protect them from accidental changes. While a flow is in a read-only state, you can only review, test, deactivate, or request to edit it.

Request to edit flow

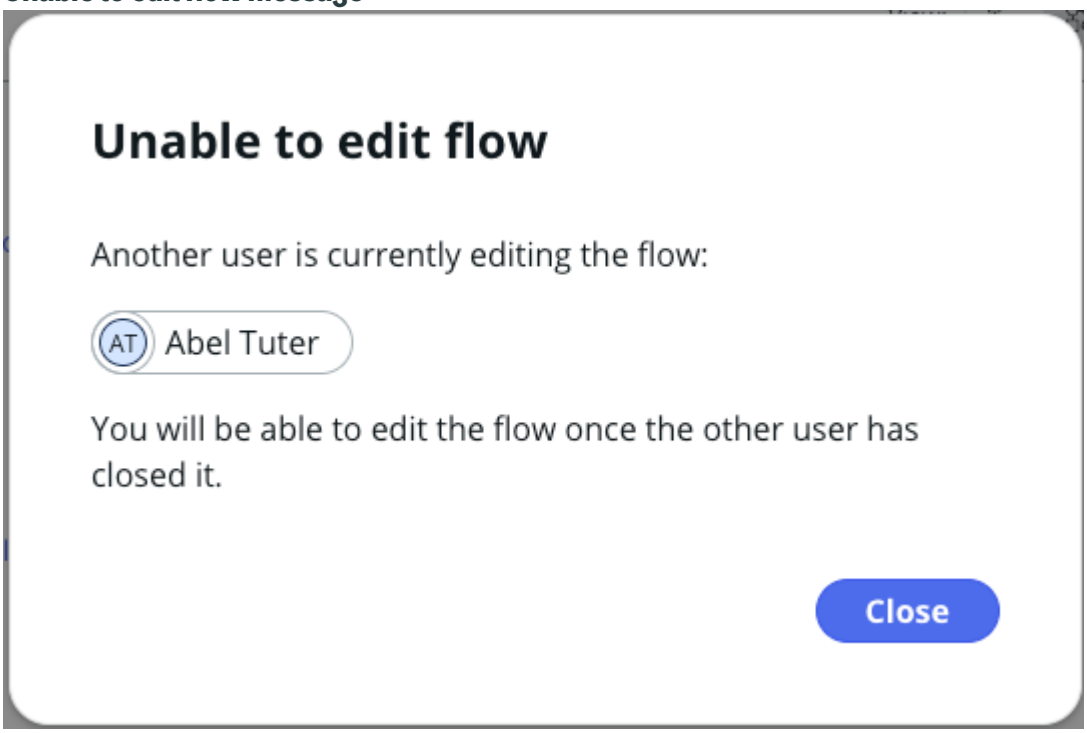
As of the Washington DC release, you must request to edit an existing flow before you can change it. If you don't have permission to edit a flow, the Edit flow button remains disabled.

Edit flow option



When you request to edit a flow, Workflow Studio checks whether another person already has the flow open for edit. If the flow is available for editing, you open the flow in an editable state. If the flow is already being edited, you see a message displaying the name of the user editing the flow.

Unable to edit flow message



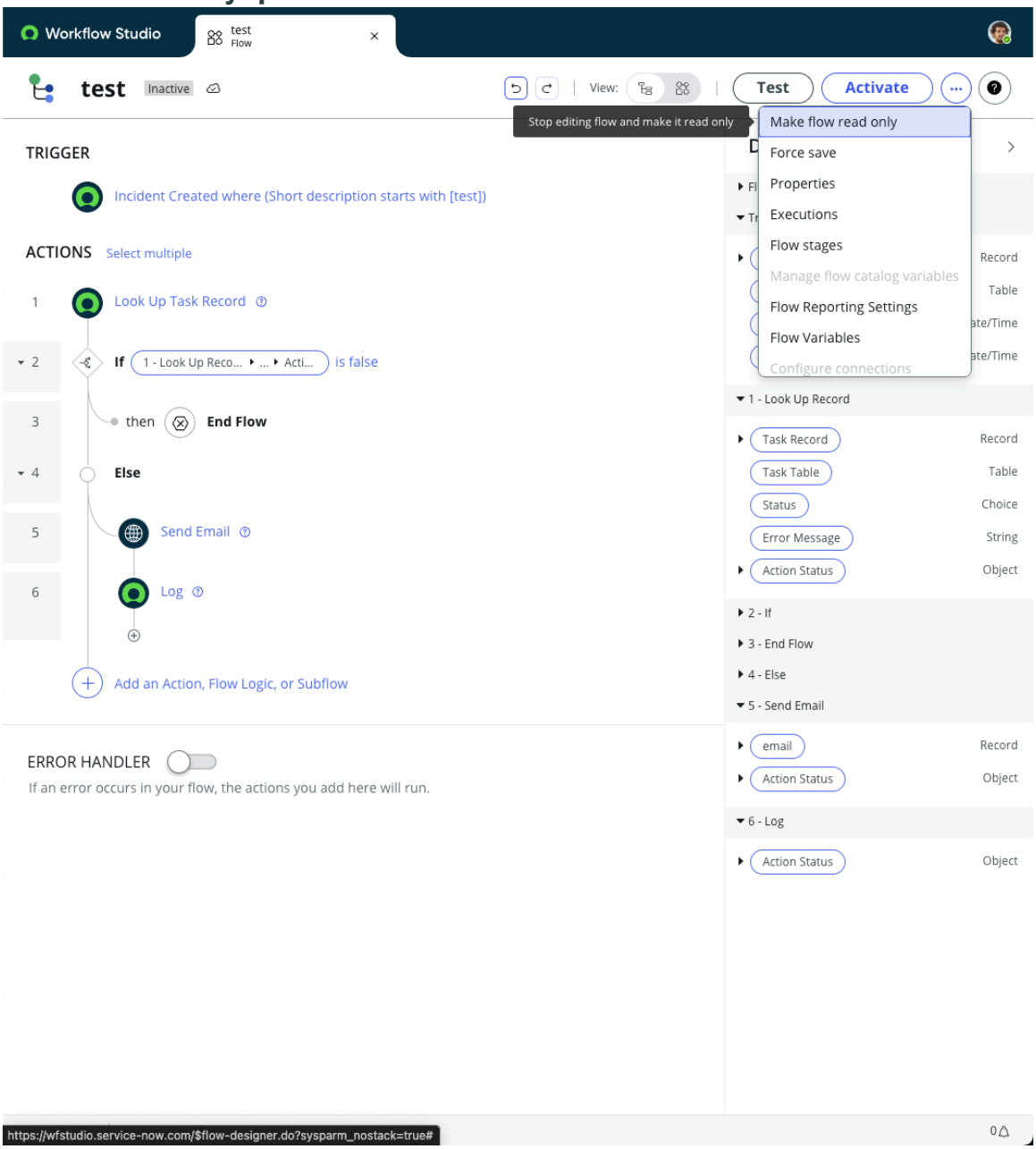
Stop editing a flow

Flows remain in an editable state until one of these events occur.

- You close the Workflow Studio flow tab.
- You close the Workflow Studio browser tab.
- You select the **Make flow read only** option.
- You remain inactive in the flow for more than 30 minutes.

When you're done editing a flow, you can make it read only so that other people can edit it. You can use the **Make flow read only** option to stop editing a flow.

Make flow read-only option



Exploring subflows

Subflows automate a repeatable multi-step process that also produces an output needed by another process. When a playbook, flow, or script calls a subflow, the subflow runs a sequence of reusable actions and flow logic to complete the process and produce output values.

Subflows overview

A subflow is an automated process consisting of a sequence of reusable actions and flow logic, data inputs, and outputs. In contrast to flows, subflows do not have a trigger but instead run when called from a playbook, flow, another subflow, or a script. The inputs describe the data used to run the subflow. The actions perform a sequence of operations on your data. For example, the **Change - Implementation tasks** subflow creates an implementation and a post implementation task given a input change record.

Building and managing subflows requires that you have some familiarity with the ServiceNow AI Platform tables and fields that the application or process uses. Process analysts can create subflows using available actions or use an existing subflow as a template. See Building subflows.

Subflows consists of the following components.

Subflow Inputs

A subflow input stores data used to run the subflow. Each input has a name and a data type. You can define one or more inputs that are available to the subflow. When calling a subflow, flow authors must provide data values for the subflow inputs.

Subflow outputs

A subflow output stores data generated by a subflow. Each output has a name and a data type. You can define one or more outputs that are available to the subflow. When calling a subflow, flow authors can use subflow outputs as data for operations later in the flow. You can use the Assign subflow outputs flow logic to set output values.

Subflow execution details

A subflow execution details page allows a flow author to view run-time information about an action or flow directly from the design environment. You can view details such as the current sate, actions or steps run, output values generated, and errors produced. See Flow execution details.

Flow error handler

A flow error handler enables a subflow to catch and report errors from the subflow execution details. Run a sequence of actions and subflows to identify and correct issues. For example, have subflows log output values, send notifications, and run corrective subflows when they produce an error. See Flow error handler.

Actions

An action is a reusable operation that enables process analysts to automate ServiceNow AI Platform features without having to write code. For example, the **Create Record** action allows process analysts to generate records in a particular table with particular values when certain conditions occur. ServiceNow core actions like Create Record require some familiarity with ServiceNow AI Platform tables and fields. Action designers can create application-specific actions to pre-set configuration details. For example, creating a Create Incident Task action ensures that the process analyst uses the correct table and field configuration each time the action is used. You can add application-specific actions by activating the associated spoke. See Workflow Studio actions.

Workflow to create subflows using Workflow Studio

The following illustration describes the basic tasks involved in creating a subflow using Workflow Studio. For detailed instructions for creating a subflow, see Create a subflow in Workflow Studio.

Subflows benefits

Subflows provide process owners and developers these benefits:

- Automates repetitive work to improve efficiency and experience.
- Describes a workflow in natural language to help non-technical users understand what it does.
- Displays a workflow as a diagram to help builders see available paths and connections.
- Enables creating and testing a workflow from a single interface to ensure it works as expected.
- Promotes process automation by enabling subject matter experts to develop and share reusable actions with flow authors.

- Reduces upgrade costs, with upgrade-safe ServiceNow AI Platform logic replacing complex custom script.
- Reduces development costs by providing a library of reusable actions.
- Scales with separate subscriptions for integration and Robotic Process Automation (RPA) functionality.

Benefit	Feature	Users
Build an automated workflow from an existing library of automated operations.	Subflow	Application developer, process owner, or administrator
Run an automated workflow on demand.	Subflow	Application developer, process owner, or administrator
Specify the input data that an automated workflow uses to run.	Subflow	Application developer, process owner, or administrator
Store one or more outputs as data for use by other automated workflows.	Subflow	Application developer, process owner, or administrator
Build an automated operation from an existing library of automated steps.	Action	Application developer, integration owner, or process owner
Run an automated operation on demand.	Action	Application developer, integration owner, or process owner
Use one or more inputs as data to run an automated operation.	Action	Application developer, integration owner, or process owner
Store one or more outputs as data for use by other automated operations.	Action	Application developer, integration owner, or process owner

Conversational subflows

Run a Workflow Studio subflow from a Now Assist conversation. Create and configure the conversational skill from Workflow Studio.

Workflow Studio offers a selection of preconfigured subflows that are available to run from Conversational Interfaces.

Conversational requirements

Currently, you can't make a subflow conversational. You can only modify the conversational settings of preconfigured conversational subflows. When editing a preconfigured conversational action, you must follow these guidelines to keep the action conversational.

- All subflow inputs are compatible with Conversational Interfaces.
- All subflow inputs have tooltip text.
- The subflow is active and published.

- You have an appropriate role to access conversational subflows.
- You activate the subflows and actions skill.

Activating the subflows and actions skill

To activate the subflows and actions skill, see Turn on the subflows and actions skill.

User role access

Personnel with the following user roles can access conversational subflows.

Role	Description	Contains roles	Groups with this role	Special considerations
Virtual Agent administrator [virtual_agent_admin]	Provides read and write access to all Virtual Agent tables, conversational experiences, and Now Assist admin operations	action_designer, flow_designer	None	Avoid granting an admin role when more specialized roles are available.
Workflow Conversational Admin [sn_conv_fa.conv_fa_admin]	Provides read access to flows, subflows, and actions. Provides read and write access to conversational experience settings.	sn_conv_fa.conv_fa_user, fd_read_actions, fd_read_flows	None	Avoid granting an admin role when more specialized roles are available.
Flow/Action Designer Conversational Workflow [sn_conv_fa.conv_fa_designer]	Provides read and write access to conversational experience settings.	None	None	This is a specialized role with limited access.
Action Designer [action_designer]	Provides read and write access to Workflow Studio actions.	sn_conv_fa.conv_fa_user	None	This is a specialized role with limited access.
Flow Designer [flow_designer]	Provides read and write access to Workflow Studio flows and subflows.	sn_conv_fa.conv_fa_user	None	This is a specialized role with limited access.

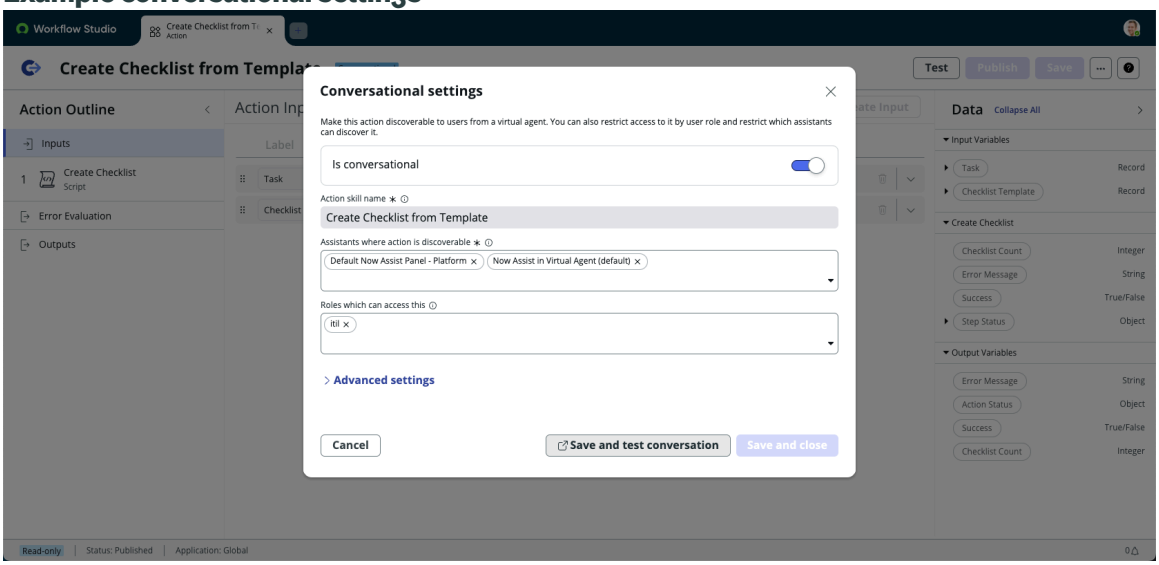
Conversational settings

You can use the conversational settings menu to manage conversational subflows and actions from Workflow Studio. Options include:

- Toggle off or on the option to make an action or subflow conversational.
- See the subflow or action skill name.

- Select one or more assistants that can discover the action or subflow skill.
- Select one or more roles users must have to access the action or subflow skill.
- Set the advanced option to make the action or subflow skill discoverable.
- Set the advanced option to include the action or subflow skill in the list of topics.

Example conversational settings



When you set these options in Workflow Studio, the system also sets the corresponding options in Virtual Agent Designer.

Supported conversational subflows and actions input data types

Conversational subflows and actions support a limited number of input data types. To be compatible with conversational interfaces, an action or a subflow must only include inputs that use supported data types.

ServiceNow AI Platform data type name	Workflow Studio data type label
array.string	Array of Strings
boolean	True/False
calendar	Calendar Date/Time
choice	Choice
date	Date
datetime	Date/Time
document_id	Document ID
date_time	Date/Time
due_date	Due Date
email	Email
glide_date	Date
glide_time	Time
glide_date_time	Date/Time

ServiceNow AI Platform data type name	Workflow Studio data type label
GUID	Sys ID (GUID)
html	HTML
integer	Integer
long	Long
longint	Long Integer String
reference	Reference
schedule_date_time	Schedule Date/Time
string	String
string_full_utf8	String (Full UTF-8)
table_name	Table Name

Available conversational subflows

Workflow Studio provides a set of subflows that are preconfigured to be compatible with and callable by conversational interfaces such as Now Assist.

Name	Application Scope	User roles required
Send Email	Conversational subflows and actions	admin
Send SMS	Conversational subflows and actions	admin
Look up Opportunities - Sample	Microsoft Dynamics CRM Spoke	admin
Look up Contacts - Sample	Microsoft Dynamics CRM Spoke	admin
Look up Accounts - Sample	Microsoft Dynamics CRM Spoke	admin
Look up Leads - Sample	Microsoft Dynamics CRM Spoke	admin
Create Lead - Sample	Microsoft Dynamics CRM Spoke	admin
Add User to Group - Sample	Microsoft Entra ID Spoke (formerly Microsoft Azure Active Directory spoke)	admin
Add User to Group using Email Address - Sample	Microsoft Entra ID Spoke (formerly Microsoft Azure Active Directory spoke)	admin
Look up Direct Reports - Sample	Microsoft Entra ID Spoke (formerly Microsoft Azure Active Directory spoke)	admin

Name	Application Scope	User roles required
Look up Group Members - Sample	Microsoft Entra ID Spoke (formerly Microsoft Azure Active Directory spoke)	admin
Look up Groups - Sample	Microsoft Entra ID Spoke (formerly Microsoft Azure Active Directory spoke)	admin
Look up Users - Sample	Microsoft Entra ID Spoke (formerly Microsoft Azure Active Directory spoke)	admin
Look up Purchase Order - Sample	Coupa Spoke	admin
Look up Images - Sample	Amazon EC2 Spoke	admin
Run Instances - Sample	Amazon EC2 Spoke	admin

Exploring actions

Actions automate a repeatable task or operation within a flow. Flows run actions by passing them data as inputs. Actions run a sequence of steps to complete the task, and pass data to the flow as outputs.

Actions overview

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.

A reusable action includes these components.

Inputs

Inputs are data variables used in your action. For example, if an action step creates a record in the incident table, your input might be a reference to the incident table. Once added as an input, the table and its fields are available to steps and outputs in the flow.

Each input you define for an action becomes a configuration option in the Workflow Studio interface. To use the action in a flow, flow designers must define a value for each mandatory input. The more inputs an action has, the more data flow designers must define and the more familiar they must be with the underlying data model to use the action effectively.

Inputs provide advanced options based on their data type. All inputs have advanced options to add a hint or provide a default value. Use advanced options to guide flow designers through adding and configuring an action to a flow. For example, create a choice input to provide flow designers with a pre-defined list of configuration options to choose from. For more information about the configuration options available to particular data types, see field types.

Outputs

Outputs are data variables that represent the results of the action. These results are available to other actions in a flow.

Steps

A step is a single reusable operation within an action. For example, the **Create Record** step allows action designers to specify the table and field values to use during record creation. Step configuration requires subject matter expertise with application tables, fields, and business logic. Application developers or IT generalists add steps to actions from the Workflow Studio action design environment. Workflow Studio provides a set of ServiceNow core steps to automate ServiceNow AI Platform processes. You can add application-specific steps by activating the associated spoke.

Workflow to create actions using Workflow Studio

The following illustration describes the basic tasks involved in creating a subflow using Workflow Studio. For detailed instructions for creating a subflow, see Create an action in Workflow Studio.

Actions benefits

Using Workflow Studio, you can:

- Create application-specific actions with pre-set configuration details, enabling process analysts to easily add actions to a flow with little configuration.
- Create scripted actions that appear code-less when added to a flow.
- Build integrations using Integration Hub.

Benefit	Feature	Users
Build an automated operation from an existing library of automated steps.	Action	Application developer, integration owner, or process owner
Run an automated operation on demand.	Action	Application developer, integration owner, or process owner
Use data from other automated operations as inputs.	Action	Application developer, integration owner, or process owner
Pass data to other automated operations as outputs.	Action	Application developer, integration owner, or process owner
Stream data in a sequence of pages.	Data Stream Action	Integration owner

Conversational actions

Run a Workflow Studio action from a Now Assist conversation. Create and configure the conversational skill from Workflow Studio.

Workflow Studio offers a selection of preconfigured actions that are available to run from Conversational Interfaces.

Conversational requirements

Currently, you can't make an action conversational. You can only modify the conversational settings of preconfigured conversational actions. When editing a preconfigured conversational action, you must follow these guidelines to keep the action conversational.

- All action inputs are compatible with Conversational Interfaces.
- All action inputs have tooltip text.
- The action is active and published.
- You have an appropriate role to access conversational actions.
- You activate the subflows and actions skill.

Activating the subflows and actions skill

To activate the subflows and actions skill, see Turn on the subflows and actions skill.

User role access

Personnel with the following user roles can access conversational actions.

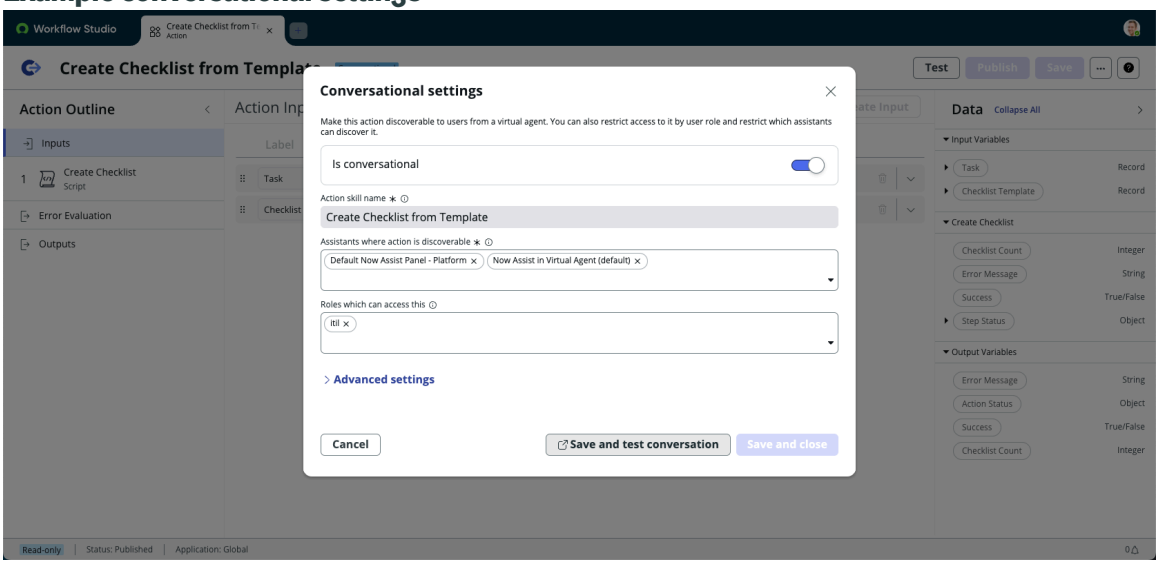
Role	Description	Contains roles	Groups with this role	Special considerations
Virtual Agent administrator [virtual_agent_admin]	Provides read and write access to all Virtual Agent tables, conversational experiences, and Now Assist admin operations	action_designer,flow_designer	None	Avoid granting an admin role when more specialized roles are available.
Workflow Conversational Admin [sn_conv_fa.conv_fa_admin]	Provides read access to flows, subflows, and actions. Provides read and write access to conversational experience settings.	sn_conv_fa.conv_fa_designer, fd_read_actions, fd_read_flows	None	Avoid granting an admin role when more specialized roles are available.
Flow/Action Designer Conversational Workflow [sn_conv_fa.conv_fa_designer]	Provides read and write access to conversational experience settings.	None	None	This is a specialized role with limited access.
Action Designer [action_designer]	Provides read and write access to Workflow Studio actions.	sn_conv_fa.conv_fa_designer	None	This is a specialized role with limited access.
Flow Designer [flow_designer]	Provides read and write access to Workflow Studio flows and subflows.	sn_conv_fa.conv_fa_designer	None	This is a specialized role with limited access.

Conversational settings

You can use the conversational settings menu to manage conversational subflows and actions from Workflow Studio. Options include:

- Toggle off or on the option to make an action or subflow conversational.
- See the subflow or action skill name.
- Select one or more assistants that can discover the action or subflow skill.
- Select one or more roles users must have to access the action or subflow skill.
- Set the advanced option to make the action or subflow skill discoverable.
- Set the advanced option to include the action or subflow skill in the list of topics.

Example conversational settings



When you set these options in Workflow Studio, the system also sets the corresponding options in Virtual Agent Designer.

Supported conversational subflows and actions input data types

Conversational subflows and actions support a limited number of input data types. To be compatible with conversational interfaces, an action or a subflow must only include inputs that use supported data types.

ServiceNow AI Platform data type name	Workflow Studio data type label
array.string	Array of Strings
boolean	True/False
calendar	Calendar Date/Time
choice	Choice
date	Date
datetime	Date/Time
document_id	Document ID
date_time	Date/Time

ServiceNow AI Platform data type name	Workflow Studio data type label
due_date	Due Date
email	Email
glide_date	Date
glide_time	Time
glide_date_time	Date/Time
GUID	Sys ID (GUID)
html	HTML
integer	Integer
long	Long
longint	Long Integer String
reference	Reference
schedule_date_time	Schedule Date/Time
string	String
string_full_utf8	String (Full UTF-8)
table_name	Table Name

Available conversational actions

Workflow Studio provides a set of actions that are preconfigured to be compatible with and callable by conversational interfaces such as Now Assist.

Name	Application Scope	User roles required
Add Comments and Work Notes	ITSM spoke	itil
Create Checklist from Template	Global	itil
Create Outage	Global	itil
Create Problem Record from Incident	Global	itil
Look up User	Microsoft Active Directory v2 Spoke	admin
Disable User	Microsoft Active Directory v2 Spoke	admin
Enable User	Microsoft Active Directory v2 Spoke	admin
Send Notification	Global	none
Create Meeting Space	Google Meet Spoke	admin
Update Meeting Space	Google Meet Spoke	admin

Exploring decision tables

Decision tables in Workflow Studio enable developers to decouple decision logic from their code by creating and maintaining decision rules.

Demo Tour

For an in-depth review and demo of decision table features and benefits, see Decision Builder Testing and Publishing | Creator Toolbox-ServiceNow.

Announcements

- As of the Washington DC release, Decision Builder is now part of Workflow Studio. Workflow Studio gives you a streamlined way to author, configure, and monitor processes, flows, subflows, playbooks, actions, and decision tables in one place.
- The core Decision Builder feature is still available with the ServiceNow AI Platform® by default, but the latest updates are available for download through the Workflow Studio application in the ServiceNow® Store.
- For more information about this update, see the Washington DC Decision Builder release notes.

Get started

Explore

Learn about decision table concepts and features

Configure

Configure environments, tools, and user access for decision tables in Workflow Studio

Use

Build decision tables in Workflow Studio

Reference

Get details about decision table properties and Domain Separation

Features

Note:
After you install Workflow Studio decision tables, you can only access decision tables previously created in the classic environment through Workflow Studio. You can no longer create or modify decision tables in the classic environment.

Workflow Studio provides an intuitive interface to create and manage decision tables, which store sets of decision rules.

Decision tables 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.

Administrators can use the delegated development capability to manage workload efficiently by assigning the delegated development role to developers or non-admin users. A delegated developer has more permissions than a user, but less than an admin. Non-admin designated users can create and manage decision tables at the application level. For more information, see Delegated development.

Decision tables in Workflow Studio enable you to do the following:

- Create inputs that read data at runtime from existing records, external data sources, or hard-coded values.
- Create and modify decision rules.
- Change the order of evaluation criteria to optimize results.

Benefits

Decision tables in Workflow Studio provide the following benefits:

- Build complex decisions easily with decision tables.
- Meet changing business requirements and quickly update decisions by modifying only the decision logic and not the application code.
- Increase efficiency by reusing decision logic across multiple applications.
- Evaluate data at runtime from multiple sources, including existing ServiceNow records and data from external sources.

Workflow Studio integration with other applications

Extend the capabilities of decision tables in Workflow Studio by integrating with applications in the ServiceNow AI Platform® suite.

Application integration

Application	Description	Reference
App Engine Studio	• App Engine Studio (AES) provides an Integrated Development Environment (IDE) application-like interface for app developers to build custom applications in one centralized location while maintaining your organization's brand and application development standards. • AES users can create decision tables using an AES wizard and access existing decision tables for their applications without leaving AES.	See Add a decision in App Engine Studio.

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

- Decision Builder in the ServiceNow Community
- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

Decision tables workflow

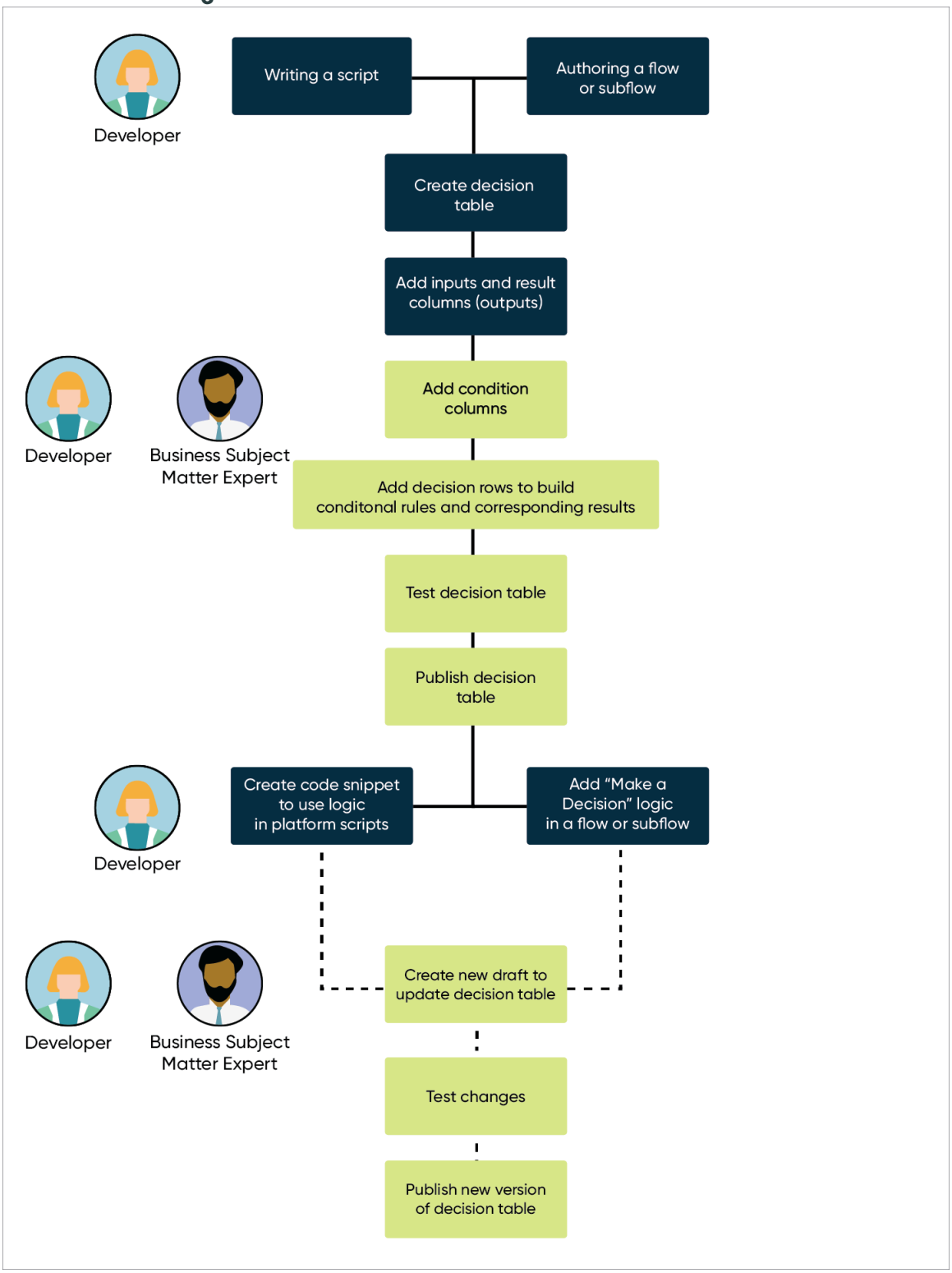
Learn how to create and maintain decision tables in Workflow Studio for use in flows, subflows, playbooks, and anywhere on the ServiceNow AI Platform where you write code.

Decision tables enable you to decouple decision logic from your code. This means that you can create a decision table in Workflow Studio with your application logic and then reference the decision table to execute the logic elsewhere, such as in a flow , playbook, or script. This decoupling enables you to create more efficient and maintainable flows and scripts and allows the decision logic to be managed directly by the relevant business subject matter expert.

Workflow for creating decision tables

The following illustration depicts the process for creating decision tables in Workflow Studio.

Process for creating decision tables



The workflow for creating decision tables in Workflow Studio is as follows:

1. A developer is writing a script or authoring a flow or subflow and realizes there is a need for a decision table. Decision tables are a more maintainable solution than hard-coded logic for the following scenarios:
   - There is a complex set of conditional, nested if/else or switch statements in the code.
   - The logic powering the code might change frequently.
   - There is a need to share the application logic with non-developers in a readable format.

- There is a need for application logic to be managed independently from where it is implemented.
- There is a need for application logic to be managed by non-developers.
- There is a need to apply the same logic in more than one flow, subflow, or script.
- There is application or business logic the developer wants to manage separately from the code.

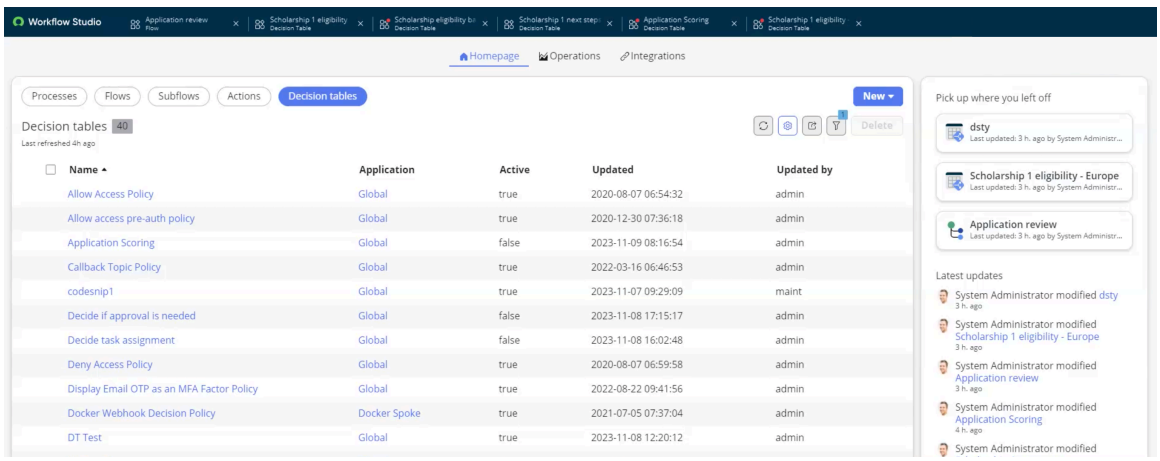
2. The developer creates the decision table in Workflow Studio and adds inputs and result columns (outputs for the decisions).
3. A developer or a business subject matter expert (SME) can perform the next steps.
   a. Add condition columns to structure the rules.
   b. Add decision rows to build conditional rules and corresponding results.
   c. Test the decision table.
   d. Publish the decision table.
4. The developer can use the decision table in a few ways.
   - They can create a code snippet of the decision table and insert it into their code.
   - They can add **Make a decision** logic in their flow or subflow and reference the decision table.
   - They can add a decision table to a playbook using the **Make a Decision - First Match** activity.
5. If the decision table must be updated to reflect new logic, the developer or business SME can create a new draft of the table. They can test the table using the new logic and then publish the new version. To create a Decision Table, See Create decision tables in Workflow Studio.

> **Note:**
> For any decision table that doesn't use draft authoring, any change the developer or business SME makes is automatically active.

Decision Builder user interface

Learn more about the decision tables user interface in Workflow Studio.

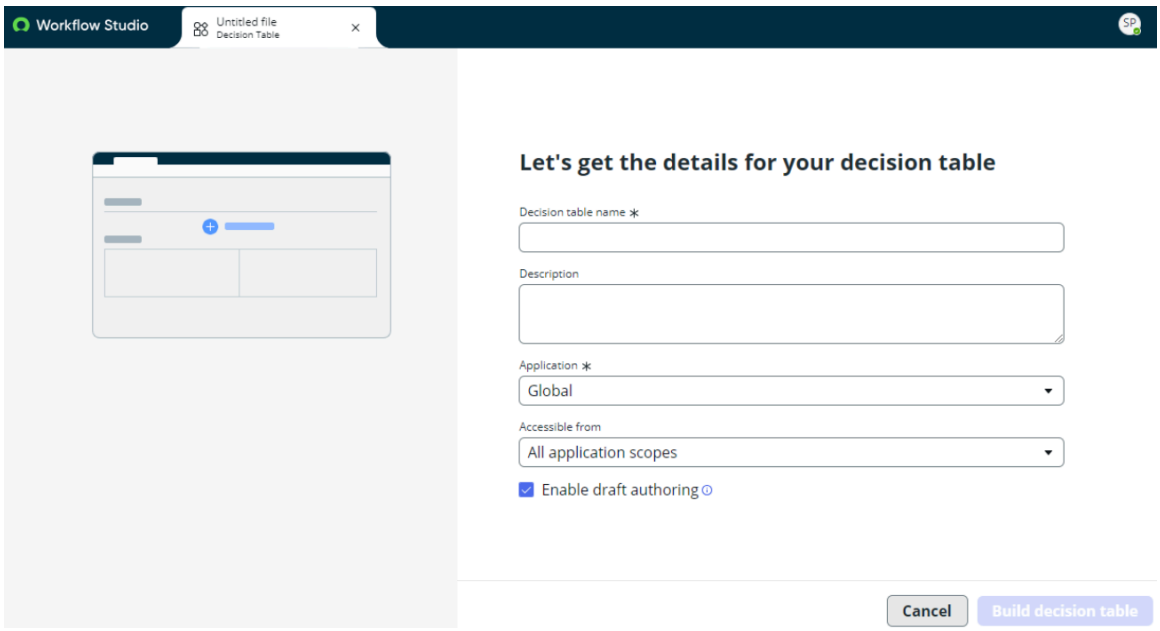
Decision tables homepage in Workflow Studio



You can do the following on the Decision tables homepage in Workflow Studio:

- Create a table by selecting **New** and selecting **Decision table**.
- Open a table that you've already created.
- Sort and filter the list of your decision tables.
- Edit the available columns by selecting the List Actions icon (⚙).
- Delete selected decision tables.

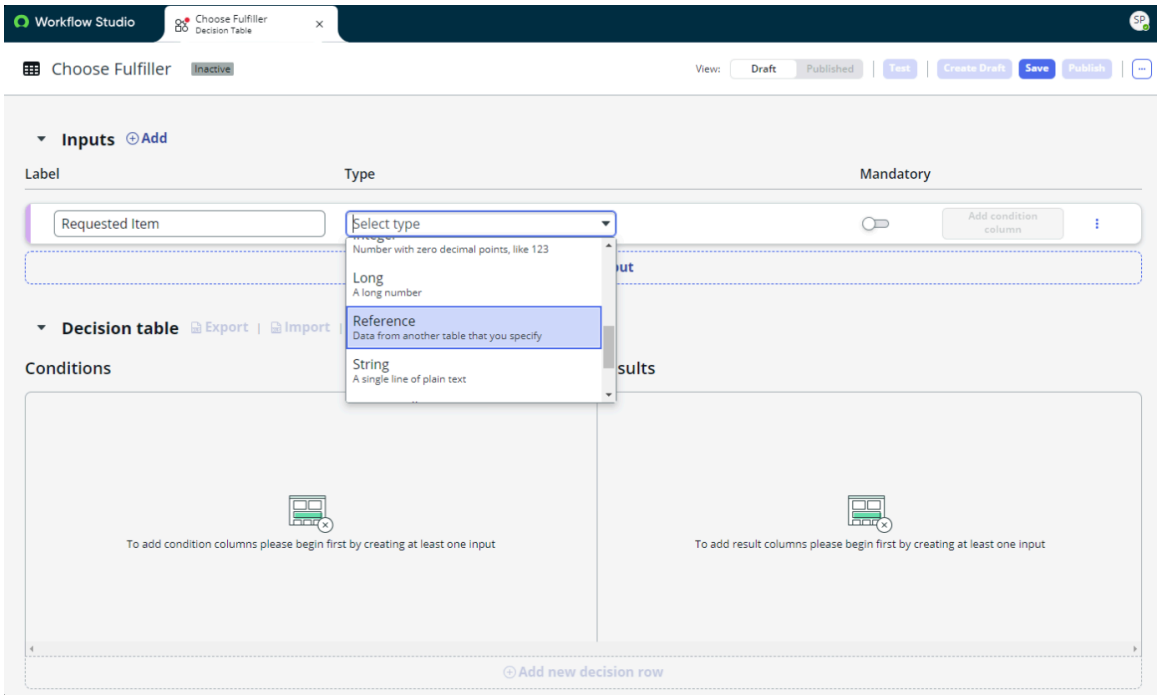
New decision table



You can do the following on a new decision table:

- Name your decision table and provide a description.
- Select an application to associate the decision table with.
- Select the scope that the decision table should be accessible from.
- Choose whether to enable draft authoring, which enables you to author decision tables in draft mode before publishing to lock it and make it available for use.

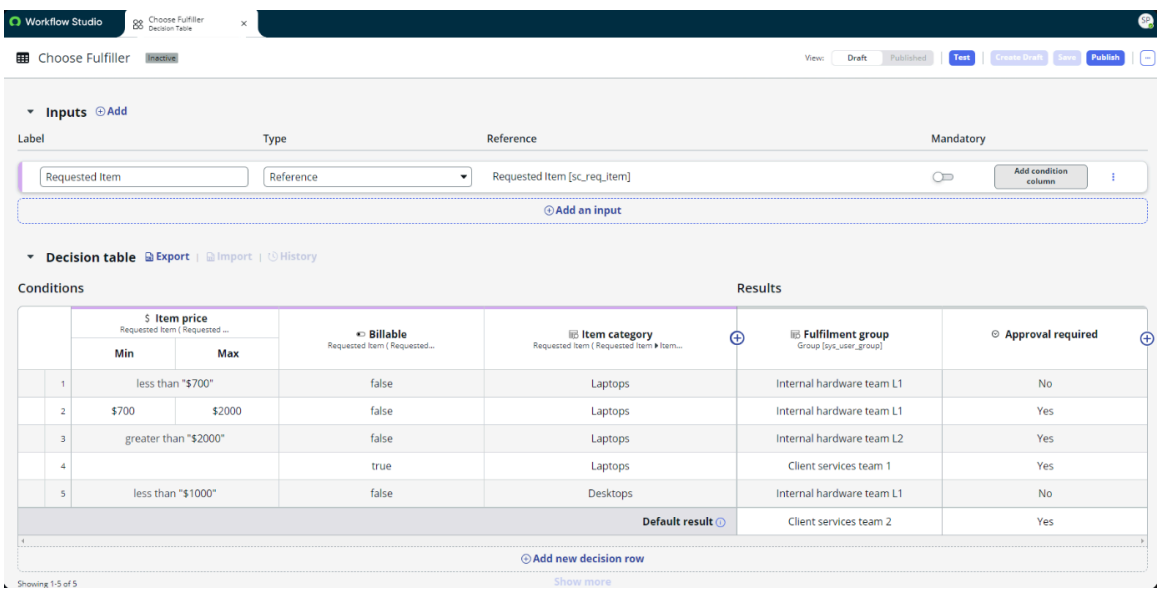
Decision table



When you first create a decision table, you can do the following:

- Add inputs, condition columns, and result columns.
- Add decision rows.
- Save the decision table.
- Export the table to Microsoft Excel and import back into Decision Builder. You can export and import only after saving the table.
- Edit the properties of the decision table.
- Delete the decision table.

Complete decision table



On a saved decision table, you can do the following:

- Publish the decision table.
- Edit the decision rows again in the Decision Builder interface.
- Edit the inputs, condition columns, or result columns.
- Export to Excel, edit the decision rows again, and import into Decision Builder to update the decision table.
- Test your decision table to make sure the rules provide the desired outcome for a given set of input data.

Configuring Workflow Studio

Enable access to Workflow Studio and allow management of workflow applications.

The Workflow Studio application is a ServiceNow AI Platform feature that is active by default, and that is also available for installation and update from the ServiceNow Store.

Each workflow application within Workflow Studio has its own installation and subscription requirements.

Playbooks

Playbooks requires a separate subscription. For activation information, see Activate playbooks.

Flows and subflows

Flows and subflows are included with Workflow Studio, which is a ServiceNow AI Platform feature that is active by default, and that is available for installation and update from the ServiceNow Store.

Actions

Actions are included with Workflow Studio, which is a ServiceNow AI Platform feature that is active by default, and that is available for installation and update from the ServiceNow Store.

Data stream actions

Data stream actions require a separate subscription to Integration Hub. For activation information, see Request Integration Hub.

Decision Builder

Decision Builder is a ServiceNow AI Platform feature that is available for installation and update from the ServiceNow Store. For installation and update information, see Configuring decision tables.

Configuring playbooks

Set up and monitor playbooks and Playbook Experience.

Overview

1. You don't need to activate the core Workflow Studio Playbooks feature because it is already part of the ServiceNow AI Platform, but many features are only available with application updates from the ServiceNow Store.

   Download the latest updates for Playbooks from the ServiceNow Store. To get started more quickly, we recommend you download the Process Automation Demo Experience, Process Automation Content, and Playbook Experience applications as well.

2. Configure Playbooks user access and properties.

Activate playbooks

Activate the application to create playbooks for any of your use cases.

Trigger playbooks off of any platform or application tables that you have access to. These can be:

- Application tables
- Custom tables that extend the application tables
- Custom tables authorized by the application subscription

Each application subscription entitles you to create playbooks for its associated tables. Purchase subscriptions to access any additional tables that you want to trigger your playbooks from. If you already have a subscription to your application but you still can't create playbooks for your application's tables, enable the appropriate plugin.

Note:
The plugins you enable determine which tables are available for you to create playbooks.

See the following sections to learn how to activate Workflow Studio Playbooks for your application.

Activate Playbooks for App Engine

Activate Playbooks on your instance to create playbooks in App Engine.

Before you begin
Role required: admin

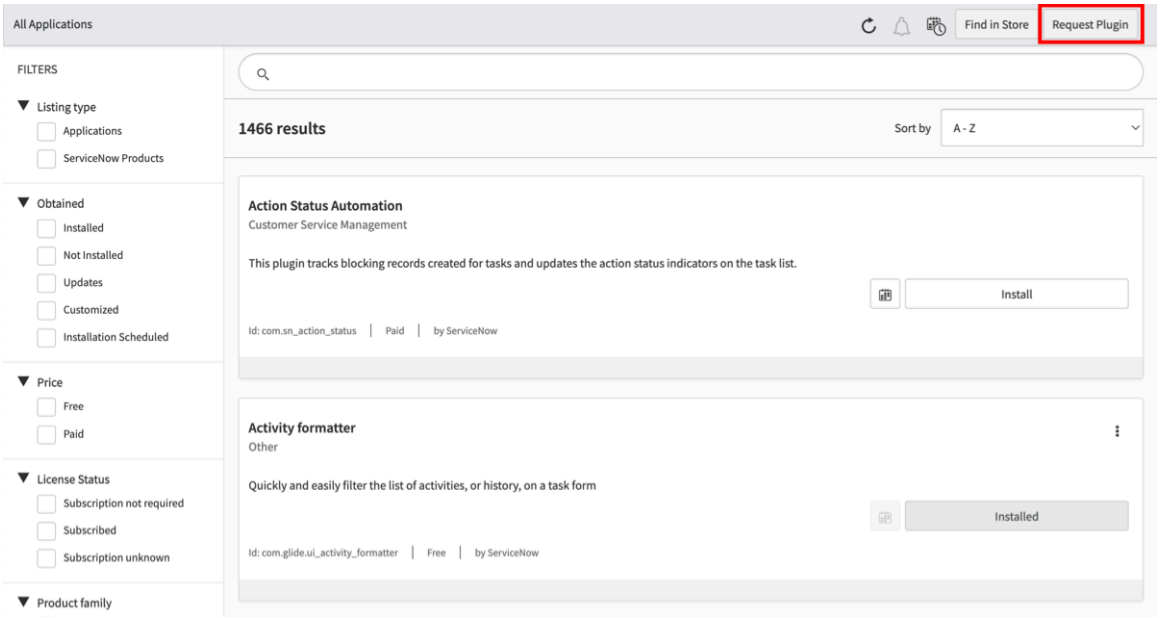
About this task
In order to create playbooks in App Engine, you must purchase a subscription to the ServiceNow AI Platform App Engine.

To purchase this subscription, contact your ServiceNow account manager. Your account manager can arrange to have the plugin activated on your organization's production and subproduction instances, generally within a few days.

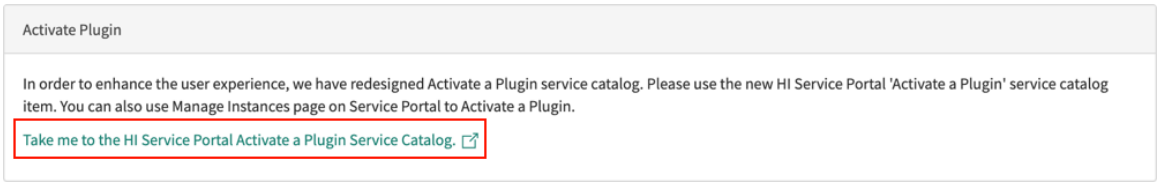
If you don't have an account manager, decide to delay activation after purchase, or want to evaluate the product on a subproduction instance without charge, follow these steps to enable the **Process Automation Designer for App Engine [com.glide.pad.license]** plugin:

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. On the All Applications page, select **Request Plugin** to open the **Activate Plugin** form on Now Support.



3. On Now Support, select the link to access the Now Support Service Portal Service Catalog.



4. Select your instance.
5. Select **Actions > Activate Plugin**.
6. On the **Activate Plugin** form, provide the following information.

Activate Plugin form

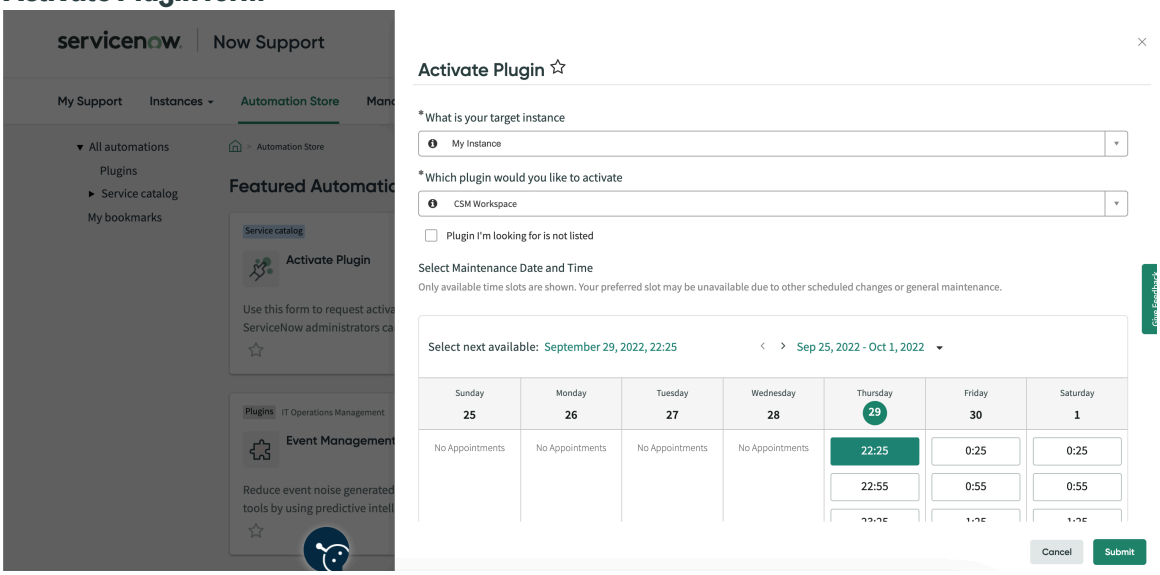
Field	Description
What is your target instance	Instance on which to activate the plugin.
Which plugin would you like to activate	Name of the plugin to activate. **Note:** If the system does not list the plugin you want or if you are activating the plugin on an OEM or on-premise instance, select the **Plugin I'm looking for is not listed** check box and then enter the name of the plugin.
Select Maintenance Date and Time	The date and time to activate the plugin.

Field	Description
	Note: Plugins are activated in two batches, once in the morning and once in the evening, on every business day in the US Pacific time zone. If the plugin must be activated at a specific time, enter the request in the **Reason/Comments** field.

Example

For example, see the following form to activate the CSM Workspace plugin on an instance named My Instance.

Activate Plugin form



7. Select **Submit**.
   For additional details about requesting a plugin, see Requesting a Plugin from the Service Catalog [KB0751715] article in the Now Support Knowledge Base.

Result

You can create triggers in Playbooks for custom tables that you create. Enabling the **Process Automation Designer for App Engine [com.glide.pad.license]** plugin lets you create playbooks for these tables and their extensions:

- Affected CIs [cmdb_outage_ci_mtom]
- Agent Capacity [awa_agent_capacity]
- Agent channel availability [awa_agent_channel_availability]
- Agent Presence [awa_agent_presence]
- Announcement [announcement]
- Assessment Category Result [asmt_category_result]
- Assessment Instance [asmt_assessment_instance]
- Assessment Metric [asmt_metric]
- Assessment Metric Template [asmt_template]

- Assessment Metric Type [asmt_metric_type]
- Assessment Metric Type Group [asmt_metric_type_group]
- Assessment Net Promoter Score [asmt_nps_result]
- Assessment Template Definition [asmt_template_definition]
- Assignment Eligibility [awa_eligibility_pool]
- Assignment Rule [awa_assignment_rule]
- Audit [cert_audit]
- Audit Result [cert_audit_result]
- AWA Agent Presence and Capacity [awa_agent_presence_capacity]
- AWA Document Size [awa_document_size]
- Base Configuration Item [cmdb]
- Building [cmn_building]
- Business Calendar [business_calendar]
- Certification Template [cert_template]
- CI Relation Filter [cmdb_rel_filter]
- CI Relationship [cmdb_rel_ci]
- CI Relationship Rollup [cmdb_rel_rollup]
- CI Relationship Type [cmdb_rel_type]
- CI/User Relationship Type [cmdb_rel_user_type]
- CIs Affected [task_ci]
- CMDB Group [cmdb_group]
- CMDB Group Event Queue [cmdb_group_event_queue]
- CMDB Group Type [cmdb_group_type]
- CMDB Health Configuration [cmdb_health_config]
- CMDB Health Metric [cmdb_health_metric]
- CMDB Health Result [cmdb_health_result]
- CMDB Health Scorecard [cmdb_health_scorecard]
- Company [core_company]
- Connection & Credential Aliases [sys_alias]
- Connection & Credential Templates [sys_alias_templates]
- Cost Center [cmn_cost_center]
- Country [core_country]
- Department [cmn_department]
- Direct Relationships [cmdb_related]
- [dms_document]
- Draft Document [draft_document]
- Follow On Task [cert_follow_on_task]
- Group [sys_user_group]
- Group Member [sys_user_grmember]

- Group Queue Priority [awa_group_queue_priority]
- Group Relationship [cmdb_rel_group]
- Group Role [sys_group_has_role]
- Group Skill [sys_group_has_skill]
- Guided Setup Task [gsw_task]
- Holiday [sys_holiday]
- Impacted CIs [task_cmdb_ci_service]
- Inbox Layout [awa_inbox_layout]
- Interaction [interaction]
- IP Address Pool [cmdb_ip_address_pool]
- IP Address Range [cmdb_ip_address_range]
- IP Address to DNS Name [cmdb_ip_address_dns_name]
- IP Service [cmdb_ip_service]
- KB Submission [kb_submission]
- Knowledge [kb_knowledge]
- Knowledge Base [kb_knowledge_base]
- Knowledge Category [kb_category]
- Knowledge Feedback [kb_feedback]
- Knowledge Feedback Task [kb_feedback_task]
- Knowledge Use [kb_use]
- Location [cmn_location]
- Metric [metric_instance]
- Model Category [cmdb_model_category]
- Offer Details [awa_offer_details]
- OS User [cmdb_os_user]
- Outage [cmdb_ci_outage]
- Page [sp_page]
- Peer Relationships [cmdb_peer]
- People Relationship [cmdb_rel_person]
- Presence State [awa_presence_state]
- Private Task [vtb_task]
- Product Model [cmdb_model]
- Queue [awa_queue]
- Related Entry [cmdb_related_entry]
- Report [sys_report]
- Role [sys_user_role]
- Roster [cmn_rota_roster]
- Rotation Escalation [cmn_rota_escalation]
- Scheduled Suite Run [sys_atf_schedule_run]

- Service [cmdb_ip_service_ci]
- Service Portal [sp_portal]
- Shift [cmn_rota]
- Shift Escalation Set [cmn_rota_escalation_set]
- Shift Escalation Step Definition [cmn_rota_esc_step_def]
- Skill Category [cmn_skill_category]
- Skill Level [cmn_skill_level]
- Skill Level Type [cmn_skill_level_type]
- Subscribers [cmdb_subscriber]
- Template [sys_template]
- Test [sys_atf_test]
- Test Results [sys_atf_test_result]
- Test Suite [sys_atf_test_suite]
- Test Suite Result [sys_atf_test_suite_result]
- Test Suite Test [sys_atf_test_suite_test]
- Test Template [sys_atf_test_template]
- Theme [sp_theme]
- Ticket [ticket]
- [universal_request]
- User [sys_user]
- User Skill [sys_user_has_skill]
- Vendor Type [vendor_type]
- Work Item [awa_work_item]
- Work Item Rejection [awa_work_item_rejection]
- Work Item Sizing [awa_work_item_sizing]
- Work Item Sort Order [awa_queue_item_sorting]

Note:
If you create a custom table such as My Table [x_my_table], you can create playbooks that trigger from it. However, you cannot create a playbook that triggers from a table belonging to another Process Automation Designer plugin.

Activate Playbooks for Customer Service Management (CSM)

Activate Workflow Studio Playbooks on your instance so that you can create Playbooks triggered by CSM tables.

Before you begin
Role required: admin

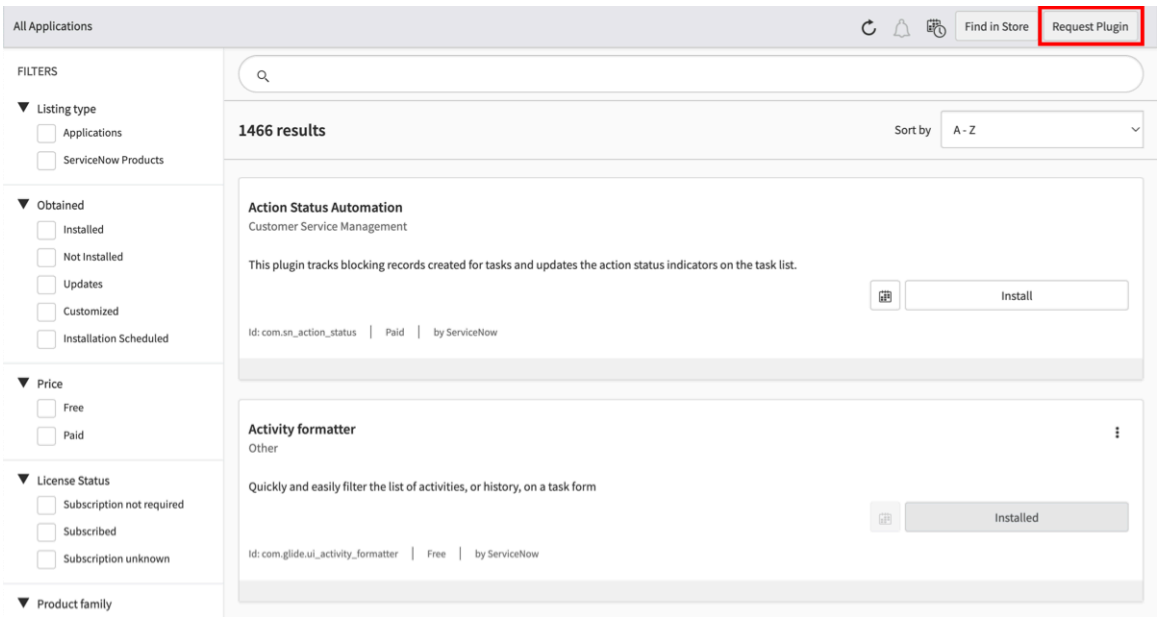
About this task
In order to create Playbooks in Workflow Studio that are triggered by CSM tables and custom tables that extend from them, you need to purchase a subscription to CSM.

To purchase this subscription, contact your ServiceNow account manager. Your account manager can arrange to have the plugin activated on your organization's production and subproduction instances, generally within a few days.

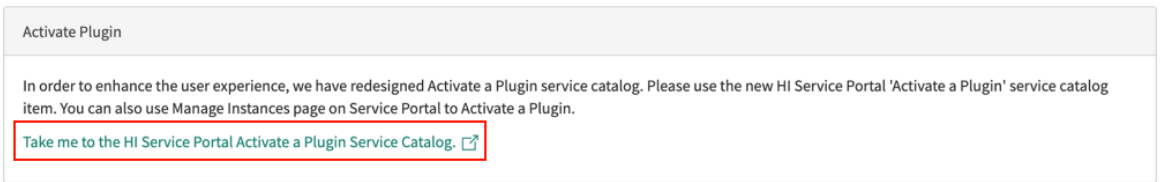
If you don't have an account manager, decide to delay activation after purchase, or want to evaluate the product on a subproduction instance without charge, follow these steps to enable the **Playbooks for Customer Service Management [com.sn_csm_playbook]** plugin:

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. On the All Applications page, select **Request Plugin** to open the **Activate Plugin** form on Now Support.



3. On Now Support, select the link to access the Now Support Service Portal Service Catalog.



4. Select your instance.
5. Select **Actions > Activate Plugin**.
6. On the **Activate Plugin** form, provide the following information.

Activate Plugin form

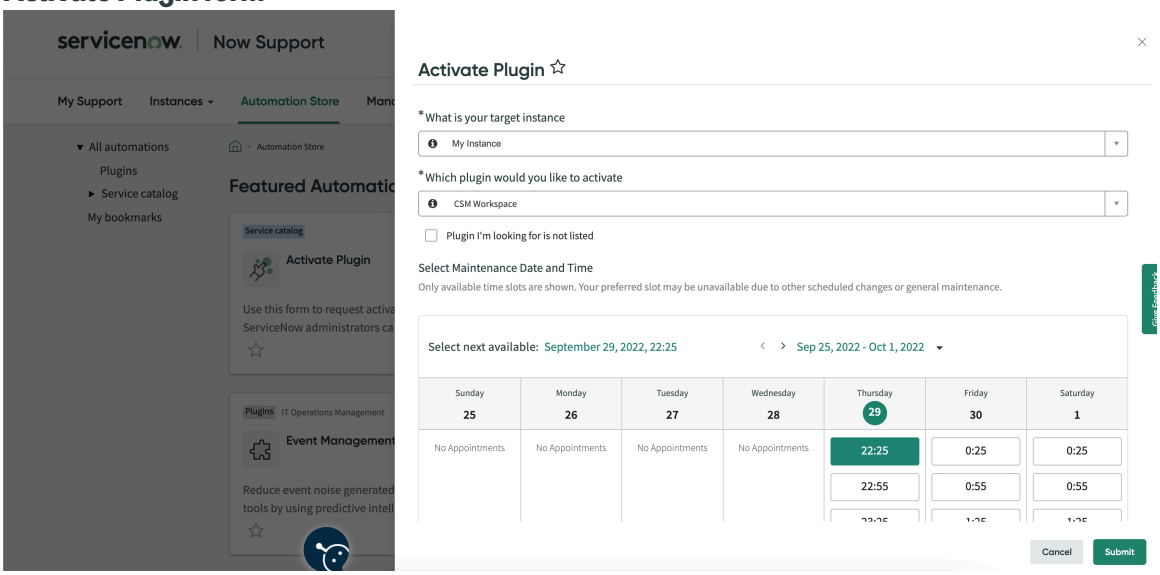
Field	Description
What is your target instance	Instance on which to activate the plugin.
Which plugin would you like to activate	Name of the plugin to activate.

Field	Description
	Note: If the system does not list the plugin you want or if you are activating the plugin on an OEM or on-premise instance, select the **Plugin I'm looking for is not listed** check box and then enter the name of the plugin.
Select Maintenance Date and Time	The date and time to activate the plugin. **Note:** Plugins are activated in two batches, once in the morning and once in the evening, on every business day in the US Pacific time zone. If the plugin must be activated at a specific time, enter the request in the **Reason/Comments** field.

Example

For example, see the following form to activate the CSM Workspace plugin on an instance named My Instance.

Activate Plugin form



7. Select **Submit**.
   For additional details about requesting a plugin, see Requesting a Plugin from the Service Catalog [KB0751715] article in the Now Support Knowledge Base.

Result

Enabling the **Playbooks for Customer Service Management [com.sn_csm_playbook]** plugin lets you create playbooks for these tables and their extensions:

- Account [customer_account]
- Case [sn_customerservice_case]

- Change Request [change_request]. Requires Customer Service with Service Management (com.sn_cs_sm)
- Consumer [csm_consumer]
- Contact [customer_contact]
- Escalation [sn_customerservice_escalation]
- Household [csm_household]
- Incident [incident]. Requires Customer Service with Service Management (com.sn_cs_sm)
- Interaction [interaction]
- Order [csm_order]. Requires Customer Service Management for Orders (com.snc.csm.order)
- Order Line Item [csm_order_line_item]. Requires Customer Service Management for Orders (com.snc.csm.order)
- Problem [problem]. Requires Customer Service with Service Management (com.sn_cs_sm)
- Request [sc_request]. Requires Customer Service with Request Management (com.sn_cs_sm_request)
- Service Organization [sn_customer_service_organization]. Requires Service Organization (com.snc.service_organization)
- Task [sn_customerservice_task]

Note:
If you create a custom table that extends a CSM table such as Case, you can create playbooks that trigger from it.

Activate Playbooks for Field Service Management

Activate Workflow Studio Playbooks on your instance so that you can create Playbooks triggered by tables.

Before you begin
Role required: Admin.

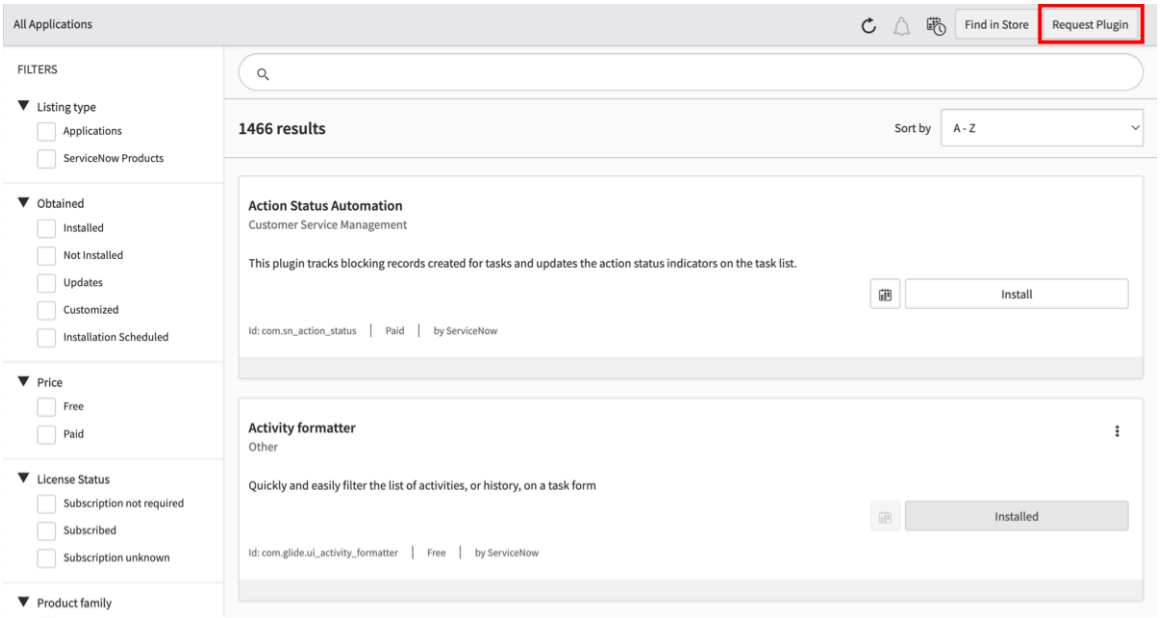
About this task
In order to create Playbooks in Workflow Studio that are triggered by Field Service Management tables, you must purchase a subscription to Field Service Management.

To purchase this subscription, contact your ServiceNow account manager. Your account manager can arrange to have the plugin activated on your organization's production and subproduction instances, generally within a few days.

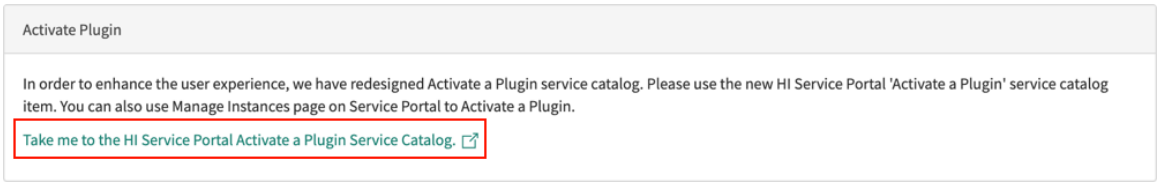
If you don't have an account manager, decide to delay activation after purchase, or want to evaluate the product on a subproduction instance without charge, follow these steps to enable the **Playbooks for Field Service Management [com.sn_fsm_playbook]** plugin:

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. On the All Applications page, select **Request Plugin** to open the **Activate Plugin** form on Now Support.



3. On Now Support, select the link to access the Now Support Service Portal Service Catalog.



4. Select your instance.
5. Select **Actions > Activate Plugin**.
6. On the **Activate Plugin** form, provide the following information.

Activate Plugin form

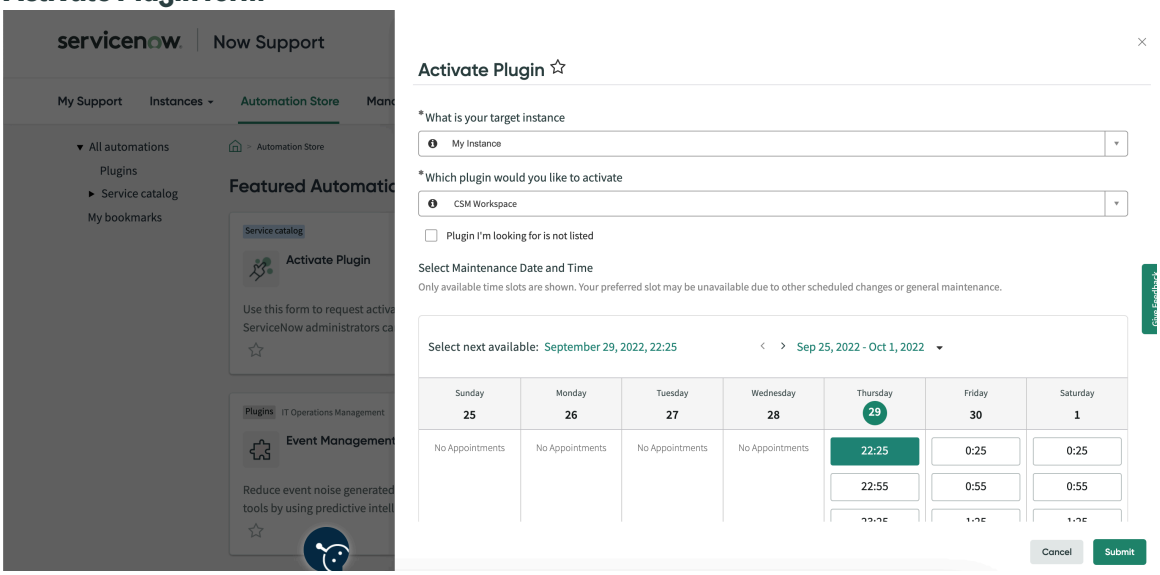
Field	Description
What is your target instance	Instance on which to activate the plugin.
Which plugin would you like to activate	Name of the plugin to activate. **Note:** If the system does not list the plugin you want or if you are activating the plugin on an OEM or on-premise instance, select the **Plugin I'm looking for is not listed** check box and then enter the name of the plugin.
Select Maintenance Date and Time	The date and time to activate the plugin.

Field	Description
	Note: Plugins are activated in two batches, once in the morning and once in the evening, on every business day in the US Pacific time zone. If the plugin must be activated at a specific time, enter the request in the **Reason/Comments** field.

Example

For example, see the following form to activate the CSM Workspace plugin on an instance named My Instance.

Activate Plugin form



7. Select **Submit**.
   For additional details about requesting a plugin, see Requesting a Plugin from the Service Catalog [KB0751715] article in the Now Support Knowledge Base.

Result

Enabling the **Playbooks for Field Service Management [com.sn_fsm_playbook]** plugin lets you create Playbooks for these tables and their extensions:

- Work Task Flow [sf_work_task]
- Work Order Flow [sf_work_order]
- Work Order Task [wm_Task]
- Work Order [wm_order]
- Work Order Model [cmdb_workorder_product_model]
- Work Task Model [cmdb_worktask_product_model]
- Work Type [wm_work_type]
- Agent Personal Schedule [agent_events]
- Appointment Booking [sn_apptmnt_booking_appointment_booking]

- Questionnaire [wm_questionnaire]
- Service Order Task [sm_task]
- Service Order Task Template Dependency [sm_m2m_task_template_dependency]
- Asset Usage [sm_asset_usage]
- Part Requirement [sm_part_requirement]
- Service Management Incidentals [sm_incidentals]

Administering playbooks

Monitor and troubleshoot issues by reviewing playbook executions.

These modules are provided for admins to monitor and manage Playbooks.

Today's Executions

Displays a list of records for Playbooks runs today.

Active Processes

Displays a list of all active Playbooks.

30+ Day Processes

Displays a list of all active Playbooks that are 30 days or older.

Activity Definitions

View or create activities.

Asset Configuration

View or create automation assets like flows, subflows, and actions.

Trigger Definitions

View or create triggers.

Data Definitions

View or create data definitions. Data definitions are pieces of information that you only reuse in a specific playbook, and are stored in the sys_flow_data_definition table.

User access to playbooks in Workflow Studio

Administrators can grant users access to playbooks by assigning delegated development permissions or directly assigning a user role. Administrators can also specify which features and content a user can access based on user roles.

Access by user role

Administrators can grant access to playbooks in Workflow Studio by directly assigning users the pd_author user role, which includes the role to view activity definitions.

Roles for playbooks in Workflow Studio

Role	Description	Contains Roles
playbook.admin	Enables users to:	• pd_author • pd_content_author

Roles for playbooks in Workflow Studio (continued)

Role	Description	Contains Roles
	• Create, update, and delete trigger definitions. • Launch Workflow Studio to create, activate, edit, and delete playbooks. • Create, edit, and delete activity definitions. • View the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.	• pd_trigger_author • pd_operator • pd_cancel
pd_author	Enables users to: • Launch Workflow Studio to create, activate, edit, and delete playbooks. • View all activity definitions. • View the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.	• pd_shared.user • playbook.write • playbook.activity_def_read
pd_content_author	Enables users to: • Create, edit, and delete activity definitions. • Create, edit, and delete trigger definitions. • View the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.	• pd_trigger_author • pd_shared.user • playbook.activity_def_read

Roles for playbooks in Workflow Studio (continued)

Role	Description	Contains Roles
pd_trigger_author	Enables users to create, update, and delete trigger definitions.	none
pd_operator	Enables users to view process executions, activity executions, and execution logs only.	none
pd_shared.user	Enables users to view the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.	none
pd_shared.admin	Enables users to edit the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.	pd_shared.user
pd_cancel	Enables users to cancel running playbooks without the playbook.admin role or write access to the parent record. For example if you want to grant an agent manager the ability to cancel playbooks, but not an agent.	none
pd_restarter	Enables users to restart active playbooks.	none
playbook.write	Enables users to: • Launch Workflow Studio to create, activate, edit, and delete playbooks. • View the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.	pd_shared.user

Roles for playbooks in Workflow Studio (continued)

Role	Description	Contains Roles
playbook.activity_def_read	Enables users to view all activity definitions.	none

A visual representation of where roles are contained:

- playbook.admin
  - pd_content_author
    - playbook.activity_def_read
    - pd_shared.user
    - pd_trigger_author
  - pd_operator
  - pd_cancel
  - pd_restarter
  - pd_author
    - playbook.write
      - pd_shared.user
      - sn_workflow_studio.workflow_studio_read

        Note:
        This role allows users to launch Workflow Studio, and is not managed by playbook administrators.

      - sn_diagram_builder.db_read

        Note:
        This role allows users to view playbooks in the diagram view in Workflow Studio, and is not managed by playbook administrators.

    - playbook.activity_def_read
  - pd_shared.admin
    - pd_shared.user
- delegated_developer

Delegated development access

Administrators can grant users access to Workflow Studio playbooks by creating an application and assigning users as developers with the playbook delegated development permission. Delegated development allows administrators to control whether playbook authors can access features normally restricted to admin users. For more information, see Developer permissions.

Role-based content filtering

Specify the user roles necessary to access Workflow Studio playbook content. For example, activity definitions. Manage content filtering by creating content definitions and content filtering rules. For more information, see Content filtering for playbooks.

Role-based activity definition access

Manage activity definition access by specifying the **Required Roles** to access an activity definition. To learn more about activity definitions, see Activity definitions.

Note:
Both playbook.admin and pd_content_author roles can edit activity definitions, but only the playbook.admin role can edit the **Required Roles** field.

Content filtering for playbooks

Specify which content a user can access based on the user's role.

Display only content that is relevant for a particular user, hiding content that is unnecessary or sensitive. Specify the Workflow Studio playbook content that you want to control access to and the role that a user must have to access it. For example, if a user with the guided_decision_builder role is creating a playbook, show only a relevant set of activities.

To implement content filtering, you need:

- Content definitions to specify types of content.
- Content filtering rules and roles to determine who can access the content.

There is one content definition for playbooks by default, the **Playbooks - All Activity Definitions** content definition. The **Playbooks - All Activity Definitions** content definition has two content filtering rules by default:

- **(Default) Playbook - Users with delegated_developer role can access all activity definitions**
- **(Default) Playbook - Users with playbook.activity_def_read role can access all activity definitions**

This means that users with the roles delegated_developer or playbook.activity_def_read role can access all activity definitions. Get started with content filtering by using default definitions and rules, or create your own.

Content definitions

Content definitions specify a type of Workflow Studio resource. Resources are key elements of Workflow Studio components, such as activity definitions for playbooks. Create content definitions to include an entire resource, or use a condition builder to refine your definitions. For example, the content definition for playbook activity definitions includes all activity definitions, but you could create a content definition that includes only the activity definitions that contain Guided Decision in the **Name** or **Package**.

You can further refine content definitions through tagging. Add resource tags to items in a resource list, then design your content definition to only include resources with that tag.

Content filtering rules

Content filtering rules specify the role that a user must have to access the content in a particular definition. Each rule associates a single user role with a single content definition. When a user accesses Workflow Studio playbooks, content filtering rules determine what activities the user may access based on the user's role.

Role-based activity definition access

Manage activity definition access by specifying the **Required Roles** to access an activity definition. To learn more about roles, see Playbooks roles. To learn more about activity definitions, see Activity definitions.

Note:
Both playbook.admin and pd_content_author roles can edit activity definitions, but only the playbook.admin role can edit the **Required Roles** field.

Restricted playbooks

Users cannot view a playbook that contains activities that they do not have access to. When a playbook contains restricted activities, the entire playbook is restricted.

Access summary

Resource filtered	User has role	User does not have role
Activity Definition	• The activity definition is visible to select when building a playbook. • The activity definition can be copied. • The activity definition can be modified.	• The activity definition is hidden and cannot be selected when building a playbook. • Playbooks with the activity definition are not visible.

Design considerations

Content definition roles for activity definitions

Give users access to the subset of activity definitions in a content definition by assigning the playbook.write role, not the pd_author role.

Configure content filtering definitions for playbooks

Specify which content a user can access by creating content definitions.

Before you begin

Content filtering requires some familiarity with user roles and Workflow Studio tables and records.

Role required: admin, playbook.admin

About this task

Filter Workflow Studio playbook content based on user role. Filtering content requires you to set up:

1. Content definitions describe the content that you want to filter. Content definitions specify types of Workflow Studio resources, such as activity definitions.
2. Content filtering rules to state the role a user must have to access the resource in a particular definition.

There is one content definition for playbooks by default, the **Playbooks - All Activity Definitions** content definition. The **Playbooks - All Activity Definitions** content definition has two content filtering rules by default:

- **(Default) Playbook - Users with delegated_developer role can access all activity definitions**
- **(Default) Playbook - Users with playbook.activity_def_read role can access all activity definitions**

This means that users with the roles delegated_developer or playbook.activity_def_read role can access all activity definitions. Get started with content filtering by using default definitions and rules, or create your own.

Procedure

1. To modify or create a content definition, navigate to **Process Automation** > **Flow Administration** > **Content Definitions**.

   Note:
   If you don't have access to Flow Administration, the Content Definitions module is directly under Process Automation instead.

2. Select the definition that you want to modify or click **New** to create one.
3. On the form, fill in the fields.

Workflow Resources form

Field	Description
Name	Name for the content definition.
Application	Application scope to which the content definition applies. This field is automatically set to the currently selected application scope. If no application scope is selected, the field is set to Global. If you set a specific application scope, then the content definition only applies to that application scope. If you select the Global application scope, then the content definition applies to all applications.

Field	Description
Table	Table containing the resource type that you're defining. For example, the Activity Definitions [sys_pd_activity_definition] table includes all the activity definitions available on your instance.
Conditions	Conditions used to filter the records in the table. For example, creating a condition where **[Name] [contains] [Guided Decisions]** returns only the activities that include the term Guided Decisions in the name.
Resource Tags	Tags used to filter the resources in the table.

4. Click **Submit**.

Configure content filtering rules for playbooks

Use content filtering rules to specify the role a user must have to access content.

Before you begin

Role required: admin, playbook.admin

Content filtering requires some familiarity with user roles and playbook tables and records.

About this task

Filter Workflow Studio playbook content based on user role. Filtering content requires you to set up:

1. Content definitions describe the content that you want to filter. Content definitions specify types of Workflow Studio resources, such as activity definitions.
2. Content filtering rules to state the role a user must have to access the resource in a particular definition.

There is one content definition for playbooks by default, the **Playbooks - All Activity Definitions** content definition. The **Playbooks - All Activity Definitions** content definition has two content filtering rules by default:

- **(Default) Playbook - Users with delegated_developer role can access all activity definitions**
- **(Default) Playbook - Users with playbook.activity_def_read role can access all activity definitions**

This means that users with the roles delegated_developer or playbook.activity_def_read role can access all activity definitions. Get started with content filtering by using default definitions and rules, or create your own.

Procedure

1. To modify or create a content filtering rule, navigate to **Process Automation** > **Flow Administration** > **Content Filtering Rules**.

   Note:
   If you don't have access to Flow Administration, the Content Filtering Rules module is directly under Process Automation instead.

2. Select the rule that you want to modify or click **New** to create one.
3. On the form, fill in the fields.

Workflow Resources Filter Rule form

Field	Description
Name	Name for the content filtering rule.
User Role	The role a user must have to access the content in the **Resource Definition** field.
Delegated Development Permission	The specific **Resource Path** that the user is a delegated developer for. Configure the **Resource Path** in the [sys_development_permission_set] table.
Active	Option to enable the rule.
Application	Application scope to which the content filtering rule applies. This field is automatically set to the currently selected application scope. If no application scope is selected, the field is set to Global. If you set a specific application scope, then the content filtering rule only applies to that application scope. If you select the Global application scope, then the content filtering rule applies to all applications.
Resource Definition	The name of the content definition that specifies the resource to filter.

4. Click **Submit**.

Archive process contexts

Improve database query performance by archiving unneeded records for playbooks that are in a **Complete**, **Error**, or **Cancelled** state.

Before you begin
Role required: admin or playbook.admin

About this task
Depending on how many playbooks you have run, you may have hundreds or thousands of records in your database that aren't used anymore. Reduce the number of records that your database queries have to search through by archiving the records for playbooks that are in a **Complete**, **Error**, or **Cancelled** state.

By default, context records are automatically archived for process executions that are:

- In a **Complete**, **Skipped**, or **Cancelled** state,
- For a playbook that hasn’t been edited within 14 days.

Note:
You can adjust the number of days. To learn more about configuring archive settings, see Configure archive settings for process contexts.

If you don't want context records to automatically be archived, see Turn off automated archiving.

Contexts for process executions that are in an **Error** state must be archived manually.

Note:
If you no longer need logs for a process execution that ended in an error, archive the context records.

When you archive context records for a process execution, the following records are compressed into a single JSON record:

- Element mapping [sys_element_mapping]
- Context log [sys_pd_context_logs]
- Variable value [sys_variable_value]
- Activity context [sys_pd_activity_context]
- Stage context [sys_pd_lane_context]

The JSON record can be viewed, but not edited. To learn more about viewing the JSON record for archived context data, see View archived process contexts.

Warning:
Archiving cannot be undone. If you need context logs or reports that use these records, don't archive. If automated archiving creates report problems, turn off the feature. To learn how to turn off automated archiving, see Turn off automated archiving.

Agents can still open playbooks to see historical data, such as why a playbook was canceled or who an activity was assigned to.

Flow context (sys_flow_context) records are also deleted automatically 2 weeks after completion. For more information about flow data retention, see Flow execution details retention.

Procedure

Manually archive context records for a process execution

1. Navigate to **All**, and enter **sys_pd_context.list** in the Filter field to open the [sys_pd_context] table.
2. Select the check boxes next to the executions that you want to archive context records for.
3. In the upper right corner, select **Archive Process Contexts** from the action menu.
4. Confirm that you want to archive the process contexts.

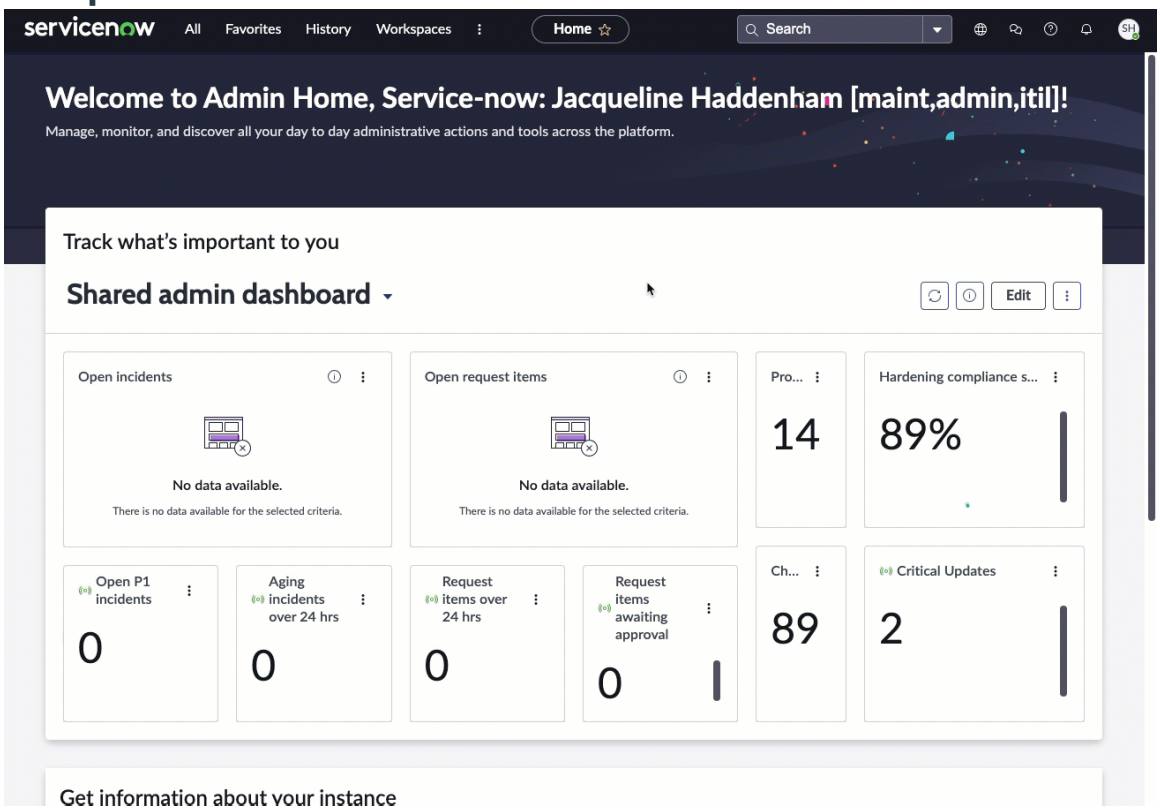
Result

The context records are archived for your selected process executions.

Trouble?

If you have a large number of context records to archive, records are archived in limited batches on an hourly basis, to avoid slowing down the instance.

Example: Archive context records for an execution



What to do next

Configure the form layout for process executions so that you can see the archived data. To learn more about configuring the form layout for a process execution, see View archived process contexts.

Configure archive settings for process contexts

Change the default settings for the automated archiving feature for process contexts.

Before you begin

Role required: admin or playbook.admin

If you want to turn off automated archiving instead, see Turn off automated archiving.

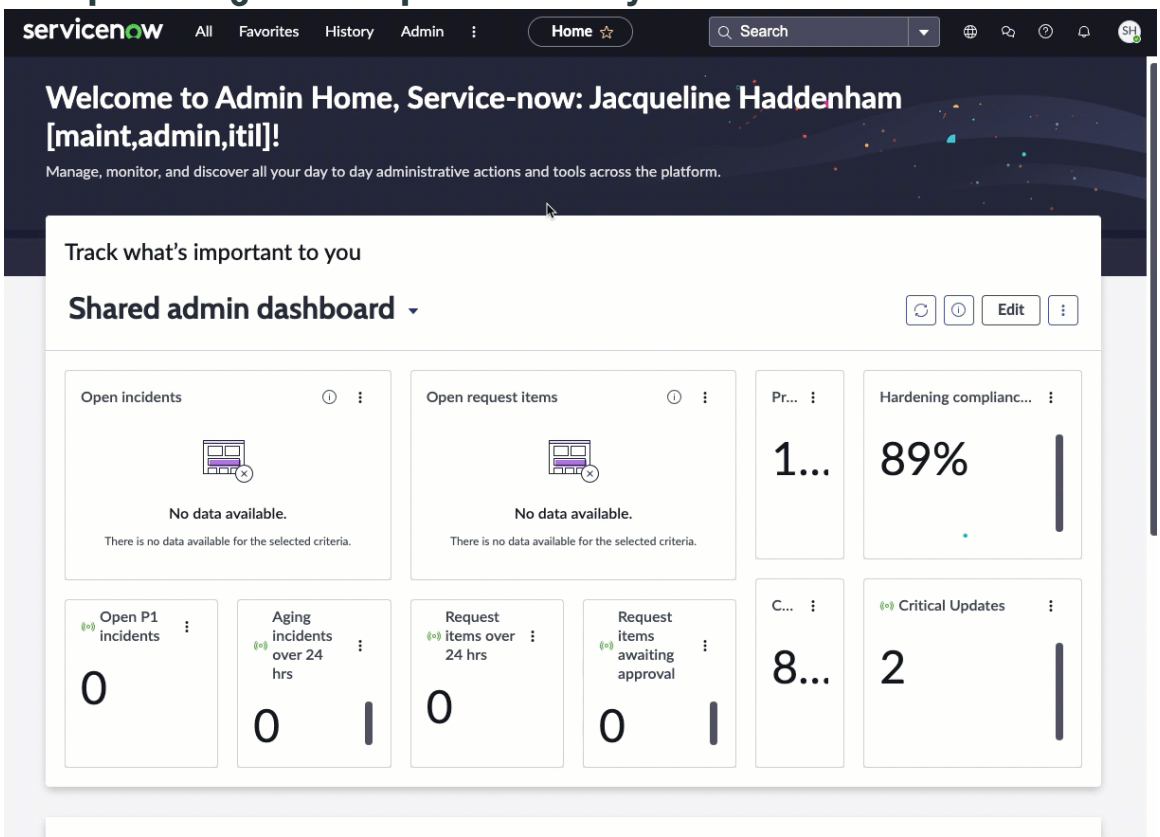
Procedure

1. Navigate to **All** > **System Properties** > **All Properties**.
2. Search for and open the **sn_pa_designer.data_retention_policy** property.
3. Select the **here** link in the warning message to edit the record.
4. Change the **Value** to the number of days a complete or cancelled playbook must be unedited before the context records for it can be archived.
   The default value is 14. If you want context records for executions to be archived as soon as a playbook is complete or cancelled, you can set the value to O. If you want to wait longer before context records can be archived, enter a greater number.

   Warning:
   Changing other fields in this record could potentially break automated archiving.

5. Select **Update**.
   Your change is saved.

Example: Change the wait period to 30 days



Turn off automated archiving

Turn off the automatic archiving of context records for your complete and cancelled playbooks.

Before you begin

Role required: admin or playbook.admin

If you want to change the number of days to before a playbook is archived instead, see Configure archive settings for process contexts.

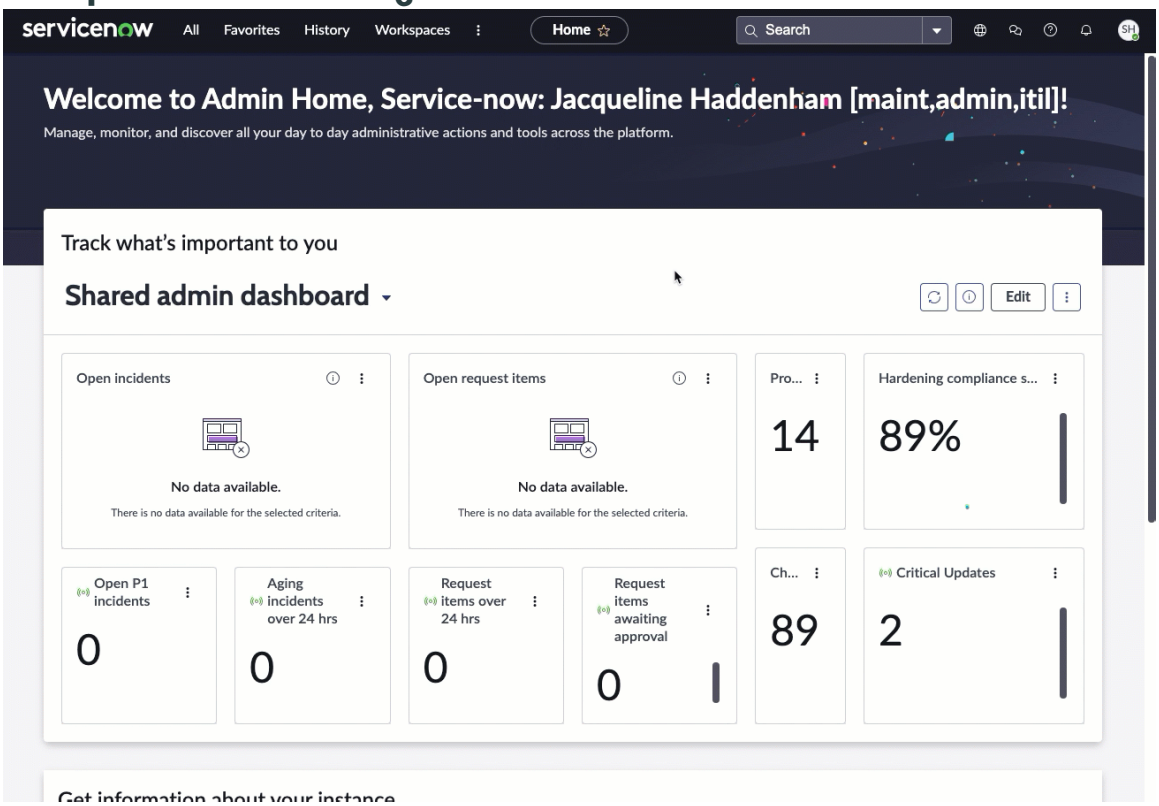
Procedure

1. Navigate to **All** > **System properies** > **All Properties**.
2. Search for and open the **sn_pa_designer.enableDataRetentionFeatures** property.
3. Select the **here** link in the warning message to edit the record.
4. Set the **Value** to **true**.

> **Warning:**
> Changing other fields in this record could potentially break automated archiving.

5. Select **Update**.
   Automated archiving is turned off.

Example: Turn off archiving



View archived process contexts

Configure the form layout for a process execution so that you can see the JSON record for archived context records.

Before you begin

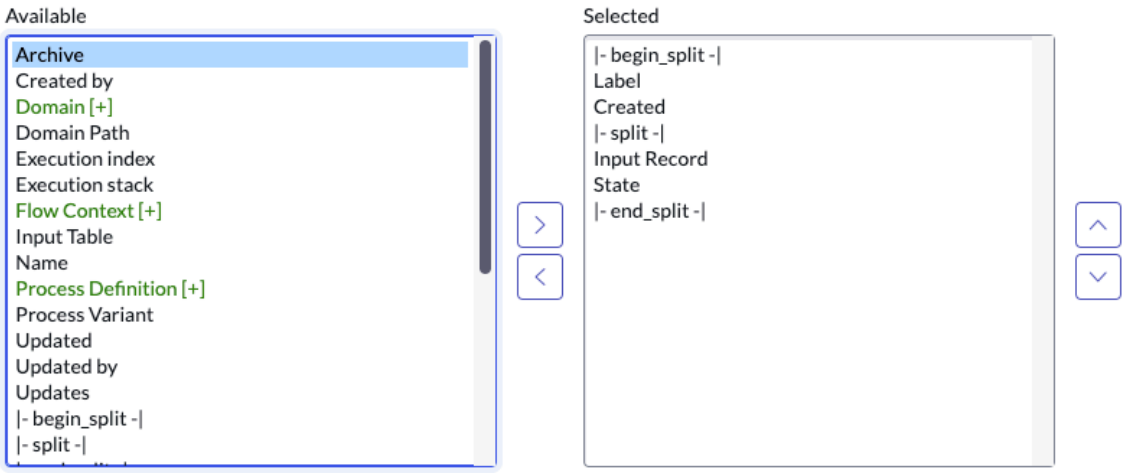
Role required: admin or playbook.admin

About this task

To view the archived context records for a process execution record, you must configure the Form Layout for process execution records. If you haven't archived any context records for a process execution and want to, see Archive process contexts.

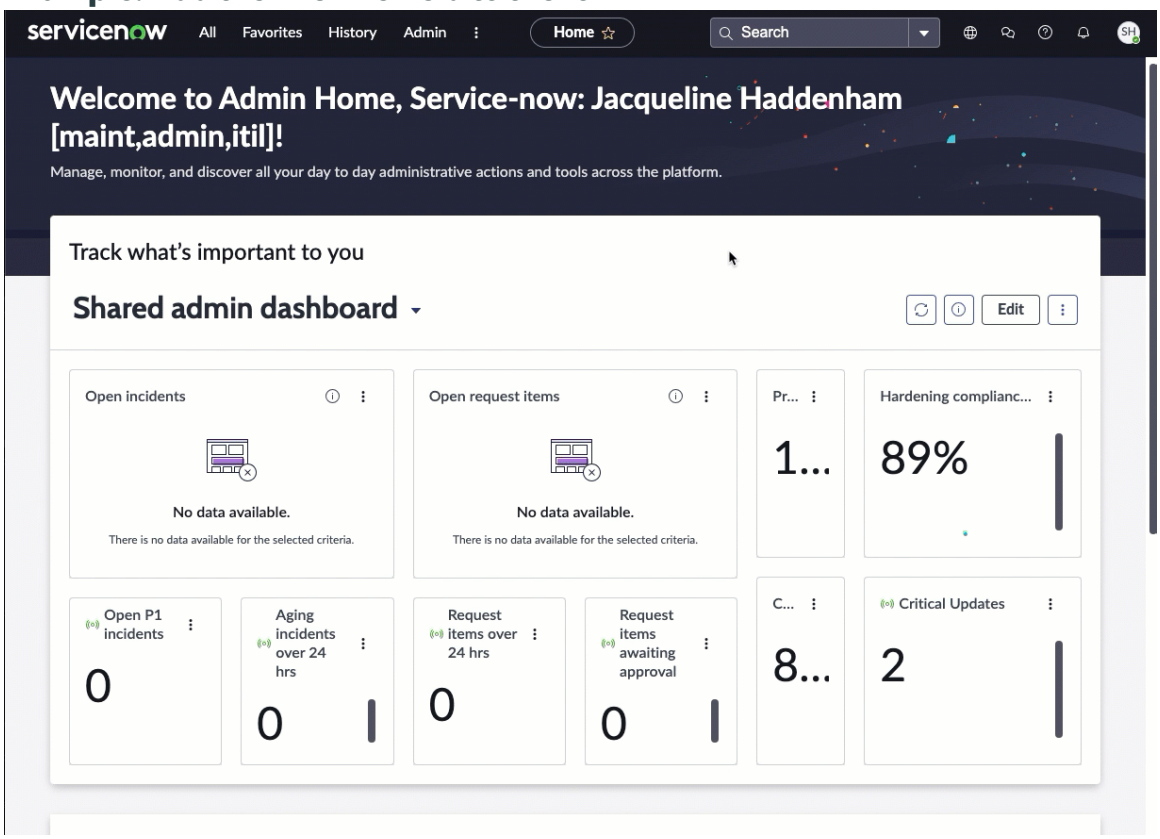
Procedure

1. Navigate to **All**, and enter **sys_pd_context.list** in the Filter field to open the [sys_pd_context] table.
2. Open any process execution.
3. Open the form context menu ( ).
4. Select **Configure** > **Form Layout**.
5. In the Available list, double-click **Archive** to move it to the **Selected** list.



6. Select **Save**.
   The Archive field appears in all process execution records.
7. Open a process execution with archived process context records.
   You can view the JSON record of the archived context records for the process execution in the Archive field.

Example: Add the Archive field to the form



Create a Data Definition

Use data definitions to collect and use pieces of information later in a playbook.

Before you begin

Important:
As of the 26.1 release, the **Collect user data** activity is no longer available in the activity picker. The activity will continue to function wherever it is used, but for new activities use the **Questionnaire** activity instead. The **Questionnaire** activity does not require you to create a data definition. To learn more about the **Questionnaire** activity, see Questionnaire activity.

Role required: admin, flow_designer

You will be working in the ServiceNow AI Platform to complete this task.

About this task

A data definition is the information that you want an agent or fulfiller to collect during a playbook run, and is the key input of the **Collect user data** activity. Playbook authors define the data they want an agent or fulfiller to collect in the `sys_flow_data_definition` table. When an agent or fulfiller collects the information, the information is stored in the `sys_flow_data` table for use later during the playbook run, instead of in the record table.

Only use a data definition if:

- The data is only needed downstream during a single playbook run. It's collected, used, and never needed again.
- You don't need to run any reports on the collected data. If you need any metrics or reports on the collected data, create a table and use the User Form activity instead.

For example, you may have multiple teams that perform activities. One team enters the inputs for a created data definition when they perform a **Collect user data** activity, and then a second team uses the collected inputs to complete the playbook, and the information is not needed afterwards.

Procedure

1. Navigate to **All** > **Process Automation** > **Process Automation Administration** > **Data Definitions**.
2. Select **New** to create a new data definition.
3. Give your new data definition a name.

   Note:
   Data definitions have the same scope as other metadata tables, by default.

4. Right-click in the record header to **Save**.
   If you select the **Submit** button, you are taken back to the **Data Definitions** list and will need to select your new data definition to re-open it.
5. Add fields for information that you want an agent to collect.
   a. In the **Flow Data Variables** table, select **New**.
   b. Enter the required fields.

Option	Description
Type	The type of input the agent is collecting for a field. For example, string, reference, integer, etc.

Option	Description
Label	The label of the field in the UI, during the play book run. The label can consist of any text.
Column name	The name of the input being collected. Spaces cannot be used to delimit words.
Max Length	The maximum length a string value can be entered for a string type of field. The variable can store longer strings than it can display.
Application	The application scope for the data variable. It is always set to Global, and cannot be changed.

Optional configurations

6. **Optional:** Under the **Default Value** tab, specify the value used when a playbook does not provide a value.
7. Right-click in the record header to **Save**.

Result
The data definition can now be used when configuring activities in Playbooks in Workflow Studio.

Example:
During a playbook run, you can use data definitions to potentially:

- Collect a shipping address, then reference the address when generating a shipping label.
- Ask the user "yes" or "no" questions, and determine subsequent activities based on the user's responses.

What to do next
Configure a **Collect user data** activity in Workflow Studio Playbooks to use your new data definition.

Configuring flows

Configure user access, API access, and properties for Workflow Studio flows.

Activation

Workflow Studio flows, subflows, and actions are ServiceNow AI Platform features that are active by default.

Configuration options

Flow Administration options

Configuration option	Reference
Retain flow execution details	Flow execution details retention
Create flow-specific execution settings	Flow execution settings
Enable flow reporting	Activate flow reporting
View flow and action dashboards	FDIH Dashboard
Set flow priority	Flow priority

User access options

Configuration option	Reference
Grant users access to build flows by role	User access to Workflow Studio flows
Restrict access to individual flow and action features by custom roles	Manage access to Workflow Studio flow features
Filter flow and action content by role	Content filtering for Workflow Studio flows

API access options

Configuration option	Reference
Grant access to flow and action APIs	API access to Workflow Studio flows
Create code snippets	Create code snippets for flows, subflows, and actions
Create a client callable flow, subflow, or action	Create a client callable flow, subflow, or action

Restricted caller access options

Configuration option	Reference
Manage cross-scope access to flows and actions	Restricted caller access to Workflow Studio flows
Upgrade restricted caller access	Upgrade restricted caller access privileges for flows and actions

Update Workflow Studio options

Configuration option	Reference
Update flow diagramming	Update to the latest version of Flow Diagramming
Update Workflow Studio and all of its dependencies	Update to the latest version of Workflow Studio

Flow administration

Identify and troubleshoot potential issues by reviewing action and flow executions, their result state, and their runtime duration.

Workflow Studio provides administrators and flow operators these modules to manage flows.

Today's Executions

Displays a list of flow context records for flows run today. Use this information to identify flows run today.

Active flows

Displays a list of flow context records for running flows where the **State** is **Waiting**. Use this information to identify flows that are waiting for a trigger or condition to continue.

Event Queue

Displays a filtered list of event records where the **Queue** is **flow_engine** and the **State** is **Ready**. Use this information to identify flows that are waiting on event processing.

Operations Dashboard

Displays a responsive dashboard containing a count of flows run and the average flow runtime. View statistics for today or over the last 30 days. Use this information to determine the health and performance of flow execution.

Settings

Displays the list of Flow and Action Settings records. Use this table to identify which actions and flows have had reporting disabled. Create records on this table to control whether reporting is done on an action or flow.

Properties

Displays the system properties used to configure how the system processes flows.

Complex Object Templates

Displays templates for complex objects that you can use in flows with Integration Hub integrations. To learn more about creating and using complex object templates, see create an action and the Object data type.

Usage Overview

Displays transaction counts between your instance and third-party systems. The Usage Overview is not available in the base system and requires the ServiceNow® Integration Hub subscription.

Flow execution details retention

Due to the large amount of data consumed by flow execution details, your instance uses data retention policies to delete this data after a set time period.

Generating flow execution details

By default, the system only generates execution details when you run a test. To generate flow execution details, see Activate flow reporting

Scheduled table cleanup

The system uses a standard table cleaner records to determine when to remove execution details. Each type of flow execution content is stored in its own table and has its own retention period. Once a record is older than its default retention period, it is deleted if it is in a completed state. To learn more about how to find and configure table cleaner records, see Table cleaner.

Flow reporting data tables

Table	Description	Default retention period
sys_flow_context	Parent table that stores all Workflow Studio context records and their associated process plans. Context records store the state and references to the data used	• 2 weeks for completed flows • 6 weeks for flows in the error or cancelled state

Flow reporting data tables (continued)

Table	Description	Default retention period
	to run a flow or action. See the child tables for context records in specific states.	**Warning:** Deactivating or increasing the retention period of Flow Context records can negatively impact instance performance. Retaining more flow contexts can impact flow performance and the ability to use new flow features.
sys_flow_context_chunk	Child table that stores context records and runtime data for currently running flows and actions. This table replaces the sys_json_chunk table as the location to store data for active context records. A running flow or action can be in one of these states. • Continue Sync • In Progress • Queued • Waiting **Danger:** Do not change or delete data in this table. Workflow Studio uses this table for flows and actions that are in an active state.	The system removes these records when the flow stops running and creates an entry in the sys_flow_context_chunk_archive table.
sys_flow_context_chunk_archive	Child table that stores context records and runtime data for flows and actions that have stopped running. This table replaces the sys_json_chunk table as the location to store data for inactive context records. A stopped flow or action can be in one of these states. • Cancelled • Complete • Error	Removed when the associated sys_flow_context record is removed. • 2 weeks for completed flows • 6 weeks for flows in the error or cancelled state

Flow reporting data tables (continued)

Table	Description	Default retention period
	Note: Workflow Studio uses this table for flows and actions that are in an inactive state.	
sys_flow_report_doc	Parent table that stores references to Workflow Studio context records that have execution detail reporting data available. See the child tables for the execution details of flows and actions in specific states.	The system removes these records when it removes the parent context record from sys_flow_context.
sys_flow_report_doc_chunk	Child table that stores the reporting data and execution details for currently running flows and actions. A running flow or action can be in one of these states. • Continue Sync • In Progress • Queued • Waiting **Danger:** Do not change or delete data in this table. Workflow Studio uses this table for flows and actions that are in an active state.	The system removes these records when the flow stops running and creates an entry in the sys_flow_report_doc_chunk_archive table.
sys_flow_report_doc_chunk_archive	Child table that stores the reporting data and execution details for flows and actions that have stopped running. A stopped flow or action can be in one of these states. • Cancelled • Complete • Error	The system removes these records when it removes the parent context record from sys_flow_context_chunk_archive. • 2 weeks for completed flows • 6 weeks for flows in the error or cancelled state

Flow reporting data tables (continued)

Table	Description	Default retention period
	Note: This table replaces the sys_json_chunk table as the location to store reporting data for inactive execution details.	
sys_json_chunk	Table that stores compiled process plans for future, running, and completed flows and actions created prior to the San Diego release. **Danger:** Do not change or delete data in this table. Workflow Studio uses this table for flows and actions that are in an active state.	The system removed these records when it removed the parent record.
sys_flow_log	Table that stores replicated log entries from the Log [syslog] table. Enables users to correlate logs with flow contexts.	The system removes these records in 28 days when the table is rotated or when it removes the context record, whichever comes first. The table rotation on sys_flow_log is configurable. For more information, see Table rotation.
sys_flow_plan_context_binding	Table that stores a unique identifier for each context record and the trigger that started it. Whenever a	The system removes these records 12 months after creation.

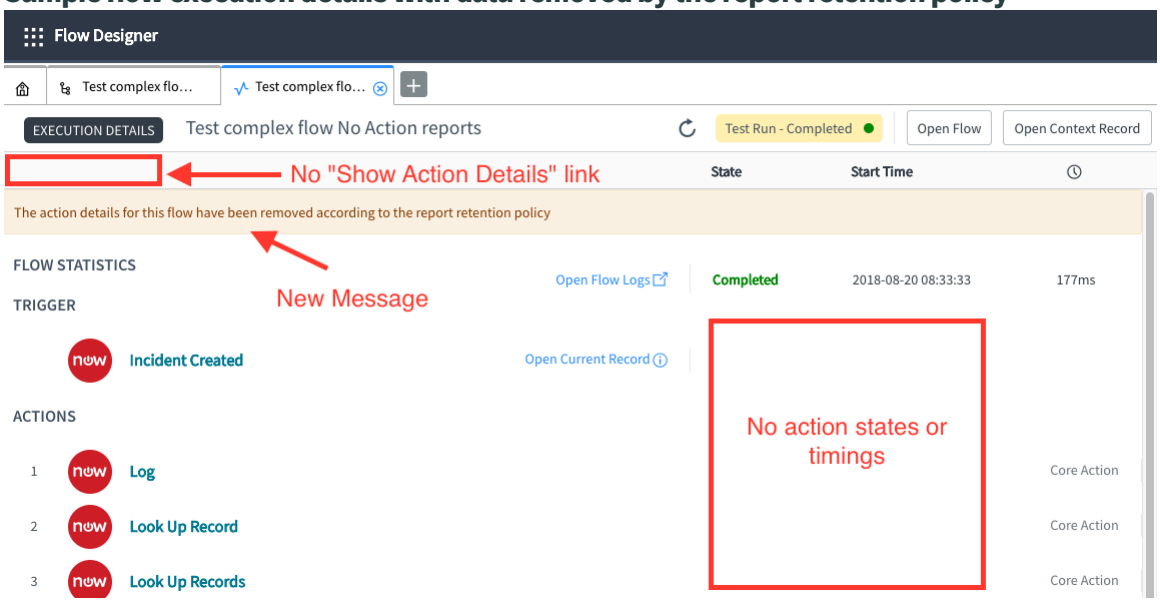
Flow reporting data tables (continued)

Table	Description	Default retention period
	triggering event occurs, the system calculates the unique identifier and compares it to a sys_flow_plan_context_binding record. If the calculated unique identifier matches an existing sys_flow_plan_context_binding record, then the triggered flow is not started. **Note:** This unique identifier is used to determine when to run flows with either the "run once" or "for each unique change" conditions.	**Important:** The system may rerun flows whose unique identifier was removed by the retention policy. For example, if the trigger conditions of a "run once" flow are met, and the sys_flow_plan_context_binding record was removed, then a new unique identifier is created and the flow runs.

Unavailable flow data

A message displays at the top of the flow report to indicate that action reports are not available for a flow because of table cleanup. The **Show Action Details** link and Action states are not available in this case. A similar message is shown to indicate when reporting for a flow has been deactivated. In this case, a link to the report settings also displays.

Sample flow execution details with data removed by the report retention policy



Recovery options

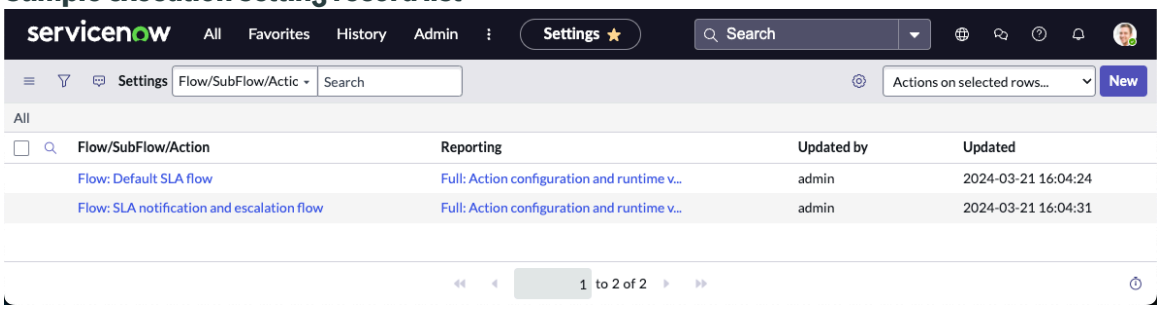
Contact Customer Service and Support to restore data from an instance backup.

To know the period until which a data recovery request is accepted, see the Instance Backup and Recovery [KB0547654] article in the Now Support Knowledge Base.

Flow execution settings

Specify the runtime settings for an individual flow, subflow, or action such as the run priority, reporting level, and logging level.

Sample execution setting record list



You can create an execution settings record for each flow, subflow, or action for which you want to collect execution details. An execution setting record [sys_flow_execution_setting] overrides the reporting system properties for a particular item. For example, you can keep the system wide reporting level off, and create execution setting records just for the items that you want to collect execution details.

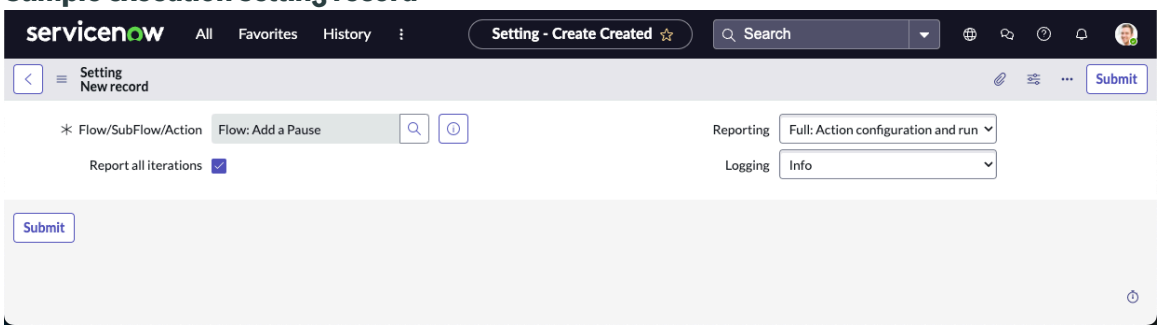
You can create as many execution settings records as you like. The system generates execution details each time the flow, subflow, or action is directly run. Actions or subflows that run from a parent flow use the execution settings record of the parent flow.

Note:
You can call actions and subflows directly using the action API or script API.

Execution setting records are available from the Flow administration module.

Execution setting record fields

Sample execution setting record



An execution setting record [sys_flow_execution_setting] contains these fields.

Execution setting record fields

Field	Description
Flow/SubFlow/Action	Individual flow, subflow, or action to which the execution settings apply.
Report all iterations	Option to gather execution details for all iterations of a loop that have run rather than just storing execution details for the first and last run iterations. If a flow pauses between iterations, then the execution details include

Execution setting record fields (continued)

Field	Description
	all iterations that have run. When this option is false, the execution details only include the first and last iterations run.
Reporting	Reporting level at which to gather execution details.
Logging	Logging level threshold required for Workflow Studio to write a message generated by the flow engine to the Flow engine log entries [sys_flow_log] table.
Flow Priority	Priority value used to run a background flow. For more information about setting flow priority, see Flow priority. **Note:** This field is hidden by default. For more information about showing the flow priority field, see Show flow priority field. .

General guidelines

Use these general guidelines when creating execution setting records to capture execution details.

Avoid reporting on production instances

Avoid performance issues on your production instance by only activating and configuring reporting on a non-production instance that you use for testing. Generating and storing execution details consumes instance resources, which can lower performance.

Create execution setting records on a non-production test instance

Create execution setting records on a non-production testing instance to minimize the performance impact of generating and storing execution details.

Reduce the amount of memory consumed by flow loops

Reduce the amount of memory consumed by generating execution details by not using the option to report all iterations. The more iterations you report on, the more memory is required.

Test flows, subflows, and actions to generate execution details

Test your flows, subflows to generate execution details at the debug level. Only deploy your items to a production instance after you verify that they generate the data you want and that they perform as expected.

Use flow reporting and the report all iterations options only during flow troubleshooting

Use the flow reporting and report all iterations options only when you need to troubleshoot a flow. These settings generate information that consume instance resources. To reduce their performance impact, either turn off these settings or delete the settings record altogether when you are done troubleshooting the flow.

Activate flow reporting

Choose whether to generate execution details for all flows and actions run, just for individual flows and actions, or just when you test a flow or action. Specify the level of detail the execution details contain.

Activate reporting for an individual flow, subflow, or action

Generate execution details for an individual flow, subflow, or action every time it runs, not just during testing.

Before you begin

Role required: flow_operator or admin

About this task

Warning:
To avoid performance issues on your production instance, activate and configure reporting on the non-production instance that you use for testing.

You can activate reporting for an individual flow, subflow, or action by creating a record on the Settings [sys_flow_execution_setting] table. Each Settings record specifies the flow, subflow, or action to generate execution details for and the level of detail to use. You can create as many Settings records as you want. The system generates execution details each time the flow, subflow, or action is directly run. Actions or subflows that run from a parent flow use the Settings record of the parent flow.

Note:
You can call actions and subflows directly using the action API or script API.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Administration** > **Settings**.
   The system displays the list of individual items for which flow reporting is activated.
2. In the **Flow/SubFlow/Action** field, select the lookup icon ( ) to select the type item for which you want to activate reporting.
   The system displays a dialog box to select the type and specific instance.
3. In the **Table name** field, select the matching table for the item.
4. In the **Document** field, select the lookup icon ( ).
   The system displays a list of items of the matching type.
5. Select the individual flow, subflow, or action for which you want to activate reporting.
6. Select **OK** to close the dialog box.
7. In the **Reporting** field, select the level of runtime data to generate and display in flow execution details.

   Off

   The system doesn't generate flow execution details. The system only generates execution details when you run a test.

   Note:
   Testing an action or flow generates execution details at the Trace level.

   Basic: Runtime states and durations only

The system generates runtime execution details for each flow, subflow, and action run. You can see the runtime state and duration for these basic items. You can also see configuration and runtime values for flow triggers, subflow inputs, and subflow outputs.

Full: Action configuration and runtime values (for debugging only)

The system generates configuration and runtime execution details for each flow, subflow, and action run. You can see the runtime state, duration, input values, and output values for all items. For custom actions, you can also see the runtime state, duration, input values, and output values of its steps. You can also see the configuration values for flow triggers, subflows, actions, and steps that are part of a custom action.

> **Important:**
> Only users with the fd_read_operations_all role can see configuration and runtime information such as record values in the flow execution details. Users without this role will only see basic details about the state and duration.

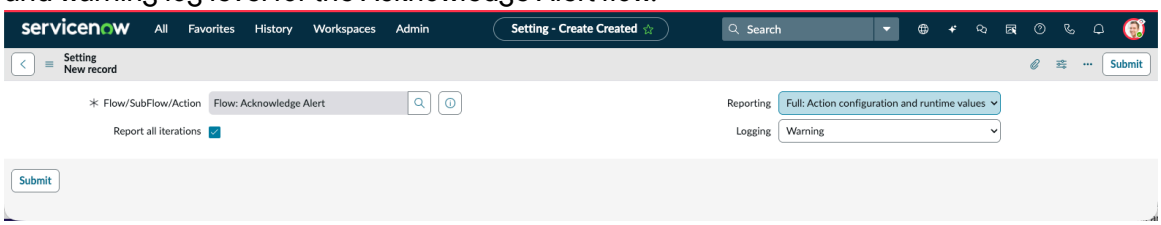
Trace: All values (for testing and Support only)

The system generates configuration and runtime execution details for each flow, subflow, action, and step run. You can see the runtime state, duration, input values, and output values for all items. You can also see the configuration values for flow triggers, subflows, actions, and steps.

> **Important:**
> Only users with the fd_read_operations_all role can see configuration and runtime information such as record values in the flow execution details. Users without this role will only see basic details about the state and duration. Testing an action or flow generates execution details at the Trace level.

8. Select **Submit**.

Example
For example, this setting enables the full reporting level and warning log level for the Acknowledge Alert flow.



Result
Workflow Studio generates execution details for the individual flow, subflow, or action.

Activate reporting for all items

Generate execution details for all items that Workflow Studio runs rather than just generating execution details during testing.

Before you begin
Role required: flow_designer, action_designer, admin

About this task

Important:
To avoid performance issues on your production instance, activate and configure reporting on the non-production instance that you use for testing.

By default, the system only generates execution details when you run a test.
You can activate reporting for all items that Workflow Studio runs by setting the *com.snc.process_flow.reporting.level* system property.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Administration** > **Properties**.
2. Set the property **Level of reporting data generated by the flow engine**.

Off

The system doesn't generate flow execution details. The system only generates execution details when you run a test.

Note:
Testing an action or flow generates execution details at the Trace level.

Basic: Runtime states and durations only

The system generates runtime execution details for each flow, subflow, and action run. You can see the runtime state and duration for these basic items. You can also see configuration and runtime values for flow triggers, subflow inputs, and subflow outputs.

Full: Action configuration and runtime values (for debugging only)

The system generates configuration and runtime execution details for each flow, subflow, and action run. You can see the runtime state, duration, input values, and output values for all items. For custom actions, you can also see the runtime state, duration, input values, and output values of its steps. You can also see the configuration values for flow triggers, subflows, actions, and steps that are part of a custom action.

Important:
Only users with the fd_read_operations_all role can see configuration and runtime information such as record values in the flow execution details. Users without this role will only see basic details about the state and duration.

Trace: All values (for testing and Support only)

The system generates configuration and runtime execution details for each flow, subflow, action, and step run. You can see the runtime state, duration, input values, and output values for all items. You can also see the configuration values for flow triggers, subflows, actions, and steps.

Important:
Only users with the fd_read_operations_all role can see configuration and runtime information such as record values in the flow execution details. Users without this role will only see basic details about the state and duration. Testing an action or flow generates execution details at the Trace level.

> **Warning:**
> Avoid enabling the Full reporting option on a production instance. Full reporting generates execution details for every flow and action run on the instance. Creating and storing these execution details consumes system memory and can lower system performance. Instead, only enable reporting for specific flows and actions or test them on a non-production instance.

3. Select **Save**.

Result

Workflow Studio generates execution details for all items you specified in the system property.

FDIH Dashboard

Use a single dashboard to view usage, execution, and debug information for Workflow Studio and Integration Hub transactions. Open links to related Workflow Studio properties, logs, events, and editors.

With the FDIH Dashboard, you can manage and see the status of Workflow Studio and Integration Hub transactions. Access to the dashboard is role-based.

The dashboard is divided into six sections.

FDIH Dashboard header

View the state of flow reporting and the log level being replicated to the flow logs. Use links to open the system properties table, the system logs, the outbound HTTP logs, Workflow Studio propertgies, and flow settings.

Flow Usage

View the number of flows run and their state over various time periods. For example, you can view the flow executions in the complete state over the last 14 days. Use links to Workflow Studio, system properties, event handlers, and flow settings to manage how flows are run.

IH Usage

View Integration Hub transactions and features used over the last month. For example, you can view the top spokes used over the last month.

Flow Executions

View a list of flow executions by creation date. View information about flow, its runtime duration, state, calling source, and associated application. For example, you can filter flows by calling source to see which ones have been tested recently.

MID Executions

View Integration Hub probes that ran on a MID Server. View information about the agent, source, topic, queue, state, and processing state. For example, you can filter probes run by source of queue. Use links to see MID Server queues sorted by source and available MID Servers.

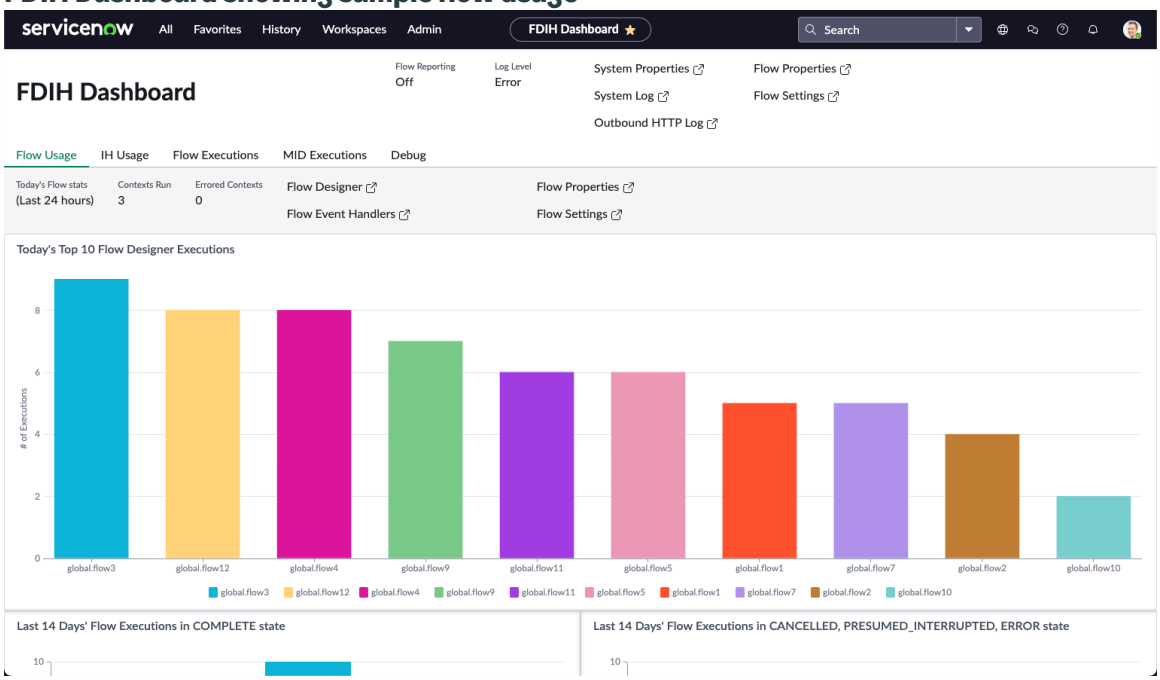
Debug

View a list of flow jobs in the ready state that are waiting to run. Identify flows that need manual intervention or redesign to complete. For example, use the list of waiting flows to identify a performance issue.

There's no specific order in which you must view the reports, set filters, or customize the columns in the report tables.

The following example shows how the dashboard data is organized into sections, charts, filters, and graphs. The filters enable you to drill beyond a chart or a graph to view more detailed data.

FDIH Dashboard showing sample flow usage



Get insights from the FDIH Dashboard

Use the FDIH Dashboard to get insights on flow executions and historical performance.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Administration** > **FDIH Dashboard**.
2. To view reports, select the Flow Usage section.

> **Note:**
> There's no order in which you must view the reports, set the filters, or customize the columns in the report tables. You can complete one or more of the following steps in any order. The Flow Usage section is selected by default.

3. **Optional:** To drill down and view more details in a report, select the graph or chart.
4. **Optional:** Use ServiceNow AI Platform list controls to filter list reports.

   Example

   For example, select the filter parameter icon (, and then select **Show matching** to show records that have the same selected value.
5. **Optional:** To customize the columns in the report table, do the following actions:
   a. Drill down the report.
   b. Click the column settings icon ( ).
   c. Click **Edit columns**.
   d. In the Available columns section, click the column name.

e. To include a column, click the include column icon ( ).

f. To exclude a column from the table, in the Selected columns section, click the remove column icon ( ).

g. To change the positions of the columns in the report table, in the Selected columns section, click and drag the drag column icon ( ).

h. To apply the changes, click **OK**.

6. **Optional:** To reset the column width, click the column settings icon ( ), and then click **Reset widths**.

Flow priority

Specify the priority that you want a background flow to have in relation to other flows waiting to be run. Run a group of higher priority flows before running any lower priority flows.

You can only set a flow priority for flows that run in the background. Background flows run from the next available worker thread. By default, Workflow Studio can use up to half of the available worker threads to run background flows. If there's no available worker thread to run a flow, the flow is queued until there's an available worker thread to run it.

You can set background flows to have one of these priority values.

- High
- Medium
- Low

By default, background flows run with medium priority.

Note:
Flows that run in the foreground run in the current thread and don't use flow priority to determine run order.

Setting a flow priority determines the order that worker threads pick flows from the queue. Worker threads run several higher priority flows before running a lower priority flow. This priority scheme ensures that some lower priority flows get run even when there are higher priority flows in the queue. When there's a large work queue to run, most low-priority flows must wait until the high priority flows are run. After the high priority flows have run, the worker threads can run lower priority flows.

Should a flow remain in the queue for more than one minute, the system delegates the flow to another node to run. When a flow is delegated to another node, it loses its priority value. A delegated node pulls flows from the event queue in chronological order. The delegated node runs the oldest flows first and then runs the newest flows. Generally, delegating flows to run from another node only happens when all the available worker threads on a node are busy.

Note:
Worker threads run all system events, not just flow events.

Flows also lose their priority value when they pause for any reason. Flows that resume from a pause run at the default Medium priority. For example, a flow could start running at high priority, then pauses to Wait for a Duration. When the flow resumes running, it runs at a Medium priority.

Automatic pausing of low-priority flows

By default, the system checks for cases where high-priority flows are being blocked by running lower priority flows. Whenever a low-priority runs for longer than two minutes, the system checks whether any higher priority flows have run in the last five minutes. If no high-priority flows have run recently, then the system checks the number of flows waiting to run in the event queue. If there's a backlog of at least 100 high-priority flows waiting to run in the event queue, the system pauses the running low-priority flow. Pausing a flow preserves its context and data. A paused flow returns to the event queue to run when there's a worker available to process it.

If your low-priority flows aren't running as expected, review the number of high-priority flows that your system generates and runs. See the design considerations for when to set a flow to high or low priority. If your low-priority flows are still not running, you can disable the pausing of low-priority with a system property. See Workflow Studio flow system properties to change the value of the com.glide.hub.pause_low_priority_flows_enabled property.

Quick API support

The quick API method retains the priority settings. Flows run by the quick API method run at the same priority that was previously defined for the flow.

Design considerations

Follow these design considerations when setting flow priority.

Avoid setting all flows to run at high priority

Use a mix of priorities rather than set all flows to high priority. Worker threads use the relative priority between flows to select work. If all of your flows run at high priority, then there are no lower-priority flows to make wait.

Avoid setting flow priority for flows that have to pause

Keep flows that have to pause at the default medium priority since a flow that pauses loses its priority value when it resumes running.

Use high priority for business critical flows

Limit high-priority to flows that have high business value, run rarely, and have a short runtime. Avoid setting high-volume flows to high priority as doing so limits the number of worker threads available to run other flows. A long-running high priority flow can also reduce the worker threads available to run other flows.

Use low priority for high-volume flows

Run high-volume flows at low priority so that other time-sensitive flows can run first. Low-priority flows shouldn't be time-sensitive.

Use medium priority for time-sensitive flows

Use the default flow priority when a flow has some time urgency when compared to other flows.

Show flow priority field

Configure the Setting list and form to show the Flow Priority field.

Before you begin

Role required: admin or personalize_form and personalize_list

About this task

The Flow Priority field is hidden by default. Configure the Setting table list and form layout to show the field.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Administration** > **Settings**.
   The system displays the list of Setting records.
2. Select the Personalize List icon.
   The system displays the Personalize List Columns dialog.
3. Select the **Flow Priority** field from the Available column and add it to the Selected column in the order you want it to display.
4. Select **OK**.
   The Flow Priority field is visible on the list.
5. From the Setting record list, select **New**.
   The system displays a new Setting record.
6. From the Additional actions menu, select **Configure** > **Form Layout**.
   The system displays the Configuring Setting form dialog.
7. Select the **Flow Priority** field from the Available column and add it to the Selected column in the order you want it to display.
8. Select **Save**.
   The Flow Priority field is visible on the form.

What to do next

Set the Flow Priority from the list or form.

Set flow priority

Determine the order in which worker threads run background flows. Use a mix of flow priorities to distinguish between flows to defer and flows to run first.

Before you begin

- Role required: flow_designer or admin
- Show flow priority field

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. Select the **Flow** data pill filter.
3. Open the flow whose priority you want to set.
4. Select **Edit flow**.
5. From the More Actions menu, select **Properties**.
6. From the Flow properties window, expand **Advanced Options**.
7. From **Flow Priority Default**, select the priority value you want for the flow.

   You can set background flows to have one of these priority values.

   - High
   - Medium
   - Low

   By default, background flows run with medium priority.

Note:
Flows that run in the foreground run with an elevated priority above background flows.

8. Select **Update**.

Result
The flow runs at the priority you specify as long as it does not pause or get delegated to another node to run.

What to do next
Use the FDIH dashboard to review the flow executions and verify that your flows are running as expected.

See related flows for action

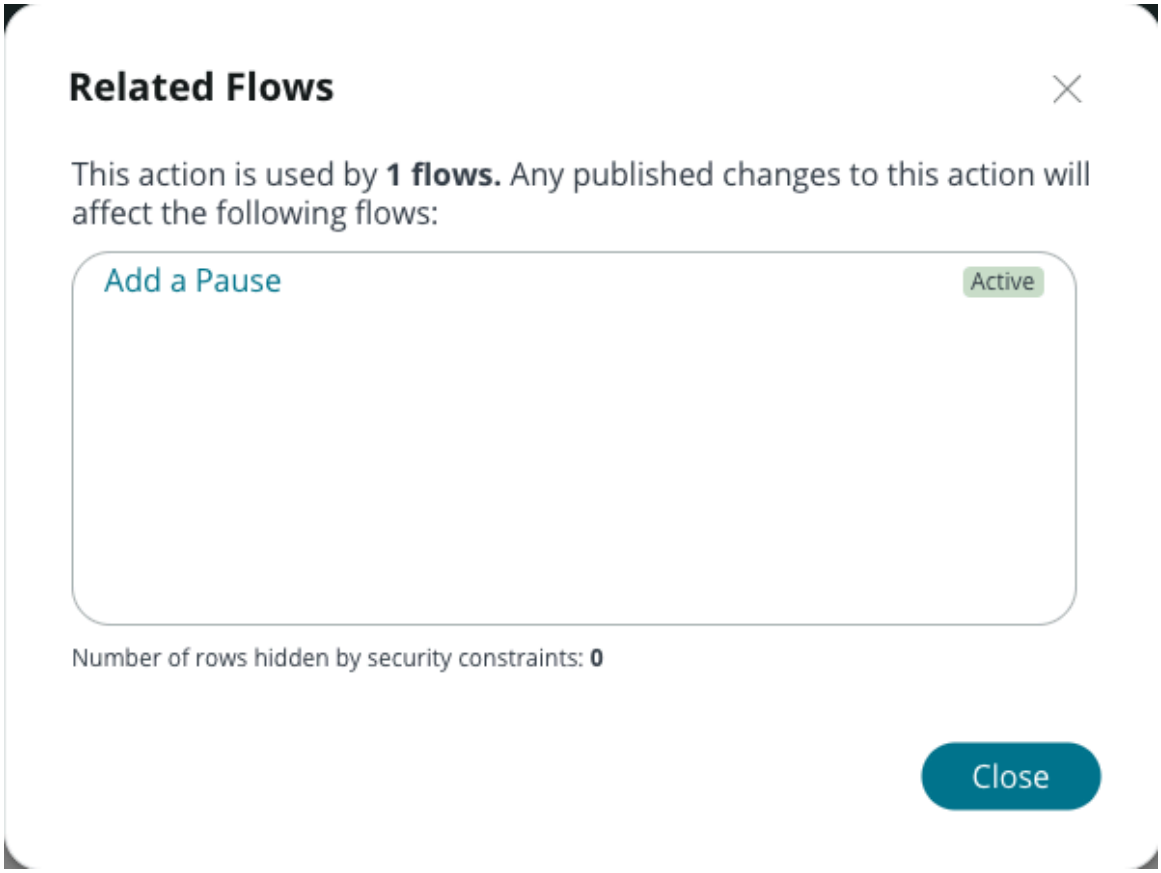
See the list of flows that include a custom action. Determine the impact that changes to an action have on published and draft flows.

Before you begin
Role required: admin or flow_designer

About this task
Determine the impact that your action changes have on published and draft flows. See a list of flows that include the action and determine if your changes require changes to the related flows. For example, if you change the inputs an action uses, someone must reconfigure the related flows to use the new or modified inputs.

Procedure
1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select **Actions**.
3. Select the action whose related flows that you want to see.
   The system displays the action in the Workflow Studio environment.
4. From the More Actions menu, select **See related flows**.

   The system shows the Related Flows dialog.



The Related Flows dialog shows information about the flows that use the action.

- The number of flows using the action
- The name of each related flow
- The activation status of each related flow
- The number of flows using the action that are hidden by security constraints

What to do next
Determine if your planned action changes require updates to the related flows. Publishing an action also displays a list of related flows that use the action.

User access to Workflow Studio flows

Administrators can grant users access to Workflow Studio flows by assigning delegated development permissions or directly assigning a user role. Administrators can also specify which features and content a user can access based on user roles. Application developers can access Workflow Studio functionality through APIs for flows, subflows, and actions.

Access by user role

Administrators can grant access to Workflow Studio flows by directly assigning users the flow_designer user role, which includes the role to view flow execution details.

Warning:
Directly granting a user the flow_designer role is equivalent to giving the user the admin role, because Workflow Studio can run flows as the System user, which has access to all tables and all database operations.

Administrators can also grant users one or more Workflow Studio roles to enable them to create flows and subflows, view flow execution details, and create actions.

Workflow Studio roles

Role title [name]	Description	Contains Roles
flow_designer	Enables you to launch the Workflow Studio flow design environment to create and edit flows and subflows.	flow_operator
flow_designer_scripting	Enables someone with the flow_designer or action_designer role to set and modify input values by writing inline scripts. For information, see Inline scripts.	none
flow_operator	Enables you to view flow execution details, dashboards, and logs. Administrators can grant this role to users that want to be able to view flow results but not create, change, or test them.	none
flow_report_viewer	Enables you to view reports for Workflow Studio flow tables. For a list of relevant flow reporting tables, see Flow execution details retention.	none
action_designer	Enables you to launch the Workflow Studio action design environment to create and edit actions. **Important:** This role provides access to all actions regardless of their application scope.	none
action_category_creator	Enables someone with the action_designer role to create action categories for actions and subflows.	none
fd_read	Enables you to launch the Workflow Studio flow and action design environments to view the configuration and execution details of flows, subflows, and actions. **Note:** Read only roles are incompatible with roles that provide write access. Avoid granting the same user both a read only and a write access role.	fd_read_flows, fd_read_actions, fd_read_operations
fd_read_flows	Enables you to launch the Workflow Studio flow design environment to view the configuration and execution details of flows and subflows. **Note:** Read only roles are incompatible with roles that provide write access. Avoid granting the same user both a read only and a write access role.	fd_read_operations

Workflow Studio roles (continued)

Role title [name]	Description	Contains Roles
fd_read_actions	Enables you to launch the Workflow Studio action design environment to view the configuration of actions. **Note:** Read only roles are incompatible with roles that provide write access. Avoid granting the same user both a read only and a write access role.	none
fd_read_operations	Enables you to view basic flow and action execution details. When reporting is enabled, you can only see basic execution details such as the runtime state and duration. If the reporting level generates additional details, you can't see them. Administrators can grant this role to users that only need to view basic execution results but not create, change, or test flows and actions. **Note:** Read only roles are incompatible with roles that provide write access. Avoid granting the same user both a read only and a write access role.	none
fd_read_operations_all	Enables you to view all generated flow and action execution details. When reporting is enabled, you can view all available execution details. You can only see as much detail as defined by the reporting level system property. Administrators can grant this role to users that need to view all flow results but not create, change, or test flows and actions. **Note:** Read only roles are incompatible with roles that provide write access. Avoid granting the same user both a read only and a write access role.	fd_read_operations

Note:
Some applications provide UI actions to view related flow or flow contexts. You need an application-specific user role to view such UI actions. For example, users require the *itil* or equivalent user role to view the **Flow Context** UI action available from Requested Item records.

API access

Application developers can access Workflow Studio functionality through APIs for flows, subflows, and actions. Flow authors can enable individual flows, subflows, and actions to be client callable during design. For more information, see API access to Workflow Studio flows.

Delegated development access

Administrators can grant users access to Workflow Studio flows by creating an application and assigning users as developers with the Workflow Studio delegated development permission. Delegated development allows administrators to control whether flow designers can access features normally restricted to admin users such as assigning user roles, creating access controls, or creating scripts. For more information, see Developer permissions.

Role-based content filtering

Specify the user roles necessary to access Workflow Studio flow content. For example, flows, flow triggers, actions, and subflows. Manage content filtering by creating content definitions and content filtering rules. For more information, see Content filtering for Workflow Studio flows.

Note:
Your users must have the flow_designer role to create and edit flows. You can specify the additional roles that a user must have to access particular features or content.

Role-based feature access

Specify additional user roles necessary to access the UI elements of Workflow Studio flows. For example, specify a role to access the buttons to save, test, or activate a flow or to access the option to copy a code snippet. Manage feature access directly through the Feature Access List. For more information, see Manage access to Workflow Studio flow features.

Note:
Your users must have the flow_designer role to create and edit flows. You can specify the additional roles that a user must have to access particular features or content.

Manage access to Workflow Studio flow features

Restrict access to individual Workflow Studio flow features by user role. Specify what additional roles a user must have to access an individual feature such as copy a flow.

Before you begin

Create any custom roles that you want to use for feature access. When you specify which roles are required to access a feature, you can select the roles from a list of existing roles.

Role required: admin

About this task

Features are elements of the Workflow Studio UI. When feature filtering is turned off, users with the flow_designer role have access to all of the Workflow Studio features. When feature filtering is turned on, you can specify which roles a user must have to access individual features.

Note:
Your users must have the flow_designer role to create and edit flows. You can specify the additional roles that a user must have to access particular features or content.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Administration** > **Feature Access List**.
2. For each feature on the list, click the edit icon () next to the feature.
3. Enter the role or roles that are required to access the feature.

> **Note:**
> Add the flow_designer role to each feature on the Feature Access List. Your users must have the flow_designer role to access the Workflow Studio features.

You can use these options to restrict access to Workflow Studio action authoring features.

Action Authoring Features

Feature	Description
Copy action	Copy an action to give it a new name and move it to another application scope. For more information, see Copy an action.
Error Evaluation	Enable actions to catch step failures and continue running. Identify when specific error conditions occur and return your own action status code, status message, and error state. For more information, see Error Evaluation.
Manage natural language title	Change the default title for an action by adding styled and dynamic text. For more information, see Manage natural language title.
Manage security	Manage access to Workflow Studio actions.
Executions	View the runtime information about an action from the design environment such as the current state, actions, or steps run, and values produced. For more information, see Executions.
Save	Save an action.
Publish	Publish an action to allow other users to use the action.
Configurations	Manage Workflow Studio configurations.
"If step fails..." Step Option	Continue running the next step or go to an error evaluation. For more information, see "If step fails..." Step Option.
Code snippet	Generate a code snippet to call an action. For more information, see Code snippet.
Test	Test an action before publishing it for other users. For more information, see Test an action.
Properties	Configure how the system processes flows. For more information, see Properties.

Feature	Description
Configure connections	Configure a connection through the Connections dashboard. For more information, see Configure a connection.

You can use these options to restrict access to Workflow Studio flow authoring features.

Flow Authoring features

Feature	Description
Manage natural language title	Change the default title for a flow by adding styled and dynamic text. For more information, see Manage natural language title.
Manage flow catalog variables	Create Service Catalog variables that are only available to a specific Service Catalog-triggered-flow. Flow-specific variables are available to the catalog tasks and actions in the flow. For more information, see Manage flow catalog variables.
Properties	Configure how the system processes flows. For more information, see Properties.
Copy flow	Copy a flow to give it a new name and move it to another application scope. For more information, see Copy a flow.
Test	Test a flow before publishing it for other users. For more information, see Test a flow.
Configure connections	Configure a connection through the Connections dashboard. For more information, see Configure a connection.
Activate	Activate a flow to make it available to other users. For more information, see Activate.
Deactivate	Deactivate a flow to make it unavailable to other users.
Configurations	Manage Workflow Studio configurations.
Code snippet	Generate a code snippet to call a flow. For more information, see Code snippet.
Executions	View runtime information about a flow from the design environment such as the current state, actions, or steps run, and data pill values produced. For more information, see Executions.
Flow Error Handler	Enable flows to catch errors. Run a sequence of actions and subflows to identify and correct issues. For more information, see Flow Error Handler.
Manage security	Manage access to Workflow Studio actions.

Feature	Description
Stages	Configure when stages display to a user, define stage state labels, and add stages to a flow within Workflow Studio. For more information, see Stages.
Diagramming View	Create and view flows as diagrams. You can see the paths a flow can follow and the connections between elements. For more information, see Diagramming View.

You can use these options to restrict access to Workflow Studio subflow authoring features.

Subflow Authoring features

Feature	Description
Manage flow catalog variables	Create Service Catalog variables that are only available to a specific Service Catalog-triggered-flow. Flow-specific variables are available to the catalog tasks and actions in the flow. For more information, see Manage flow catalog variables.
Code snippet	Generate a code snippet to call a subflow. For more information, see Code snippet.
Save	Save a subflow.
Manage security	Manage the access to subflows.
Properties	Configure how the system processes subflows. For more information, see Properties.
Manage natural language title	Change the default title for a subflow by adding styled and dynamic text. For more information, see Manage natural language title.
Configure connections	Configure a connection through the Connections dashboard. For more information, see Configure a connection.
Copy subflow	Copy a subflow to give it a new name and move it to another application scope. For more information, see Copy a subflow.
Test	Test a subflow before publishing it for other users. For more information, see Test a subflow.
Publish	Publish a subflow to allow other users to use the subflow. For more information, see Test.
Configurations	Manage Workflow Studio configurations. For more information, see Configure a connection.

You can use these options to restrict access to Workflow Studio flow template authoring features.

Flow Template Authoring features

Feature	Description
Save	Save a flow template.
Properties	Configure how the system processes flow templates. For more information, see Properties.
Deactivate	Deactivate a flow template to make it unavailable to other users.
Manage flow catalog variables	Create Service Catalog variables that are only available to a specific Service Catalog-triggered-flow. For more information, see Manage flow catalog variables.
Activate	Activate a flow template.

4. Continue adding roles to each feature on the list.
5. Click **OK**.
6. To turn on the feature access for your users, select the **Enable feature access filtering?** option. After the feature access is enabled, your users must have the required roles before they can access the features. If a user doesn't have the required roles for a feature, the feature does not work for that user.

What to do next
Assign your users with the roles that they need to access your features.

Content filtering for Workflow Studio flows

Specify which content a user can access based on the user's role.

Display only content that is relevant for a particular user, hiding content that is unnecessary or sensitive. Specify the Workflow Studio flow content that you want to control access to and the role that a user must have to access it. For example, if a user with the hr_manager role in human resources is creating a flow, show only the set of actions and subflows that are relevant to HR cases.

Content filtering uses:

- Content definitions to specify types of content.
- Content filtering rules to determine who can access the content.

Workflow Studio includes several default definitions and filtering rules. Set up content filtering by modifying pre-existing rules or creating your own.

Content definitions

Content definitions specify a type of Workflow Studio flow resource. Resources are the key components of Workflow Studio flows, such as triggers, actions, and subflows. Create definitions to include an entire resource, or refine your definitions through conditions. For example, you can create a definition that includes all flow triggers, or you can use conditions to include only triggers with a category of date.

You can further refine content definitions through tagging. Add tags to items in a resource list, then design your content definition to only include resources with that tag.

Content filtering rules

Content filtering rules specify the role that a user must have to access the content in a particular definition. Each rule associates a single user role with a single content definition. When a user accesses Workflow Studio flows, content filtering rules determine what content the user may access based on the user's role.

Feature access

You can also filter access to Workflow Studio flow features. Features are UI elements and sections. Access to both elements and sections can be managed by configuring content definitions and filtering rules. However, access to UI elements can also be managed through a simplified UI. For more information, see Manage access to Workflow Studio flow features.

Read-only flows

Users may be able to view a flow, subflow, or action containing content that they can't normally access. For example, a flow that's visible to a user might include an action the user wouldn't usually be able to view. When a flow contains restricted content, the entire flow becomes read-only. Users can run the flow but can't modify or copy it.

The creation of read-only flows doesn't apply to feature filtering. If a user doesn't have access to a feature, the feature doesn't render for that user. It doesn't affect the ability to copy or modify a flow. If a user doesn't have access to transform functions and uses a flow that already has a transform function applied, the transform function is read-only. The rest of the flow can still be copied and modified.

Access summary

Resource filtered	User has role	User does not have role
Flow	• The flow is visible to select during design. • The flow can be copied. • The flow can be modified.	• The flow is hidden and cannot be selected during design. For example, the flow is hidden when creating a Playbooks activity definition. • The flow cannot be copied. • The flow is read-only.
Flow execution details	The flow execution details are visible.	The flow execution details are hidden.
Trigger	• The trigger is visible to select during design. • Any flow that includes the trigger can be copied. • Any flow that includes the trigger can be modified.	• The trigger is hidden and cannot be selected during design. • Any flow that includes the trigger cannot be copied. • Any flow that includes the trigger is read-only.

Resource filtered	User has role	User does not have role
Subflow	• The subflow is visible to select during design. • Any flow that calls the subflow can be copied. • Any flow that calls the subflow can be modified.	• The subflow is hidden and cannot be selected during design. • Any flow that calls the subflow cannot be copied. • Any flow that calls the subflow is read-only.
Subflow execution details	The subflow execution details are visible.	The subflow execution details are hidden.
Flow logic	• The flow logic is visible to select during design. • Any flow that includes the flow logic can be copied. • Any flow that includes the flow logic can be modified.	• The flow logic is hidden and cannot be selected during design. • Any flow that includes the flow logic cannot be copied. • Any flow that includes the flow logic is read-only.
Action	• The action is visible to select during design. • Any flow that includes the action can be copied. • Any flow that includes the action can be modified.	• The action is hidden and cannot be selected during design. • Any flow that includes the action cannot be copied. • Any flow that includes the action is read-only.
Action execution details	The action execution details are visible.	The action execution details are hidden.
Step	• The step is visible to select during design. • Any action that includes the action can be copied. • Any action that includes the step can be modified.	• The step is hidden and cannot be selected during design. • Any action that includes the step cannot be copied. • Any action that includes the step is read-only.
UI elements and sections, excluding transform functions	• The UI element or section is visible to use during design. • Any flow, subflow, or action that includes the UI element or section can be copied.	• The UI element or section is hidden and cannot be used during design. • Any flow, subflow, or action that includes the UI element or section can be copied. • Any flow, subflow, or action that includes the UI element or section can be modified.

Resource filtered	User has role	User does not have role
	• Any flow, subflow, or action that includes the UI element or section can be modified.	
Transform functions	• Transform functions are visible to use during design. • Any flow, subflow, or action that includes a transform function can be copied. • Any flow, subflow, or action that includes a transform function can be modified.	• Transform functions are hidden and cannot be used during design. • Any flow, subflow, or action that includes a transform function can be copied. The transform function is read-only. • Any flow, subflow, or action that includes a transform function can be modified. The transform function is read-only.

Configure content filtering definitions

Specify which content a user can access by creating content definitions.

Before you begin

Content filtering requires some familiarity with user roles and Workflow Studio tables and records.

Role required: flow_designer, action_designer, or admin

About this task

Filter Workflow Studio flow content based on user role. Filtering content requires you to set up:

1. Content definitions to describe the content that you want to filter. Content definitions specify types of Workflow Studio flow resources, such as actions and subflows.
2. Content filtering rules to state the role a user must have to access the resource in a particular definition.

Workflow Studio includes several content definitions and filtering rules by default. Get started with content filtering by modifying the preexisting definitions and rules or creating your own.

Procedure

1. To modify or create a content definition, navigate to **Process Automation** > **Flow Administration** > **Content Definitions**.
2. Select the definition that you want to modify or click **New** to create one.
3. On the form, fill in the fields.

Workflow Studio Resource form

Field	Description
Name	Name for the content definition.
Application	Application scope to which the content definition applies. This field is automatically set to the currently selected application scope. If no application

Field	Description
	scope is selected, the field is set to Global. If you set a specific application scope, then the content definition only applies to that application scope. If you select the Global application scope, then the content definition applies to all applications.
Table	Table containing the resource type that you're defining. For example, the Flow [sys_hub_flow] table includes all the flows and subflows available on your instance.
Conditions	Conditions used to filter the records in the table. For example, creating a condition where **[Flow Type] [is] [SubFlow]** returns only the subflows from the Flow table.
Resource Tags	Tags used to filter the resources in the table.

4. Click **Submit**.

Configure content filtering rules

Use content filtering rules to specify the role a user must have to access content.

Before you begin

Role required: flow_designer, action_designer, or admin

Content filtering requires some familiarity with user roles and Workflow Studio tables and records.

Role required: flow_designer, action_designer, or admin

About this task

Filter Workflow Studio flow content based on user role. Filtering content requires you to set up:

1. Content definitions to describe the content that you want to filter. Content definitions specify types of Workflow Studio flow resources, such as actions and subflows.
2. Content filtering rules to state the role a user must have to access the resource in a particular definition.

Workflow Studio includes several content definitions and filtering rules by default. Get started with content filtering by modifying the pre-existing definitions and rules or creating your own.

Procedure

1. To modify or create a content filtering rule, navigate to **Process Automation** > **Flow Administration** > **Content Filtering Rules**.
2. Select the rule that you want to modify or click **New** to create one.
3. On the form, fill in the fields.

Workflow Studio Resource Filter Rule form

Field	Description
Name	Name for the content filtering rule.
User Role	The role a user must have to access the content in the **Resource Definition** field.
Active	Option to enable the rule.

Field	Description
Application	Application scope to which the content filtering rule applies. This field is automatically set to the currently selected application scope. If no application scope is selected, the field is set to Global. If you set a specific application scope, then the content filtering rule only applies to that application scope. If you select the Global application scope, then the content filtering rule applies to all applications.
Resource Definition	The name of the content definition that specifies the resource to filter.

4. Click **Submit**.

API access to Workflow Studio flows

Application developers can access Workflow Studio functionality through APIs for flows, subflows, and actions. Flow authors can enable individual flows, subflows, and actions to be client callable during design.

Available Workflow Studio flow APIs

Trigger flows, subflows, and actions using these APIs from server or client scripts.

Server side

FlowAPI : Trigger a flow, subflow, or action using synchronous or asynchronous methods, with or without execution details.

Client side

GlideFlow : Perform client-side interactions with actions, flows, and subflows. Flow designers must enable a flow, subflow, and action to be called from the client.

FlowAPI quick methods

Use quick methods in the FlowAPI class to run an action, flow, or subflow from a server-side script without creating execution details or other related records. Use these methods to increase the speed of high-volume processing in a production environment, and to improve performance by eliminating record-keeping overhead. Methods include:

- `executeActionQuick()`, `executeFlowQuick()`, `executeSubflowQuick()`: Run an action, flow, or subflow from a server-side script synchronously from the current user session.
- `startActionQuick()`, `startFlowQuick()`, `startSubflowQuick()`: Run an action, flow, or subflow from a server-side script asynchronously.

XML and JSON streaming APIs

Builds a large streaming or non-streaming JSON or XML payload to use in a REST or SOAP request to send bulk data to a third-party API. For example, you can use these APIs to create a JSON payload in the Workflow Studio Script step and pass the returned value to the REST step to send the request to a third-party service. For more information, see JSONStreamingBuilder and XMLStreamingBuilder .

Client callable APIs

By default, the flows, subflows, and actions can only be called by the FlowAPI within a server script. Flow and action designers can make individual flows, subflows, or actions available to client calls by enabling the **Client callable** option during the design process.

Run as support

Flows and subflows can run as either the system user or the user who initiates the session. Set this behavior from the flow properties. All API quick methods ignore the run as property, and always run as the system user.

Actions always run as the user who initiates the session.

Code snippets

Application developers can generate a JavaScript function that calls a specific flow, subflow, or action with the **Code Snippet** option. Use the code snippet in scripts such as business rules or the **Scripts - Backgound** module to call specific Workflow Studio elements. The system only generates code snippets for published flows, subflows, and actions. Workflow Studio elements in the draft or modified status do not generate code snippets.

Create code snippets for flows, subflows, and actions

Generate a code snippet to call a specific flow, subflow or action.

Before you begin

- Activate the flow or subflow you want to call.
- Publish the action you want to call.

Role required: flow_designer or admin

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Open a flow, subflow, or action.
3. Click the **More Actions** icon in the upper right corner of the flow designer.
4. Select **Create Code Snippet**.
   The code snippet appears in a new window and is selected by default.
5. Copy the code by manually copying it from the pop-up modal or clicking the **Copy Code Snippet Clipboard** button.

What to do next

Edit the script to add any mandatory values, such as inputs.

Create a client callable flow, subflow, or action

Enable a client script to trigger a flow, subflow, or action.

Before you begin

- Role required: security_admin
- Consider the implications of making a flow, subflow, or action client callable, such as whether it exposes protected data or bypasses validation logic.

About this task

By default, the flows, subflows, and actions can only be called by the FlowAPI within a server script. Flow and action designers can make individual flows, subflows, or actions available to client calls by enabling the **Client callable** option during the design process.

Procedure

1. Elevate privileges to security_admin.
2. Navigate to **System Security** > **Access Control (ACL)**.
3. Click **New**.
4. Create an access control.

Field	Description
Type	client_callable_flow_object
Operation	execute
Admin overrides	Selected
Name	Enter a name for the ACL.
Requires role	Create a role to provide access to the APIs. For example, create a flow_api_access role.

5. Assign the role to the user you would like to grant access to.
6. Enable a client script to trigger the flow, subflow, or action.
   a. Open the flow, subflow, or action you want to make client callable.
   b. In the **More Actions** menu, select **Manage security**.
   c. Select **Callable by Client API**.
   d. Add the access control record created earlier to the **ACLs** field.
   e. Click **Update**.

Result

The user with the designated permissions can trigger a client callable flow, subflow, or action from a client script using the GlideFlow API.

Restricted caller access to Workflow Studio flows

Track flows and actions that require access to cross-scope resources. Allow or deny flows and actions access to cross-scope resources.

The Restricted Caller Access Privileges table has dedicated source types to identify Workflow Studio calling sources.

Flow

The system uses the flow source type to track operations run by ServiceNow Core actions. Restricted Caller Access Privileges records allow a flow to perform a specific operation on a specific cross-scope resource. Approving a flow to run an operation allows any other core action within the same flow to perform the same operation on the same cross-scope resource.

For example, suppose you have a flow that runs the Look Up Records action on a cross-scope table. When caller restriction is enabled for the cross-scope table, the

Look Up Records action generates a request to perform a read operation. When you allow the flow to perform read-operations on the cross-scope table any other read operations performed by core actions can also run. For example, your flow could run the Look Up Record and Lookup Attachments actions on the same cross-scope table. Suppose you add the Look Up Records action for the same cross-scope table to another flow or subflow. Since this read operation comes from another flow, the core action generates a separate access privilege request for approval. If you configure the Look Up Records action to access another cross-scope table, that too generates a separate access privilege request for approval.

Flow Action

The system uses the flow action source type to track operations run by custom actions to a specific cross-scope resource. Restricted Caller Access Privileges records allow a custom action to perform a specific operation on a specific cross-scope resource. Approving an action to run an operation allows the custom action to perform the operation on the cross-scope resource in any context.

For example, suppose you create a custom action that runs the Look Up Records step on a cross-scope table. When caller restriction is enabled for the cross-scope table, the Look Up Records step generates a request to perform a read operation. When you allow the custom action to perform read operations on the cross-scope table you can run the custom action from any context. For example, you can add the custom action to multiple flows or call the custom action from a script. As long as the custom action performs the operation on the allowed target cross-scope resource, the system allows the custom action to run. If you configure the custom action to access another cross-scope table, the custom action generates a separate access privilege request for approval.

Upgrade restricted caller access privileges for flows and actions

Allow instances upgraded from San Diego and earlier releases to generate restricted caller access privilege requests for flows and actions.

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

In San Diego and earlier releases, the Restricted Caller Access Privileges table did not recognize flows and actions as source types. Customers who wanted to generate Restricted Caller Access Privileges records for flows and actions could only do so indirectly. They could use a script include or business rule to call a flow or action. When the script include or business rule ran it would generate an access privilege request to the cross-scope resource for approval.

Warning:
Upgrading restricted caller access privileges to track flows and actions can cause service disruptions on instances that previously tracked cross-scope access from script includes or business rules. After upgrade, all flows and actions that attempt to access restricted resources will be blocked from running and instead generate their own restricted caller access privilege requests for approval. Someone must approve the access privilege requests before cross-scope flows and actions can run. Customers who already allowed indirect tracking of flows and actions using script calls may want to skip this task and continue calling flows and actions from scripts. Customers who want to replace their existing access privileges with the new Flow and Flow Action source types may want to schedule an outage to generate and approve the new access privilege requests.

Procedure

1. Add a system property .
   Create this property.

Field	Value
Name	`com.glide.hub.flow.restricted_caller_access.track_flows_as_source`
Type	**true|false**
Value	true

2. Define cross-scope access to an application resource .
   Enable Caller Restriction for the tables you want flows and actions to request access to.
3. If you are replacing existing script-based access-permissions, identify the existing cross-scope flows and actions that need access permissions.
   You must regenerate any existing access privileges for cross-scope flows and actions. The flow and action access privileges replace the existing script include and business rule access privileges.
4. Generate access privilege requests for any existing cross-scope flows and actions.
   You can run cross-scope flows and actions by using or testing the cross-scope application. Cross-scope flows and actions generate access privilege requests to the tables set to caller restriction.
5. Allow flows and actions to access your application resources .
   Identify access privilege requests with these source types.
   - Flow
   - Flow Action

   Set the Status to **Allowed** for each operation you want a flow or action to perform on your restricted application resource.

Result
Flows and actions that attempt to access your restricted application resources generate an access privilege request.

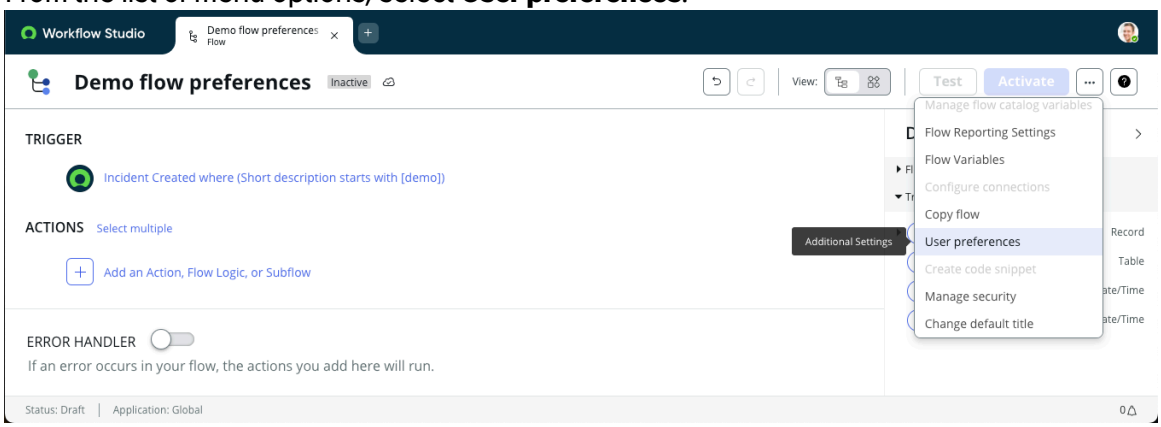
What to do next
Review and approve access privilege requests from your application record.

Set flow user preferences

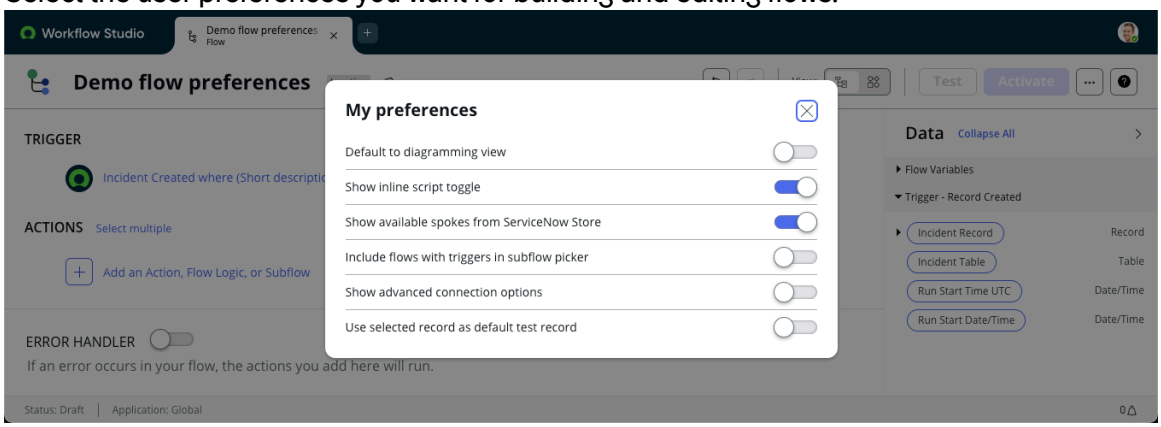
Set your preferences when building and editing flows such as the default editor view.

Before you begin
Role required: admin or flow_designer

Procedure
1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Flows**.
3. Create or select the flow you want to edit.
4. Select the **More Actions menu** icon.
5. From the list of menu options, select **User preferences**.



6. Select the user preferences you want for building and editing flows.



For more information about available user preferences, see User preferences for flows.

Update to the latest version of Flow Diagramming

Flow diagramming is automatically installed on your instance as a dependency of Workflow Studio.

Before you begin
Role required: admin

About this task
As of Washington DC Patch 3, updating Workflow Studio automatically updates all of its application dependencies such as Flow Designer, Playbook, and Decision Builder. You can no longer see or update the individual application dependencies of Workflow Studio from the ServiceNow Store or the list of plugins.

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. Find the Workflow Studio application using the filter criteria and search bar.
3. Using the version list on the right, select the latest version of Workflow Studio.
4. Select **Update**.

Result

The Flow Diagramming application is updated to the latest application version. See the Flow Diagramming application listing on the ServiceNow Store for the latest available version and features.

Configuring decision tables

Workflow Studio is installed automatically with each family upgrade. By updating to the latest version of Workflow Studio, you have the latest updates to decision tables. Configure Workflow Studio and assign roles to your staff based on the permission level they require.

For more information, see Update to the latest version of Workflow Studio.

Decision tables personas and roles

The responsibilities of your staff are controlled by roles assigned to each member. Personas aren't explicitly part of decision tables in Workflow Studio, but administrators assign roles to give team members permission to configure or use decision tables.

Personas and roles

Review the personas and roles used in decision tables in Workflow Studio.

Decision table admin

The decision table admin can create and edit any part of a decision table. The admin may make high-level decisions about when a decision table is needed, create the table, and then pass on the work of authoring the rules in the table to someone else. Decision table admins are the only persona who can create inputs and decision table columns and values.

Decision table reader

The decision table reader has read-only access to decision tables. They can't add or delete inputs, conditions, or results, but they can export a decision table to Excel.

Decision rule author

The decision rule author has the most access to update decision tables after the decision table admin. The admin may create a table with inputs and table columns and then pass on the rest of the work to the decision rule author. The decision rule author can add, rearrange, and delete decision rows and edit condition and result values in all rows and columns.

Decision result editor

The decision result editor can only edit result values in existing decision rows. The decision result editor may change a value based on a new business decision or policy.

Decision table user roles and permissions

Role	Tasks
admin and decision_table_admin	Create, read, edit, and delete inputs and decision table columns and values.
decision_table_reader	Read-only access to inputs and decision table columns and values. In decision tables, users with this role can export decision tables to Excel.
decision_rule_author	Read-only access to inputs and decision table columns. In decision tables, users with this role can do the following: • Add new decision rows • Rearrange decision rows • Delete decision rows • Edit condition and result values in all rows and columns
decision_result_editor	Read-only access to inputs, decision table columns, and conditions. In decision tables, users with this role can edit result values in existing decision rows.

Update to the latest version of Workflow Studio

Workflow Studio is automatically installed on your instance. However, Workflow Studio is a ServiceNow Store application, so to get the latest features, you must update your version manually to the most recent version.

Before you begin

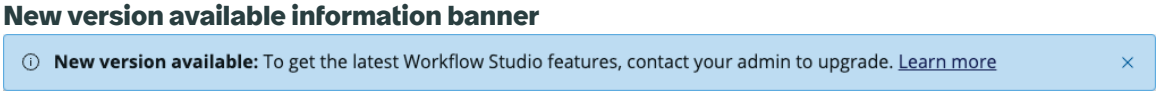
Role required: admin

About this task

As of Washington DC Patch 3, updating Workflow Studio automatically updates all of its application dependencies such as Flow Designer, Playbook, and Decision Builder.

As of Xanadu Patch 3, you can also find and update each individual Workflow Studio application dependency from the ServiceNow® Store or the list of plugins.

New version available information banner



Beginning with the Xanadu release, Workflow Studio displays an information banner when a new version is available.

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. Search for the Workflow Studio application.
   Check the **Updates** and **Installed** tabs to locate Workflow Studio (app id: sn_workflow_studio).
3. Select **Workflow Studio**.
4. Using the version list on the right, select the latest version of Workflow Studio.
5. Select **Update**.

Using Workflow Studio

Create a Workflow Studio component to automate a workflow.

Workflow Studio allows you to create and manage workflow components.

Building flows

Flows automate a repeatable multi-step process. When the flow trigger conditions are met, the flow runs a sequence of reusable actions and flow logic to complete the process.

Building subflows

Subflows automate a repeatable multi-step process that also produces an output needed by another process. When a playbook, flow, or script calls a subflow, the subflow runs a sequence of reusable actions and flow logic to complete the process and produce output values.

Building actions

Actions automate a repeatable task or operation within a flow. Flows run actions by passing them data as inputs. Actions run a sequence of steps to complete the task, and pass data to the flow as outputs.

Building flows

Workflow Studio is the default ServiceNow AI Platform process automation builder used to create flows. Workflow Studio replaces the Workflow Editor.

Flows consist of a trigger and a sequence of actions and flow logic. When you add actions and flow logic to a flow, the data collected or created appears in the data panel, which you can use in other actions and flow logic.

Building and managing flows requires that you have some familiarity with the ServiceNow AI Platform tables and fields that the application or process uses. Process analysts can create flows using available actions and flow logic, or they can copy an existing flow to use it as a template.

Action limit

By default, flows can have no more than 50 actions. To change the default behavior, increase the value of the *sn_flow_designer.max_actions* system property. However, consider the performance impact that a large flow may have on your instance.

Missing actions

If an administrator added your flow from an update set, you might have some missing actions in your flow. This normally happens when your instance doesn't have the appropriate Integration Hub spokes installed. For more information on how to install the spokes you need to get these actions to appear, see spokes.

More Actions menu

Select the **More Actions** icon ( ) to access additional options for your flow.

Stages

Access the stages for a flow.

Manage flow catalog variables

Manage the catalog variables available to Service Catalog-triggered-flows.

Flow Variables

Create flow variables to set and retrieve data throughout a flow.

Copy flow

Create a copy of the open flow in an application you specify.

Flow preferences

Enable or disable the **Show draft actions**, **Show triggered flows**, **Show store spokes**, and **Show inline script toggle** options.

Create code snippet

Generate a code snippet to call a specific flow, subflow, or action.

Manage security

Enable or disable the **Callable by Client API** option.

Change default flow title

Change the default title for your flow by adding styled or dynamic text. For more information, see Create a natural language title.

Testing flows

You can test a flow directly from the Workflow Studio interface. Each test runs your flow as if the trigger conditions were met. If the flow has record trigger, you can specify the record to use for your test. After the flow runs, use the flow execution details to verify that your flow is running properly.

Important:
Always run tests on a non-production instance where flow record changes cannot interfere with your production data.

Flow execution details

Each time you test a flow, the system generates flow execution records, log messages, and reports. The flow context is a related record containing the current state and runtime values of the flow. The system generates a context record each time a flow is run.

Optionally, you can configure the system to generate execution details anytime a flow is run, not just during testing. For more information, see Activate flow reporting.

Flow properties

The flow properties contain information about your flow. In the main header, select **Properties** to view or edit your flow's properties.

Property	Description
Name	Enter a unique name for the flow
Protection	Choose whether the flow is read only by choosing from **None** or **Read-only**
Application	Select an application for the flow. This property is set when creating the flow and cannot be changed afterwards.
Description	Enter a description of the flow.
Run As	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 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.
Run with role(s)	Roles that the flow runs with. This option is only available when **Run as** is set to **user who initiates the session**.

Printing flows

Workflow Studio supports multipage printing for flows, subflows, actions, and flow execution details. For a list of supported browsers, see Browser support.

Roles

To access Flows, a user must have the flow_designer or admin roles.

General guidelines

Flows should be short, modular, reusable collections of work. If they take more than an hour to execute, they're probably too long and can be more efficient.

Any general guidelines that apply to flows also apply to subflows.

Prevent conflicting or duplicate business logic

Automations can be created with Flow Designer, business rules, workflows, and Integration Hub. Before you start using Workflow Studio, make sure you understand how existing ServiceNow AI Platform automations work. Deactivate automations

before replacing them with Workflow Studio flows and actions. See the Architecture Overview to learn how Workflow Studio works within the ServiceNow AI Platform.

Review Flows, Sub-flows, and Actions documentation, if necessary.

Determine whether your flow needs a trigger or variable input

Flows always run when their trigger conditions are met. Triggers always provide the same data as input for flows. If you need variable input to initiate a flow instead, create a subflow.

Reuse business logic

Create a set of reusable operations as a subflow that can then be used in multiple flows.

Grant flow roles to access role-protected data and preserve user information

Flow roles help keep permissions for your flows simple. Use flow roles to preserve user information and grant access to data, instead of running a flow as the system user. Adding flow roles also gives access to additional data that a user-initiated flow does not usually have. The roles granted only apply to the flow. They do not apply to the user who initiated the flow.

Use flow logic or a schedule-based trigger to control flow timing

Flow logic or schedule-based triggers help to optimize the performance of your flows. Do not use the `gs.sleep()` method to wait within a flow. The `gs.sleep()` method prevents the thread from performing other work. To run a flow at a specific time, use a schedule-based trigger. To pause a flow for a specific duration, use the Wait for a duration or wait for condition flow logic.

Avoid dependencies

Parallel branches that depend on each other stall a flow when a branch has to wait for output from another branch.

Scope loop counters

Script loops don't have a maximum number of iterations, so loops execute infinitely when there is not a valid exit condition.

To make sure that there is a valid exit condition, use scope loop counters in inline scripts or in script steps within an action. Add `var` to`for (i=0; i< length; i++)` and get `for (var i=0; i< length; i++)`

Limit For Each and Do Until loops to 1000 iterations

Iterations with 1000 or more loops can lead to memory issues.

- Set max records on Look Up Records to 1000.
- Avoid changing property sn_flow_designer.max_iterations, which defaults to 1000.
- For large amounts of data processing, consider batching into smaller batches.
- For bulk imports, consider concurrent imports.

Use QuickAPI for faster executions (business rule alternative)

- QuickAPI executions are much faster, but there is less debugging capability.
- Foreground QuickAPI executions run in the user session as the user who called the flow.
- Background QuickAPI executions run in a background thread and are run in the 'system' user session.

Use Do Until loops instead of calling flows from themselves

A flow calling itself is not allowed and errors out. But if flow A is calling flow B, then flow B can call flow A up to three times.

Execute flows in the background

Executing flows in the background enables UI threads to be released rather than keeping users waiting for flow executions.

Avoid flow logic that waits after collecting a large output

Using a large payload immediately after it is retrieved can help prevent memory issues.

Minimize switching between environments

Constantly switching back and forth between instance and MID Server steps in a flow can lead to delays in processing. To minimize the risk of delays, limit switching between instance and MID to only one time.

Include sys_complex_object records generated by the flow in update sets

Missing complex data schemas can cause execution issues. Make sure you include sys_complex_object records generated by the flow in update sets.

Calling flows from a script

Start flows with a custom trigger by calling from a script.

Avoid deploying newer release flows to instances on older releases

Workflow Studio does not support deploying newer release flows to instances running on earlier releases.

> **Danger:**
> The flow data model can change between releases, which can prevent newer flows from running or produce unexpected results when running on earlier release instances. Upgrade your instances to be on the same release versions before deploying them.

Turn flow reporting off in production

Minimize the amount of memory required to run flows by disabling Flow reporting. Flow reporting stores configuration and runtime information for the Execution Details page. These reports are good for troubleshooting, but requires a large amount of data to be retained both in memory and in the database. By default, flow reporting is disabled, and the system only generates execution details when you manually test a flow or action. Instead you can use log files, which are still available when reporting is turned off.

Reduce the amount of memory consumed in flows with nested looping

When reporting is activated, set com.snc.process_flow.reporting.iteration.lastn to a value of "1" to reduce the amounts of the amounts of memory that previous loop iterations consume. The more iterations you report on, the more memory is required.

Create a flow in Workflow Studio

Run a sequence of actions and flow logic when the trigger conditions occur.

https://player.vimeo.com/video/984477123?h=6189b3336b&badge=0&autopause=0&player_id=0&app_id=58479

Before you begin

- Create or select an application to store your Workflow Studio content.
- Role required: flow_designer or admin

About this task

Users with the flow_designer or admin role should know the application table structure and be aware of any existing business logic associated with the target tables of a flow or subflow. Be sure to disable any conflicting business rules or workflows before creating a flow or subflow.

Creating a custom application to contain your Workflow Studio content enables you to deploy the application using the application repository or the ServiceNow Store.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. From the Workflow Studio home page, select **New** > **Flow**.
3. Select whether to build the flow with Now Assist or to build the flow from scratch.

   To build a flow with Now Assist, see Create a flow with Now Assist.

   To build a flow from scratch, continue to the next step.
4. Select the **Build from scratch** tab.
5. On the Flow Properties screen, fill in the fields.

Field	Description
Flow name	Name to uniquely identify your flow. The system converts the flow name into an internal name by replacing space characters with underscore characters.
Description	Description of your flow.
Application	Application scope to create your flow in. Global is the default. The application scope determines what data your flow can access and what data it can share.
Domain	Domain scope of the flow. For more information about domain separation, see Domain separation explained.

Additional properties

Field	Description
Protection	Selection to specify 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	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 the flow. Settings for the Run as option in a flow don't apply to child subflows. **Note:** By default, flows run as the user who initiates the session. 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.

Field	Description
	When flows run as the user who initiates the session, the system limits flow actions by user ACL restrictions. Ensure that security restrictions don't prevent users who trigger the flow from performing flow actions. Flows run by the initiating user also respect user-specific settings such as 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**.
Flow Priority Default	Priority level at which you want the system to run this flow by default. Options include: ○ Low ○ Medium (Default) ○ High To learn about flow priority levels, see Flow priority.

6. Select **Submit**.

   Note: If this is your first time in Workflow Studio, a welcome screen appears. You can choose to either take the welcome tour or skip the tour for now.

   The system displays the Workflow Studio design environment.

7. 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**.

8. 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**. 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.

9. Click **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.

  Note:
  The system only triggers activated flows.

- Deploy or transfer your flow to another instance.

- Deploy your flow from the application repository when your application is complete and ready for release.
- Transfer your flow from an update set XML file when you want to test the flow on another instance. When you save a flow, Workflow Studio generates a single update set file containing its subflows and actions.

Create a flow with an inbound email trigger

Start a flow when your instance receives an email.

Before you begin

Role required: flow_designer or admin

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select **+ New** > **Flow**.
3. In the flow properties, set **Run as** to **User who initiates session**.

> **Important:**
> Inbound email flows use the email sender as the user who initiates the session. If the system doesn't recognize the sender, the inbound email flow runs as the Guest user. Setting the inbound email flow to run as the user who initiates the session ensures that the flow actions are limited by user access controls. If the initiating user needs elevated privileges for some reason, have the inbound email flow call a subflow that runs with the required roles.

4. In the Trigger section, select the plus icon (⊕) to add a trigger.
5. Select **Inbound Email**.
6. On the form, fill in the fields.

Field	Description
Email conditions	Conditions to qualify which emails start your flow. The condition builder uses fields from the Email [sys_email] table. For example, to trigger a flow for an inbound email in which a user replies to a system notification, set the condition to **[Receive type] [is] [Reply]**.
Reply Record Type	Table that is associated with the target email. For example, to trigger a flow from a reply email that a user sends from an incident record notification, select **Incident [incident]**.

7. Select **Done**.
8. Add actions, subflows, and flow logic to the flow.
9. **Optional:** Add stages to the flow to report progress to the requester.
   See Flow and subflow stages.
10. Test the flow.
    To test access controls for an inbound email flow, impersonate a typical inbound email user and manually trigger the flow.. For more information, see Test a flow.
11. If the test is successful, activate the flow.
    For more information, see Activate a flow.

Result
When an inbound email meets the conditions that you set, the associated flow triggers and runs the actions.

Allow multiple triggers to process an inbound email

Configure Workflow Studio to allow an inbound email to be processed by multiple inbound email triggers in a specific order.

Before you begin
Role required: admin

About this task
Although you can process an inbound email with multiple inbound email actions, you can't process an inbound email with multiple flows by default. You can add a system property to let process owners use multiple triggers to process an inbound email.

Warning:
Allowing multiple triggers to process an inbound email may increase maintenance and decrease system performance.

Procedure

1. Add a property with the following settings:
   - Name: *glide.hub.flow.inbound_email_trigger.show_advanced*
   - Type: true | false
   - Value: true

   After you activate the system property, the **Order** and **Stop processing** fields appear on the inbound email trigger.
2. Create multiple flows with an inbound email trigger.

   For more information, see Create a flow with an inbound email trigger.
3. Specify the processing order for each of the inbound email flows:

   a. Open each flow and enter a value in the **Order** field.

   To give the flow higher priority over other flows, enter a lower number.

   b. Enable or disable stop processing for each flow in your sequence.

   To allow an inbound email to be processed by the next flow in your sequence, clear the **Stop processing** option.

   To end the sequence on a particular flow, leave the option selected in that flow.

Create a flow with a Kafka Message trigger

Build a flow that processes events from a Kafka stream. Start the flow when an event is available in the specified topic.

Before you begin
Role required: flow_designer or admin

This trigger requires a Stream Connect subscription. For more information, see https://www.servicenow.com/products/automation-engine.html.

This trigger requires the ServiceNow Stream Connect Installer [com.glide.hub.stream_connect.installer] plugin.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Click **New** > **Flow**.
3. Define the flow properties.
   For more information, see Create a flow in Workflow Studio.
4. In the Trigger section, click **Add a trigger** and select **Application** > **Kafka Message**.
5. On the Kafka Message form, fill in the fields.

Field	Description
Topic	Reference to the topic to monitor for messages.
Serialization format	The serialization format for the message. Select one of the following. ○ **Plain Text**: Select this option for any plain-text messages. This is the default format. ○ **Encoded**: Select this option for messages in an Apache Avro format. Converting plain-text messages to an Avro format requires a schema. Select the schema registry in the **Schema registry** field. For information on schemas, see Schema management in Stream Connect.
Schema registry	Registry for the selected schema. Select one of the following. ○ **Standalone Schema Registry** ○ **Confluent Schema Registry** This field appears only when the **Serialization format** is set to **Encoded**. For the Confluent Schema Registry, if the received message's schema ID isn't in the schema table, the system imports the schema dynamically, using the configured REST connection.
Advanced Options	
Start processing from	Option to begin processing messages from the beginning or end of the queue, organized by date. Select one of the following. ○ **The earliest message in the queue**: Begin processing from the oldest messages in the queue. ○ **The end of the queue**: Begin processing from the newest messages in the queue.
Number of messages to process per run	Option to specify the number of messages processed per run or let the system decide on the number of messages. Select one of the following. ○ **Automatically optimize**: The system determines the number of messages to process per run. ○ **Manually override (advanced)**: You specify the number of messages to process per run.

Field	Description
Enter the number of messages (ignored if system-generated value is lower)	Number of messages to process each run. This field appears only when **Number of messages to process per run** is set to **Manually override (advanced)**. ○ Type: integer ○ Default value: 100 ○ Minimum value: 1 ○ Maximum value: 100,000 If the selected number is greater than the number calculated by the system, the flow runs with the system-calculated number. Depending on the size of the messages, the number of messages processed per run might be lower than the number specified.

6. Click **Done**.
7. **Optional:** Set the **Max concurrency** and **Relative weight** fields.
   The **Max concurrency** field determines the number of partition groups to create and the number of parallel processors to use. The **Relative weight** field enables you to allocate more processing time to the consumer relative to other consumers.

   a. Navigate to **All** > **Process Automation** > **Flow Administration** > **Settings**.

   b. In the **Flow/SubFlow/Action** field, select the info icon () for the flow, then select **Open Record**.

   c. Select the Additional actions icon () and select **View** > **Kafka**.
   The **Max concurrency** field appears on the form.

   d. To view the **Relative weight** field, select **Advanced**.

   e. Set the values for **Max concurrency** and **Relative weight**.

Field	Description
Max concurrency	(Optional) Maximum number of parallel processors to use to consume messages, and the number of partition groups to create. Default value: 1
Relative weight	(Optional) For each cycle, the maximum amount of time allocated to the consumer for processing messages, relative to other consumers. For example, a consumer with a relative weight that's twice as high as another consumer's gets twice as much time. Consumers that have the same relative weight get the same amount of time. The minimum value is 5. The maximum value is 2000. The default value is equal to the max concurrency multiplied by the value of the *glide.ih.kafka.consumer.max_seconds_per_partition_group*

Field	Description
	property. The *glide.ih.kafka.consumer.max_seconds_per_partition_group* property specifies the maximum time, in seconds, allocated to each partition group. Its default value is 10. The system uses the relative weight to calculate the partition group timeout for the subscription. The partition group timeout specifies the maximum time, in milliseconds, allocated to each partition group in a specific subscription. This field appears only when **Advanced** is selected.

f. Select **Update**.

g. Navigate back to your flow in Workflow Studio.

8. Add actions, subflows, and flow logic to the flow.
9. To test the flow, click the **Test** button.
   The **Test** button opens the Test flow dialog, where you can create a message to send to the flow. The system tests the flow with this newly created message, not with messages from the Kafka topic. The flow doesn't start receiving messages from the topic until the flow is activated. For more information, see Test a flow.
10. To activate the flow, click the **Activate** button.
    When you activate the flow, the system begins looking for messages in the topic. You must activate the flow to receive messages. For more information, see Activate a flow.

Result

When there's a message in the Kafka topic, the flow triggers and runs the actions.

Create a flow with a MetricBase trigger

Start a flow when a MetricBase trigger is met. MetricBase triggers track time series data and can monitor when a threshold is reached, when a trend is detected, or when a system stops reporting data.

Before you begin

Role required: flow_designer or admin

MetricBase triggers are not available on the base system. The MetricBase application requires a separate subscription and must be activated by ServiceNow personnel.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Click **+ New** > **New Flow**.
3. Define the flow properties.
   For more information, see Create a flow in Workflow Studio.
4. In the Trigger section, add a trigger and select **MetricBase**.
5. Complete the trigger fields.

Field	Description
MetricBase Trigger	Select or create a MetricBase trigger record to start the flow. For types of MetricBase triggers, see MetricBase triggers.

Field	Description
Table	Read-only table that contains the metric.
Condition	Click **Add filters** to set field-based conditions that determine when the flow runs.
Condition Script	Define a script in the Additional MetricBase Trigger Filtering and Moderation table to prevent duplicate metric events from re-triggering a flow. For example, if a metric hovers at a defined trigger threshold, create a script that defines whether to run the flow once when the metric is met, or every time the triggering threshold is exceeded.

6. Add actions, subflows, and flow logic to the flow.
7. Test the flow.
8. Once the flow is behaving as desired, activate the flow.
   For more information, see Test a flow and Activate a flow.

Result

The MetricBase application monitors time series data on the ServiceNow AI Platform. When the selected trigger is met, the flow runs.

Create a flow with a Proactive Analytics trigger

Use Performance Analytics indicators to start a flow. Define the flow start conditions as a set of Proactive Analytics KPI scores and KPI threshold values.

Before you begin

Important:
This flow supports Performance Analytics indicators inside Platform Analytics. The flow requires a subscription to Performance Analytics. Furthermore, you should have a working Performance Analytics implementation. For more information, see Performance Analytics (Indicator data sources).

To create the flow, you should be familiar with Performance Analytics artifacts such as indicators and their associated targets and thresholds. You should also be familiar with your organization's Performance Analytics implementation. If you are not an expert in Performance Analytics, consider contacting such an expert from inside your organization. This expert can identify your organization's use cases for Proactive analytics triggers, which you can then implement. Even if you are only testing the flow, this expert can identify an appropriate indicator to test it on.

- Activate your Platform Analytics subscription. See Activating your Performance Analytics subscription.
- Select the relevant Performance Analytics indicators (KPIs) and KPI Signals.
- Run the **Proactive Analytics Trigger Definition Activation Job** from Schedule [sys_trigger].

Role required: admin or flow_designer

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. From the Workflow Studio home page, select **New** > **Flow**.
3. Define the flow properties.
   For more information, see Create a flow in Workflow Studio.

4. In the Trigger section, add a trigger and select **Proactive Analytics**.
5. For **Proactive analytics trigger**, select a trigger type.

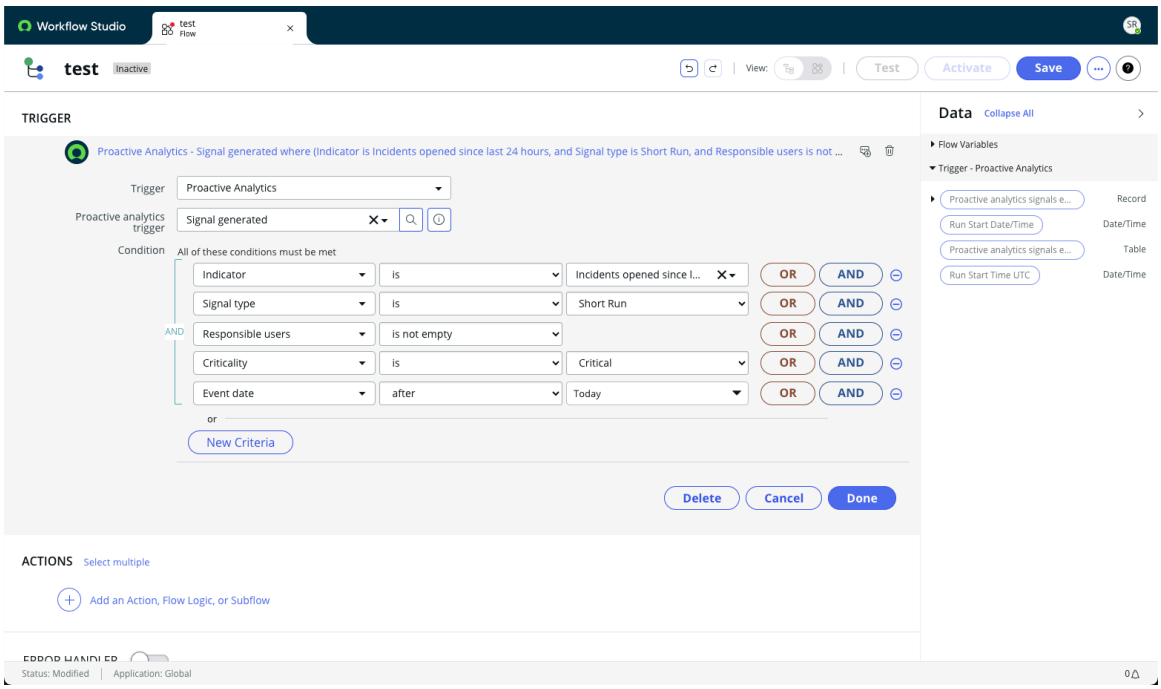
Proactive Analytics trigger type	Description
Predictive target missed	Start the flow when a KPI target is likely to be missed.
Predictive threshold breached	Start the flow when a KPI threshold is likely to be breached.
Signal generated	Start the flow when a KPI indicator generates a signal.
Target missed	Start the flow when a KPI target is missed.
Threshold all time	Start the flow when a KPI threshold is breached by an all-time high or low value.
Threshold breached	Start the flow when a KPI threshold is breached.

Note:
For more information about Proactive Analytics options, see Proactive analytics insights on dashboards.

6. In **Condition**, select the KPI indicator, target, or threshold values needed to start the flow. The list of Conditions varies by Proactive Analytics trigger type. You can use the condition builder to add or remove conditions as needed.

Example

For example, start a flow when the Incidents opened since last 24 hours indicator generates a Short Run signal.



7. Add actions, subflows, and flow logic to the flow.

Example
For example, send a notification or create a task.

8. Test the flow.
   For more information about testing a flow, see Test a flow.

Related topics

Performance Analytics indicators

KPI Signals

Performance Analytics targets and thresholds

Create a flow with a Service Catalog trigger

Start a flow when a Service Catalog item is requested to automate the fulfillment process.

https://player.vimeo.com/video/984477123?h=6189b3336b&badge=0&autopause=0&player_id=0&app_id=58479

Before you begin
Role required: flow_designer or admin

About this task
Unlike a record trigger which runs on all records in a table, the Service Catalog trigger runs on all catalog item requests for a specific item when the catalog item is configured to run a specific flow. For example, you can define a flow that runs every time there is a request for a tablet. The requested item becomes the flow trigger record.

Procedure

1. If not already created, create the catalog item to associate with the flow.

   a. Navigate to **Service Catalog** > **Catalog Definition** > **Maintain Items**.
   The Catalog Items [sc_cat_item] table opens.

   b. Click **New**.

   c. Complete the fields.
   For a detailed description of catalog item fields, see Catalog Item form.

   Note:
   If you plan to add stages to your flow, verify that there is not an existing workflow associated with the item that also has stages. Clear the values of the **Workflow** and **Execution Plan** fields to prevent conflicting stages from reporting to the requested item stage field.

2. Create the flow to associate with the catalog item.
   When triggered, this flow processes the catalog item request.

   a. Navigate to **Process Automation** > **Flow Designer**.

   b. Click **+ New** > **New Flow**.

   c. Define the flow properties.
   For more information, see Create a flow in Workflow Studio.

d. In the Trigger section, add a trigger and select **Service Catalog**.

> **Note:**
> Service Catalog triggers do not support catalog variables as part of the trigger condition. Instead, get or create catalog variables in the main body of the flow.

e. **Optional:** Create flow-specific catalog variables available only to flow actions.
See Create flow Service Catalog variables.

f. Add actions, subflows, and flow logic to the flow.
Some actions enable you to manage catalog items. For example, the Create Catalog Task action generates a task for the requested item, and the Get Catalog Variables action enables you to access catalog variables as data pills in the flow. See Create Catalog Task action and Get Catalog Variables action.

g. **Optional:** Add stages to the flow to report progress to the requester.
See Flow and subflow stages.

h. Test the flow.
Once behaving as desired, activate the flow. For more information, see Test a flow and Activate a flow.

> **Note:**
> You can't activate a flow if it references catalog variables that are inactive or don't exist.

3. Add the flow to the **Flow** field of the catalog item you created.

a. Navigate to the catalog item.

b. In the **Flow** field, select the flow you created.

> **Note:**
> The **Flow** field only displays flows with a Service Catalog trigger. If the **Flow** field is not visible, configure the form to display it.

c. Click **Update**.

Result
When the catalog item is requested, the associated flow triggers and runs the actions.

What to do next
Create and deploy catalog item records to your instances.

> **Note:**
> Service Catalog records are created in the global scope. They are not part of the source control or application repository transfer, and they are not part of the default Workflow Studio update set.

Create a flow with an SLA Task trigger

Configure your Service Level Agreement (SLA) definition to run a flow as the action plan.

Before you begin

- Set up an application in Guided Application Creator to store Workflow Studio content.
- Role required: flow_designer or admin

About this task

An SLA Task trigger only runs when a task record matches the conditions of a Service Level Agreement (SLA) definition. For example, you can run a flow whenever an incident record matches the **Priority 1 resolution (8 hour)** SLA Definition.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select **+ New** > **New Flow**.
3. On the form, fill in the fields.

Flow Properties form

Field	Description
Name	Name to uniquely identify your flow. The system computes the internal name of the flow from the name.
Application	Application scope to create your flow in. Global is the default.
Description	Description of your flow.
Protection	Selection to specify 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	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 the flow. Settings for the Run as option in a flow don't apply to child subflows. The system runs flow actions as the user who initiates the flow. Verify that all users who can trigger a flow have the necessary security access to run its actions. The initiating user also determines user-specific settings such as 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.

4. Select **Submit**.
   The system displays the Flow Designer page.
5. Select **Select to add a Trigger** > **Application** > **SLA Task**.
6. Add actions, subflows, and flow logic to the flow.
   Add SLA Percentage Timer actions to specify what to do when a task record attached to an SLA reaches specific percentages of completion.
7. Test the flow.

For more information, see Test a flow and Activate a flow.
Once behaving as desired, activate the flow.

8. Create an SLA definition

   a. In the **Flow** field, select the SLA Task flow you previously created.

   > **Note:**
   > The **Flow** field only displays flows with an SLA Task trigger.

   b. Click **Submit**.

Result
When the SLA Definition conditions are true, the system runs the specified SLA task flow.

Create a flow with an external trigger

Set up a flow with a base system external trigger definition. The flow responds to an event-driven external trigger from a third-party system. For example, run a flow when an issue is created in a third-party issue-tracking system.

Before you begin
Role required: flow_designer or admin

Subscription required: This feature requires an Integration Hub subscription. For more information, see Legal schedules - IntegrationHub overview.

Procedure
1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select **Create New** > **Flow**.
3. Fill out the **Flow properties** form.

Field	Description
Name	Name to uniquely identify your flow. The system creates the internal name of the flow from this name.
Description	Description of your flow.
Application	Application scope to create your flow in. Global is the default.
Protection	Specify if the flow is read-only. **Note:** This is an optional field with two values: None and Read-only. Choose read-only if the flow is within an application scope that you own.
Run as	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 the flow. Settings for the Run as option in a flow don't apply to child subflows.

Field	Description
	Note: By default, flows run as the user who initiates the session. 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 flow actions by user Access Control Rule (ACL) restrictions. Ensure that security restrictions don't prevent users who trigger the flow from performing flow actions. Flows run by the initiating user also respect user-specific settings such as 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**.

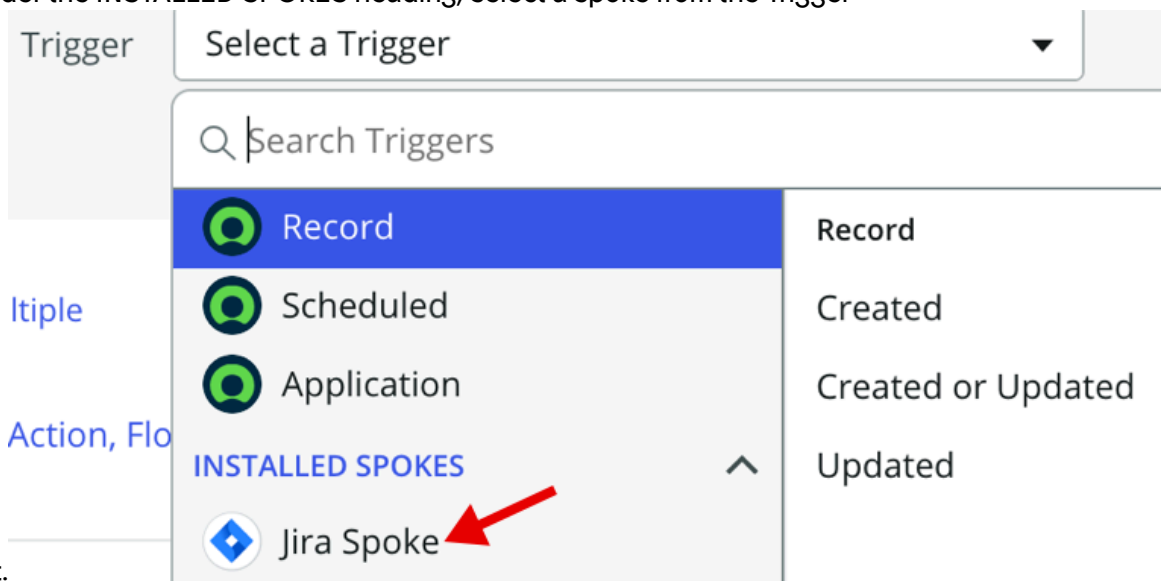
4. Select **Submit**.

   Note:
   If this is your first time in Workflow Studio, a welcome screen appears. You can choose to either take the welcome tour or skip the tour.

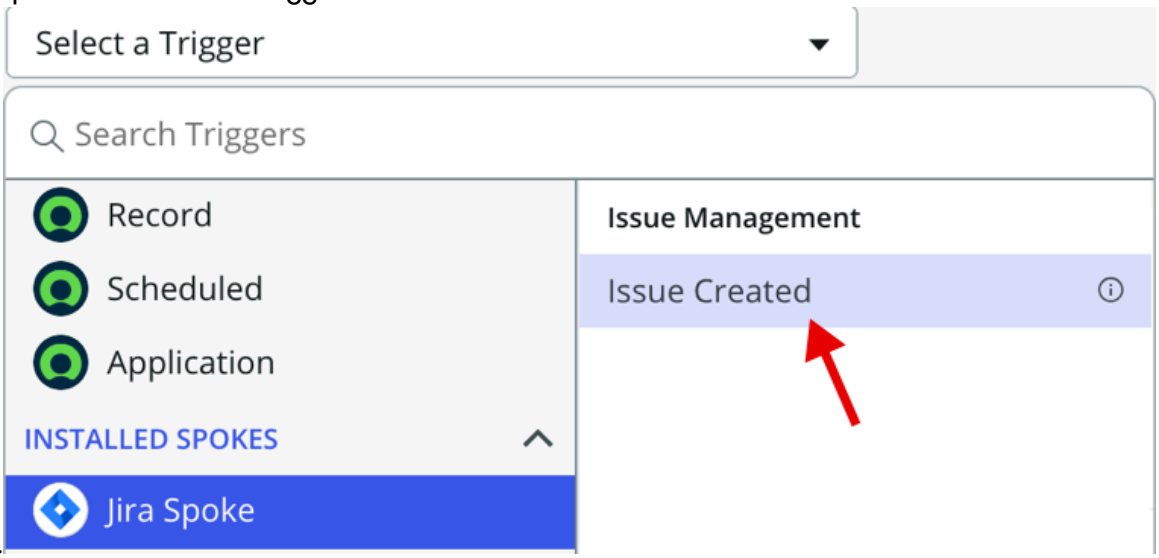
5. Add a trigger to your flow.

   a. Under the TRIGGER section, select **Add a trigger**.

   b. Under the INSTALLED SPOKES heading, select a spoke from the Trigger list.



c. Select the predefined external trigger definition.



To learn more about external trigger definitions, see Conditional and event-driven inbound integration.

d. **Optional:** To define the conditions for the flow to execute, in the Condition 1 field, drag the appropriate data pill and enter the value.

e. Select **Done**.
You've set up the trigger.

6. Add actions, subflows, and flow logic to the flow.
7. Test the flow.
8. Activate the flow.
Activate the flow if the test returns the desired outcome. For more information on testing and activating your flow, see Test a flow and Activate a flow.

Create a flow with Now Assist

Use generative AI to create a flow from text directions.

https://player.vimeo.com/video/995190972?h=16b49d841d&badge=0&autopause=0&player_id=0&app_id=58479

Before you begin

- Roles required:
  - admin, flow_designer, or a delegated developer permission
  - now.assist.creator
- Turn on the flow generation skill. See Turn on the flow generation skill.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. From the Workflow Studio home page, select **New** > **Flow**.
3. Select the **Build with Now Assist** tab.

4. Fill in the fields to build your flow.

Field	Description
Flow name	Name to uniquely identify your flow. The system converts the flow name into an internal name by replacing space characters with underscore characters.
Now Assist directions	Text used by Now Assist flow generation skill to create your flow. Describe the flow trigger and its data first. Then describe all the actions and flow logic in the order you want them in the flow. Include as much detail as possible for the best results.
Try an example	Series of text examples to insert into the Now Assist directions field. Select this option to insert example text in the directions field. Each example illustrates directions that will build a flow outline. Trying an example overwrites any directions text that you previously added. You can use the undo and redo options to revert to your previous directions.
Application	Application scope to create your flow in. Global is the default. The application scope determines what data your flow can access and what data it can share. **Important:** You can't change the application scope of a flow after you've generated a preview for it.

Follow these general guidelines when writing Now Assist directions.

Always describe the trigger first

Describe the flow trigger and its data conditions first. After the trigger, describe the actions and flow logic in the same order that you want them to be in the flow.

Avoid spelling errors

Avoid misspelling the names of actions, flow logic, or tables. Consider using hash tags to avoid making mistakes with table names.

Be precise and descriptive in your request

Make sure that your request is precise and descriptive. Describe the flow trigger, record data, actions, and flow logic in as much detail as you can.

Be succinct and direct in your request

Start by specifying whether you want to generate a flow or a subflow. For example, use the phrase, "`Create a flow that`" to generate a flow. Describe each step the flow in order.

Refer to actions, flow logic, and tables by name

Use action, flow logic, and table names as part of your directions. The closer your directions are to the actual names, the easier it is for the LLM to recognize them. For example, use the text `for each` or `do the following in parallel` to refer to those specific flow logic options. For table names, consider using hash tags.

Review the generated flow outline and input values

Review each action, flow logic, and subflow in the generated flow outline. Review the generated inputs values to confirm that they contain relevant data references.

Use hash tags to refer to data in a specific table

Use a hash tag to select a specific table name. Hash tags are particularly useful to distinguish between tables that have identical or similar display names such as the User [sys_user] and User [imp_user] tables.

Use numbers to distinguish the branches of do the following in parallel flow logic

Add a number to each parallel branch. For example, the directions, "`When a P1 incident is created, do the following in parallel: 1. Log its short description and 2. Look up the user assigned to it and send an email`," makes it clear that there are two branches.

Use quotation marks to set exact values

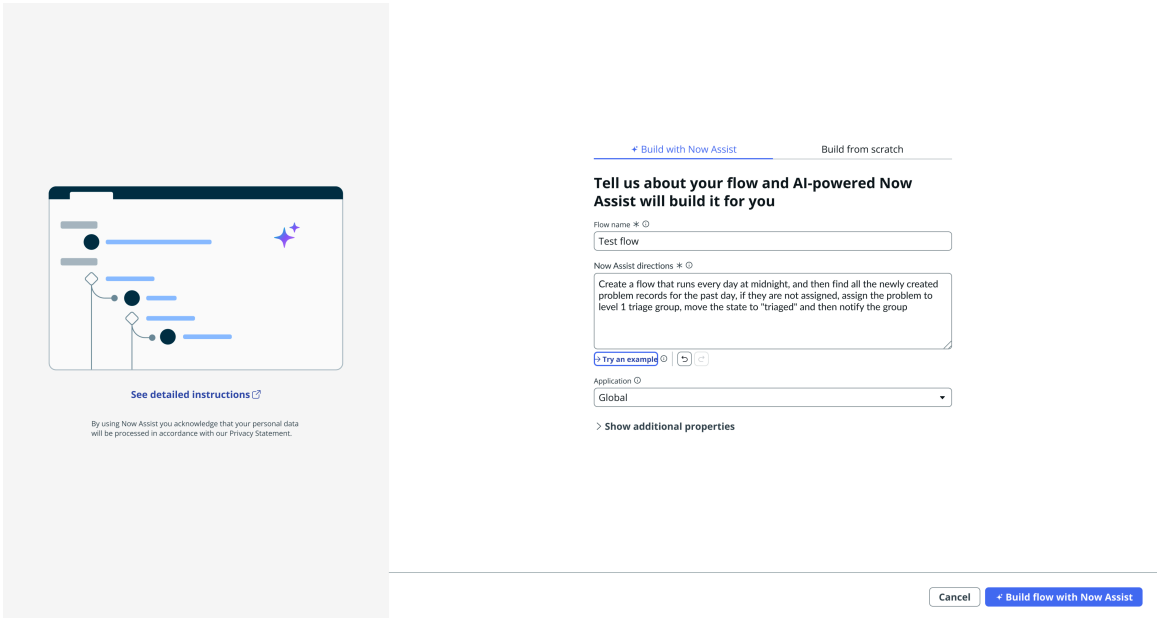
Enclose exact data values in quotations marks to help the LLM distinguish between operation names and data values. For example, the directions, "`Log the value, 'incident reopened'`" make it clear that the text "incident reopened" is a data value.

Show additional properties

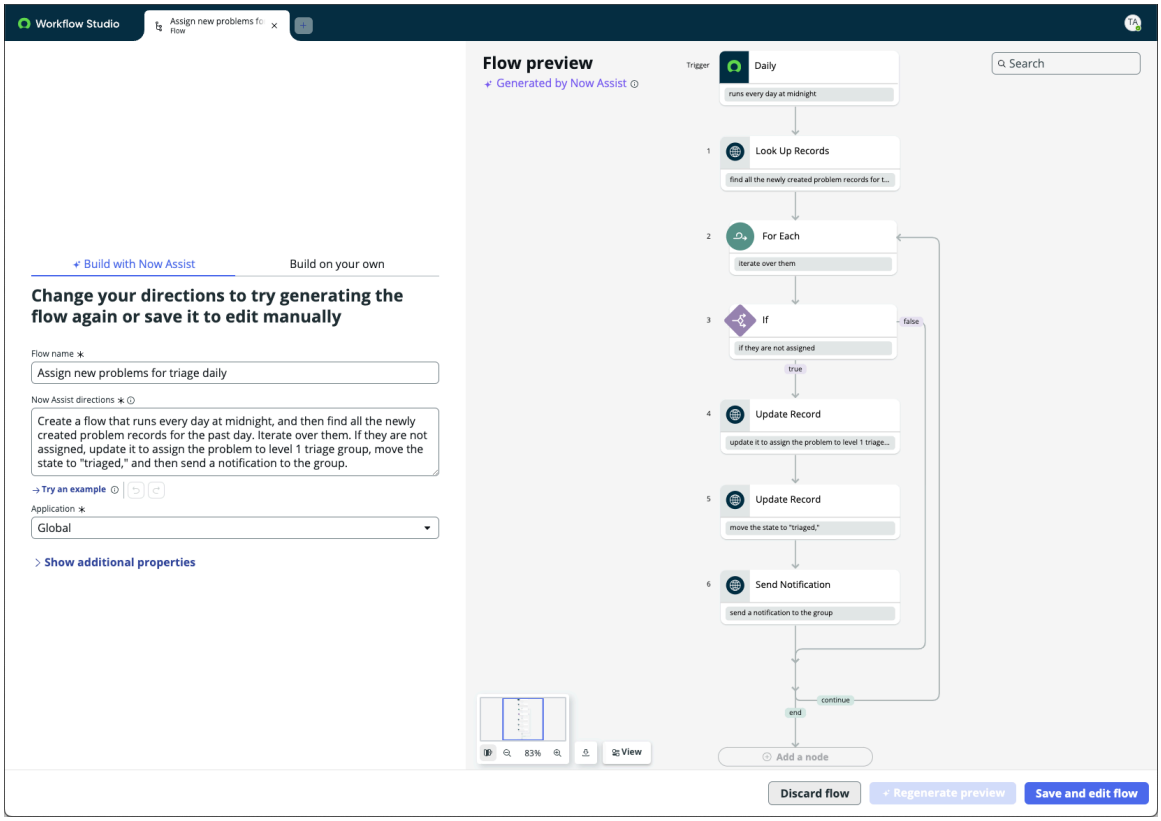
Field	Description
Protection	Selection to specify if the flow is read-only. You can only select a value when you create the flow in an application scope that you own.
Run as	Option to specify the user that runs the flow. You can select the system user or the user who initiates the session. Select the 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 the flow. Settings for the Run as option in a flow don't apply to child subflows. **Note:** By default, flows run as the user who initiates the session. 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 flow actions by user ACL restrictions. Ensure that security restrictions don't prevent users who trigger the flow from performing flow actions. Flows run by the initiating user also respect user-specific settings such as date/time formats.

Field	Description
	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**. You can add one or more roles that the flow can use to access data. These roles only apply to running the flow. For example, add the itil role to flows that need to access ITSM data.
Flow Priority Default	Priority level at which you want the system to run this flow by default. Options include: ○ Low ○ Medium (Default) ○ High To learn about flow priority levels, see Flow priority.

5. **Optional:** Select the **Try an example** option to insert a valid example of flow directions. Each time you select the option, the system inserts a different example of flow directions text. There are three example directions to choose from. Each example direction produces a different type of flow. Selecting Try an example overwrites any directions text that you previously added. You can use the undo and redo options to revert to your previous directions.
6. Select **Build flow with Now Assist**.



Workflow Studio uses your text directions to build a flow outline. If successful, Workflow Studio displays a preview of the flow in the diagramming view. Beneath each node of the flow there is an annotation showing the text directions used to generate the item.



7. **Optional:** If the generated flow preview does not meet your needs, you can update the Now Assist directions, and select **Rebuild flow**.
   Each time you build or rebuild a flow, the operation counts as an assist tracked by your Now Assist subscription. To track your Now Assist usage, see Monitoring Now Assist usage in Subscription Management.
8. **Optional:** If you want to stop creating a flow and return to the Workflow Studio homepage, you can select **Discard flow**.

> **Important:**
> Workflow Studio does not save flow previews. If you discard a flow, all information about it is lost.

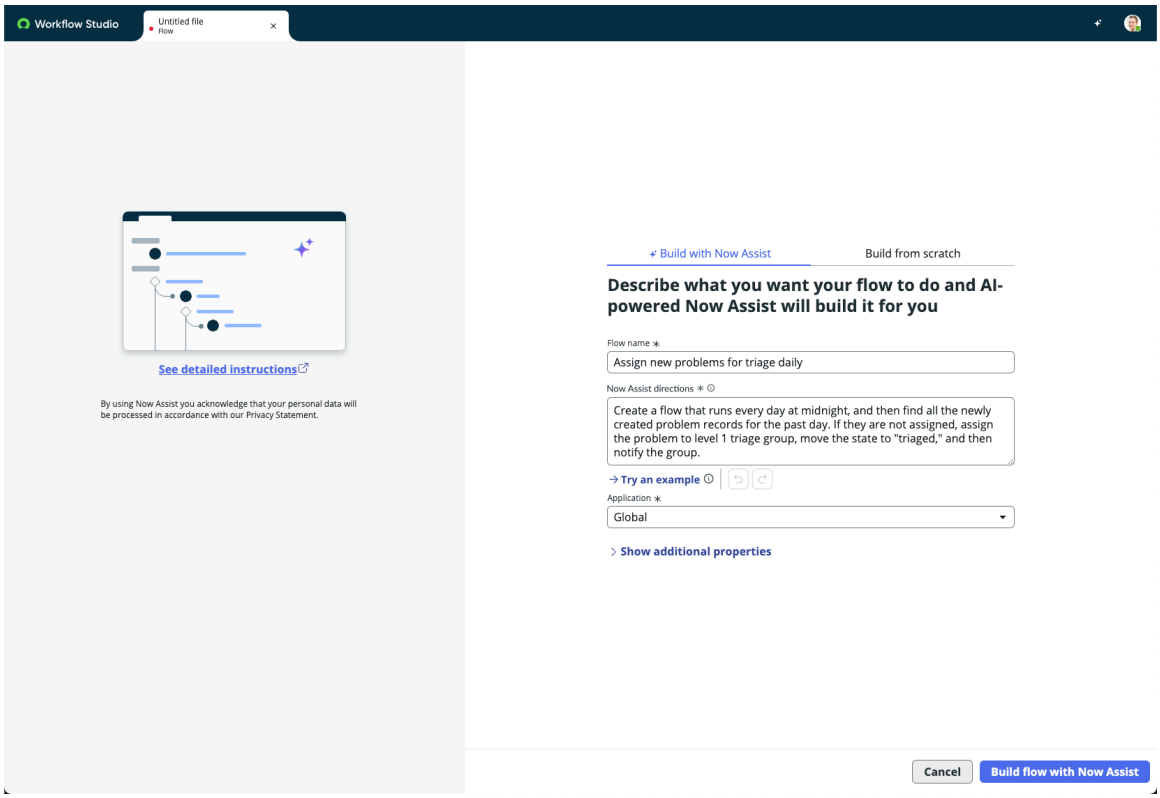
9. Select **Save and edit flow** to review the flow outline.

> **Note:**
> Opening a flow does not count as an assist.

Result

Now Assist creates a flow outline with the name you provided. If the large language model couldn't find a matching action, flow logic, or subflow for your request, it may add one or more placeholder steps instead.

Example: Create a flow with a scheduled trigger



You can use these prompt values to create a flow with a scheduled trigger.

Flow name

Assign new problems for triage daily

Now Assist directions

Create a flow that runs every day at midnight, and then find all the newly created problem records for the past day. Iterate over them. If they are not assigned, update it to assign the problem to level 1 triage group, move the state to "triaged," and then send a notification to the group.

What to do next

- Replace any placeholder steps with actual actions or subflows.
- Configure the inputs of each action, flow logic, and subflow to use appropriate data.
- Test the flow before activating it.

Create a flow with roles

Create a flow or subflow that runs with assigned roles. Assigning roles enables you to create a user-initiated flow that runs with its own roles rather than the user's roles.

Before you begin

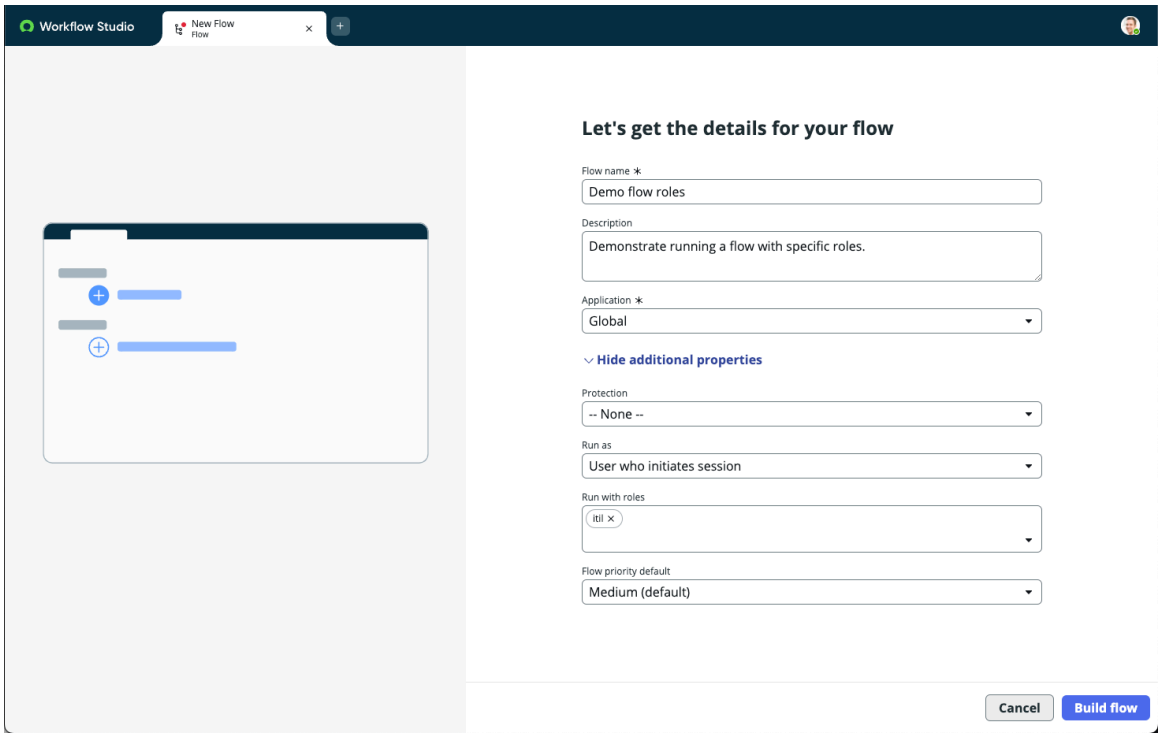
Role required: flow_designer or admin

About this task

Create a user-initiated flow that runs with its own roles and not the roles of the user. For more information about assigning roles to a flow, go to Flow roles. For example, allow a flow to run with the itil role so that it can access data belonging to IT Service Management applications such as incidents and problems.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Click **+ New** > **Flow** or **+ New** > **Subflow**.
3. On the Flow Properties form, define the Name, Application, and Description for the flow.
   For more information, see Create a flow in Workflow Studio.
4. Expand the **Additional properties** section.
5. In the **Run As** field, select **User who initiates session**.
   Role selection is not available if the **System User** option is selected in the **Run As** field.
6. In the **Run with roles** field, select one or more roles that you want the flow to use while it runs.



The roles you select replace any roles that the user normally has. If you don't select any roles, then the flow runs with the roles normally associated with the user.

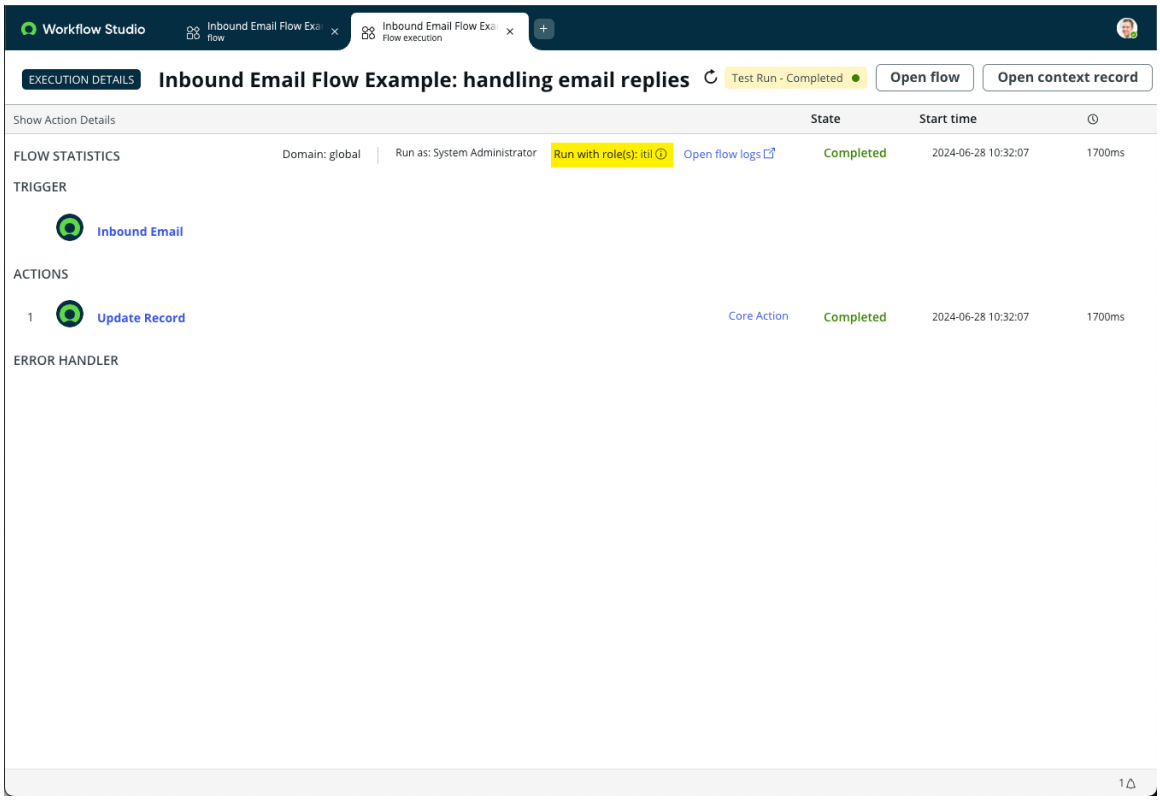
Tip:
If you have the Explicit Roles plugin (com.glide.explicit_roles) activated, add the snc_internal role to your flow.

Example

For example, an inbound email flow normally runs as an existing user or as the Guest user when there is no existing user. Guest users don't have access to IT Service Management data such as incidents and problems. Running a flow without any roles may produce access errors when the flow tries to access restricted data on the guest user's behalf. Running a flow with a role such as itil ensures that the flow can access the data it needs.

7. Select **Build Flow**.

What to do next



Continue to build and test your flow until you're ready to activate it. You can modify your flow's roles at any time by updating the Flow Properties form.

Create flow Service Catalog variables

Create Service Catalog variables that are only available to a specific Service Catalog flow. Flow-specific variables are available to catalog tasks and actions in the flow.

Before you begin

Role required: flow_designer or admin

About this task

Flow Service Catalog variables display in the **Catalog Variables** field of the Create Catalog Task and Get Catalog Variables actions. They display in the **Flow:variablename** format and are only available to the flow in which they are defined.

For more information about Service Catalog actions, see Create Catalog Task action and Get Catalog Variables action.

Procedure

1. Open or create a flow with a Service Catalog trigger.
2. Click ... and select **Manage flow catalog variables**.
   The Flow catalog variables table opens.
3. Click **New** to add a new variable available to the flow.
4. Complete the form.

Variable form

Field	Description
Application	Read-only field that indicates which applications can use this variable.
Map to field	Maps the variable to a specific field on the table for the record producer. This field appears if the variable belongs to a record producer.
Type	The variable type that you want to create. For more information, see Types of service catalog variables.
Catalog item	Catalog item using the variable.
Order	Order that the variable is placed on the page for the catalog item. The variables are organized from top to bottom from least to greatest order value. For example, a variable with an order value of 1 is placed above other variables with higher-order values.
Active	Check box to make the variable available for use
Mandatory	Check box to make the variable mandatory as part of the ordering process. **Note:** This behavior is applicable only on page load, and can be changed via client APIs.
Read only	Option to make a variable read only or editable.
Hidden	Option to hide a variable.
Unique	Option to disallow duplicate values for this variable within a multi-row variable set. **Note:** ○ This field appears only for a variable created in a multi-row variable set. ○ This behavior is applicable only in ServiceNow AI Platform, Service Portal, and variable editor in Agent Workspace.
Selection Required	Check box to require users to select the check box variable. For example, use this option to require users to select an **I agree** check box for an agreement form. If users try to submit the agreement form without selecting the check box, an alert message is displayed to tell users that they must select the check box.
Global	If selected, the variable is available for all catalog tasks within service catalog workflows or execution plans by default. If deselected, the variable must be associated with individual catalog tasks.
Field	Field that the variable maps to. This field appears if the variable belongs to a record producer.
Record producer table	Table that the record producer creates a record in.

Field	Description
	This field appears if the variable belongs to a record producer.
Question	
Question	Question to ask users ordering the catalog item.
Name	A name to identify the question. **Note:** If this field is empty, its value is auto-populated based on the **Question** field for all variable types except break, container split, and container end.
Tooltip	Tooltip text to display when users point to the variable. Enter a brief note to describe the purpose of the 'Question'.
Example text	Hint that is displayed in the question field before a user enters a value. Applicable for the following variables: ○ IP Address ○ Email ○ URL ○ Single Line Text ○ Wide Single Line Text ○ Multi Line Text ○ Date ○ Date/Time
Rich Text	Formatted label to be displayed on a catalog item form. Applicable for the Rich Text Label variable.
Annotation	
Show help	If selected, displays the help text and instructions for the variable. **Note:** ○ It is not available for break and check box variables. ○ Help text and instructions are not available for a variable set.
Always Expanded	If selected, the **Help text** and **Instructions** field value are expanded by default when the catalog item page loads. This check box appears only when the **Show help** check box is selected.

Field	Description
	Note: ○ This field is also applicable in Service Portal. ○ This field is not applicable in the variable editor in Workspace and Catalog Item UIB component. ○ If the **Expand help for all questions** check box is selected at the catalog item level, then the **Always Expanded** field setting at the variable level is overridden. ○ If the **Expand help for all questions** check box is deselected at the catalog item level, then the **Always Expanded** field setting at the variable level is applicable.
Help tag	If the **Always Expanded** check box is deselected, click the value specified in this field to display the **Help text** and **Instructions** field values. This field is not applicable in the variable editor in Workspace and Catalog Item UIB component.
Help text	Help information for a service catalog variable. This field is applicable in ServiceNow AI Platform, Service Portal, and variable editor in Workspace, and Catalog Item UIB component. However, in Workspace and Catalog Item UIB component, you can view either help text or instructions. If both instructions and help text are available, you can view only the instructions. This field is not applicable for Break, CheckBox, Container End, Container Split, Macro, and UI Page variables. In Workspace and Catalog Item UIB component, this field is additionally not applicable for a Masked variable.
Instructions	Information that requires rich text formatting or adding images to support help information. This field is applicable in ServiceNow AI Platform, Service Portal, and variable editor in Workspace, and Catalog Item UIB component. However, in Workspace and Catalog Item UIB component, you can view either help text or instructions. If both instructions and help text are available, you can view only the instructions. In Workspace and Catalog Item UIB component, this field is additionally not applicable for a Masked variable. **Note:** For HTML tables, use sizes that are within the width of the variable.
Type Specifications (The fields in this section vary for each variable type)	
Variable Width	Width for the variable on the catalog item page, to specify what percentage of the screen size that it can span. For details, see Configure a default width for service catalog variables.

Field	Description
	This field appears for all variable types except for break, container end, container start, container split, container layout, and label variables.
Enable also request for	Option to allow a catalog item request to be submitted for multiple users. After you select this option, the **Also request for** field is displayed along with Requested For variable in a catalog item. ○ This functionality is only applicable in Service Portal. ○ This field is applicable only for the Requested For variable. For information about delegated request experience, see Delegated request experience.
Roles to use also request for	Option to specify the roles that can submit a catalog item request for multiple users. ○ This functionality is only applicable in Service Portal. ○ This field is applicable only for the Requested For variable. ○ This field appears only when the **Enable also request for** check box is selected. **Note:** If no role is specified, anyone who has access to the catalog item can submit the request. For information about delegated request experience, see Delegated request experience.
Choice direction	The direction in which the choice list is arranged. ○ **Across**: Arranges choices horizontally. ○ **Down**: Arranges choices vertically. This field appears for lookup multiple choice variables. **Note:** The selected direction is also applicable in Service Portal.
Choice field	Table field to populate options for the variable. If no choices are defined for a field, then the variable loads field-related distinct values from the table. This field appears for select box variables.
Choice table	Table with values to populate in the **Choice field**. This field appears for select box variables.
Do not select the first choice	Check box to leave all options for the variable cleared on the catalog item page. If this check box is selected, the first choice for the variable selected by default. This field appears for multiple choice and numeric scale variables.
Dynamic ref qual	Dynamic qualifier. Select a dynamic filter to run a query against the reference field.

Field	Description
	This field appears for reference variables when **Use reference qualifier** is set to **Dynamic**.
Include none	Check box to include the **None** option in a list of choices. This field appears for lookup multiple choice, lookup select box, multiple choice, and select box variables.
Layout	Layout for a container, whether one or two columns. This field appears for container start variables.
List table	Table with the values for the list collector. The table should have a display column specified. This field appears for list collector variables.
Lookup from table	Table from which values are obtained for users to select. The values from this table are populated in the **Lookup value field**. This field appears for lookup multiple choice and lookup select box variables.
Lookup value field	Field in the lookup table that populates options for the variable. This field appears for lookup multiple choice and lookup select box variables.
Lookup label field(s)	Comma-separated list of fields in the lookup table whose values are used to display options. This field appears for lookup multiple choice and lookup select box variables.
Lookup price field	Field in the lookup table whose value is used to modify the price of the item being ordered. This field appears for lookup multiple choice and lookup select box variables.
Lookup recurring price field	Field in the lookup table whose value is used to modify the recurring price of the item being ordered. This field appears for lookup multiple choice and lookup select box variables.
Macro	UI macro to insert into the catalog item. This field appears for macro, macro with label, and UI page variables.
Summary macro	Applicable only for Marco, and Macro with Label type variables.
Widget	Applicable only for Marco, and Macro with Label type variables.
Price if checked	Price of the item.

Field	Description
	This field appears for check box variables.
Recurring price if checked	Price that increments for the item, when the user requests more than one order of the item. This field appears for check box variables. For more information about prices and recurring prices, see Using variables for price setup.
Reference	Reference table for the variable. The table should have a display column specified. This field appears for reference variables.
Reference qual	Qualifiers to restrict data that is available in the field. Supports reference qualifiers and advance qualifiers. For more information, see Reference qualifiers. Returns all matching results (no maximum). **Note:** For security reasons, the use of scripts in the **Reference qual** field is restricted to system administrators through the **Allow javascript in Default Value** business rule. This field appears for list collector, lookup multiple choice, lookup select box, reference and Requested For variables. It appears for reference variables when **Use reference qualifier** is set to **Dynamic**.
Reference qualifier condition	Options to build conditions. This field appears for reference variables when **Use reference qualifier** is set to **Simple**.
Scale max	Highest value on the scale of available options for the variable. This field appears for numeric scale variables.
Scale min	Lowest value on the scale of available options for the variable. This field appears for numeric scale variables.
Unique values only	Check box to require a unique value for the field. When this check box is selected, two records cannot have the same value for that field. This field appears for lookup multiple choice, lookup select box, and select box variables.
Use confirmation	Check box to prompt users to reenter data to verify their entries. This field appears for masked variables.
Use encryption	Check box to store the answer in encrypted format in the database. If not encrypted, the answer is stored in plain text format. Encryption uses Triple DES with system encryption.

Field	Description
	This field appears for masked variables.
Use reference qualifier	Type of qualifier to use. This field appears for reference variables.
Validation Regex	Regular expression that validates the variable value. This field is displayed only for Single Line Text and Wide Single Line Text variable types. To define regular expressions, see Define a regular expression for a variable. **Note:** ○ This field is also applicable in Service Portal. ○ The max_length attribute value is valid even when the validation regex is set. ○ You cannot add a catalog item with regex validation errors to the wishlist.
Variable attributes	Attributes that define the behavior and restrictions for a variable. For information on variable attributes, see Service catalog variable attribute.
Default Value	
Default value	Default value for the variable.
Permission If no role is specified in this tab for the read, write, or create actions, all users who can access the catalog item can perform these actions irrespective of their role. For example, if no role is specified for the **Write roles** field, all users who can access the catalog item can edit the variable value in the variable editor. A user with a role that does not match any of the following roles cannot set variable values even through scripting. These roles are not available for Label, Break, Container Split, Container End, Macro, Macro with Label, and UI Page variables.	
Read roles	Roles that can view the variable before or after requesting the catalog item or record producer. Only a user with the roles specified in this field can view the variable.
Write roles	Roles that can edit the variable in the variable editor after requesting the catalog item or record producer. If a user does not have the roles specified in this field, the variable is read-only in the variable editor.
Create roles	Roles that can create values for the variable before requesting the catalog item or record producer. If a user does not have the specified role, the variable is read-only before requesting the catalog item or record producer.
Availability	
Visible Elsewhere	If selected, the variable is visible in the item form before ordering the item, in VEditor after ordering the item, and in the cart view of the item.

Field	Description
Visible on Bundles	If selected, the variable is visible when the item is added to a bundle.
Visible on Guides	If selected, the variable is visible when it is added to an order guide, or when it is added to a catalog item that is included in the order guide. **Note:** If an order guide has too many items and variables, consider clearing this check box on as many items as possible, to improve loading performance on order guides.
Remove from Conversational Interfaces	If selected, the variable is removed from all the conversational interfaces, such as Virtual Agent. If the catalog item is non-conversational because of a variable, removing the variable makes the item conversational.
Visible on Summaries	If selected, the variable is visible on any variable summarizer of the catalog item. In Service Portal, the variable is visible in the RITM ticket page and the Approval page. In Now Mobile, the variable is visible in the RITM and the Approval records.

5. Click **Submit.**

Result
Access the variable in the flow by adding a Create Catalog Task or Get Catalog Variables action.

Create a decision table in a flow

Create the structure for a decision table while you author your flow in Workflow Studio. Use data from the flow to create inputs, conditions, and results for the decision table, all in a convenient modal. For example: You can store the logic for incident assignments in a Decision Table, and then use that Decision Table within a flow.

Before you begin
Role required: admin, decision_table_admin, flow_designer, or delegated developer permissions

About this task
Creating a decision table in-line in a flow creates only the structure of the table with inputs, conditions, and results. To complete the table by adding the actual decision rules, you must open and edit the decision table in its own tab.

Procedure
1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Flows**.
3. Select a flow.
4. Under Actions, select **Flow Logic**.
5. Select **Make a Decision**.
6. In the **Decision Label** field, enter a unique label for the decision.

7. In the **Decision Table** field, select the Create new record (+) icon.
   In the Create decision table modal, on the Set Properties page, two editable fields are populated from your flow.

   Set Properties

Field	Description
Decision table name	Unique name for the table. This name is populated from the **Decision Label**.
Application	Application where the decision table lives. The application is populated based on the application that the flow is in.

8. Select **Next**.
9. Select **Add input** to add inputs to the decision table.
   Inputs are the variables that define the type of data the decision table looks for to make decisions. When creating a decision table in a flow, you can add inputs directly from the data in the flow. For more information about the types of inputs you can add, see Create decision tables in Workflow Studio.

   Note:
   Some inputs must be added or adjusted when you open the decision table to populate its values.

   - If you want to add inputs of type Reference to this decision table, you must do that when you open the decision table in its own tab.
   - If you want to add inputs of type Choice in this modal, the choices themselves don't appear when you begin editing the decision table on its own. You must add them manually, because choice options are based on the scope you're in. For example, if you add a **Priority** field with **Choice** as the type, the options for priority (Low, Medium, High, etc.) are not added automatically. You can add them when you open the decision table to populate its values.

10. Select **Next**.
11. Select **Add Result Column** to begin adding the result columns to your table.
    Results are the outputs your decision table returns when certain conditions are met.
12. **Optional:** Add any additional configurations required for the result type that you selected.
13. Select **Create and Open**.
    The decision table opens in its own integrated tab within Workflow Studio. You can edit any part of the table here, including adding Reference type filters to the inputs or results.

    Condition columns are added to the decision table based on corresponding input fields.
14. Complete the decision table by adding decision rules to the condition columns and result values.
15. Select **Save**.
16. **Optional:** In your original flow, review and test the decision to make sure it produces the expected results.
17. Select **Done**.

Copy a flow

Copy a flow to give it a new name and move it to another application scope.

Before you begin

Role required: admin or flow_designer

About this task

You can copy a flow to give it a new name or move it to another application scope. The new flow has the same flow properties, trigger, actions, and flow logic as the source flow.

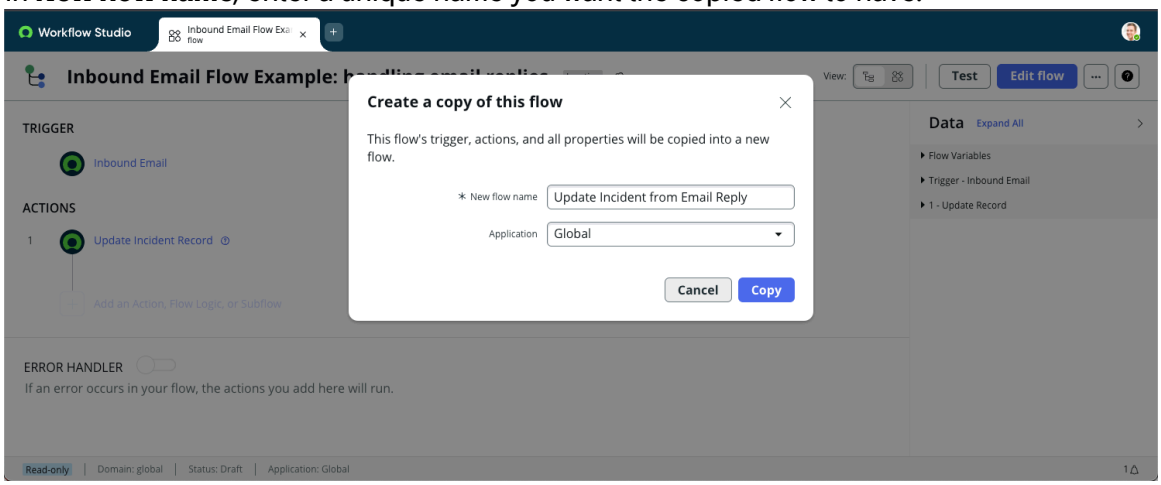
You can't copy flows that have a protection policy. You must have write access to a flow to copy it.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Flows**.
3. Open the flow that you want to copy.
4. Click the more actions icon (...) and select **Copy flow**.

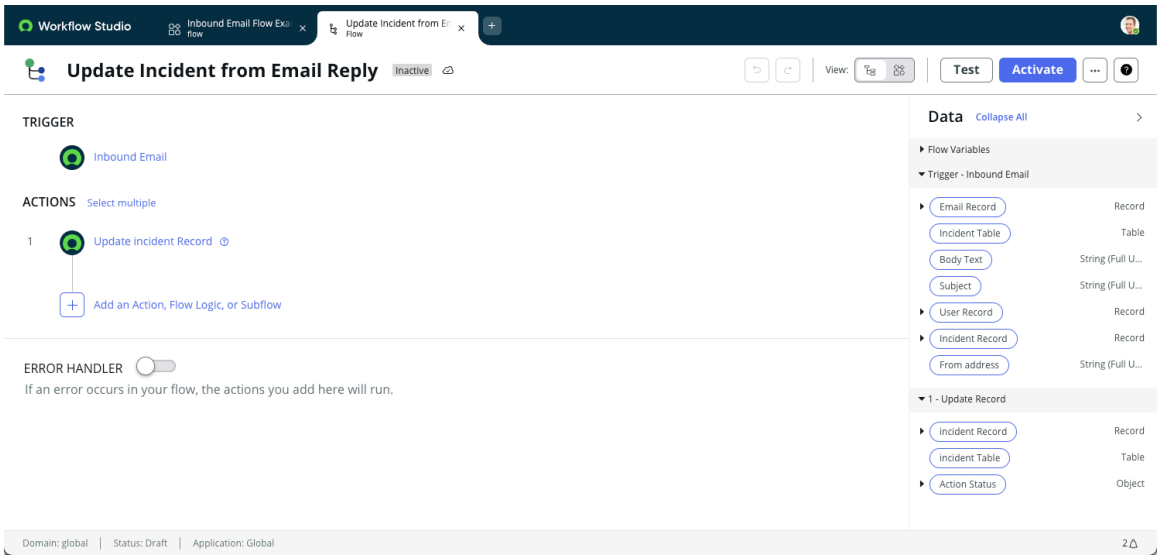
> **Note:**
> If the **Copy flow** option is not visible, then you don't have permission to copy the flow. This could be because the flow has a protection policy or because you lack the necessary user role or developer permissions.

5. In **New flow name**, enter a unique name you want the copied flow to have.



6. **Optional:** From **Application**, select the application scope where you want to copy the flow.
7. Select **Copy**.

Result



Workflow Studio opens the new flow.

Duplicate an action or subflow

Duplicate an action or subflow within a flow.

Before you begin

Role required: admin or flow_designer

About this task

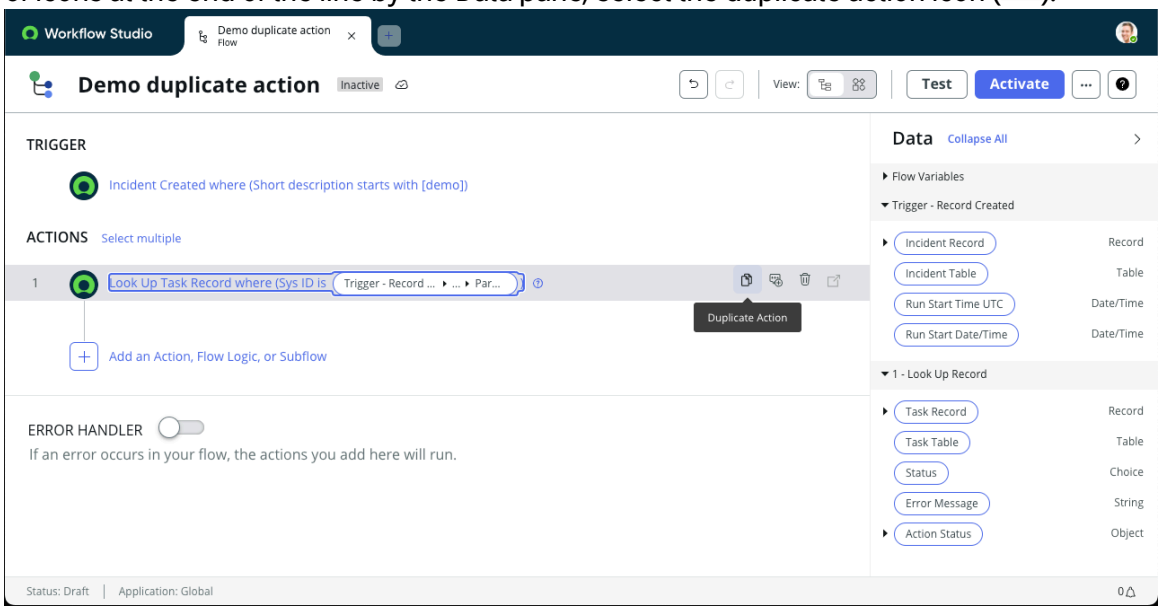
This procedure can only duplicate an action or subflow within the current flow. You can't directly duplicate an action to move it from one flow to another. If you need to move content between flows, you can use one of these alternatives.

- Create a custom action that hard-codes the values you want to duplicate between flows. For more information about creating a custom action, see Create an action in Workflow Studio.
- Convert one or more items in the current flow to a subflow. Call the subflow from each flow that needs the duplicated content. For more information about converting items to a subflow, see Convert items to subflow.

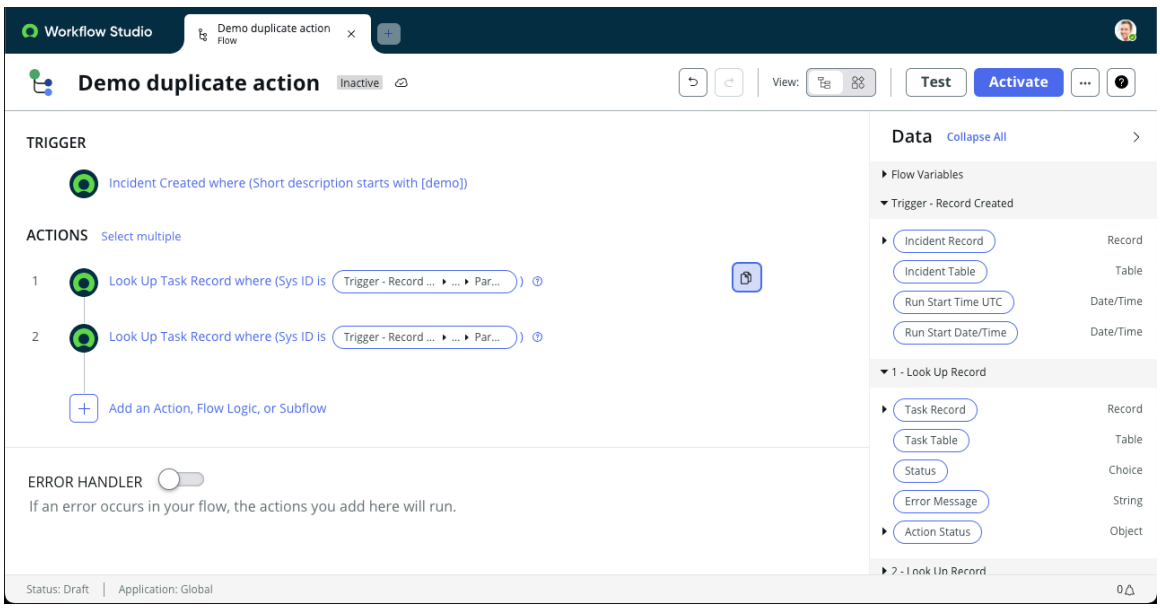
Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Flows**.
3. Select the flow containing the action or subflow that you want to duplicate.

4. Under Actions, point to the action or subflow that you want to duplicate, and then from the list of icons at the end of the line by the Data pane, select the duplicate action icon ( ).



Result



Your selected action or subflow duplicates directly under itself. All configurations, including transform functions, are copied over to the duplicated action or subflow.

Test a flow

Before activating a flow so other users can access it, test to make certain it works the way you expect.

Before you begin

Role required: flow_designer or admin

About this task

Testing a flow bypasses the flow trigger conditions to run it with the test data you provide. For example, testing a flow with a record **Created** trigger causes the system to act as if the selected record was created. For a list of data pills available by trigger type, see Workflow Studio flow trigger types.

Note:
Because testing a flow creates or changes records on the instance, flow designers should always test flows on a non-production instance containing relevant demonstration data.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**, then double-click the row for the flow you want to test.
2. Save the flow.
3. Select **Test**.
   The system displays the Test flow dialog. The contents of the Test flow dialog depend on the type of trigger.
4. If the flow has a record trigger, create or select a record to use for the test.

   To create a record, select the **Create new record** button +.
5. If the flow has a record **Updated** or **Created or Updated** trigger, specify which fields and values changed in the update.

   To specify a field value change, select the **Create new changed field** button [+] for each field whose value you want to change. Complete the changed field details for each changed field.

Field	Description
Field Name	The field updated by the test.
Previous Value	The field value prior to the update.
Current Value	The field value after the update.
Previous Display Value	The field display value prior to the update.
Current Display Value	The field display value after the update.

6. Select **Run Test**.

   Note:
   Select the **Run test in background** option to test a flow asynchronously in the background.

   The system tests the flow.
7. Select **Your test has finished running. View the flow execution details**.

   Note:
   This link is created irrespective of your choice for the **Run test in background** option. If you have selected the **Run test in background** option, the execution details are displayed only after the execution is completed asynchronously in the background. Also, the execution details are associated with the flow only after execution is completed.

   The system displays the flow execution details for the test.

What to do next
Review the Flow execution details.

Activate a flow

Activate a flow to make it available to run.

Before you begin
Role required: flow_designer or admin.

Test that your flow produces expected results and outputs. Verify that you have write access to the flow.

About this task
After testing that your flow runs properly, activate it to make it available to run. Only activated flows run when their trigger conditions are met or when called by a script.

Procedure
1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. Select the **Flows** data pill filter.
3. Select the flow that you want to activate from the list of saved flows to open it.
4. Select **Activate**.

   You must have write access to a flow to activate it. The activate option might be inactive for a number of reasons.
   - The flow might belong to another application scope that you don't have access to.
   - The flow might be marked read-only to protect it.
   - The flow might require developer permissions that you don't currently have.

Change a flow or action's default title

Change the default title for a flow, subflow, or action by adding styled and dynamic text.

Before you begin
Role required: admin, flow_designer, or action_designer

Procedure
1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. On the Workflow Studio landing page, click **New** and then select **Flow**, **Subflow**, or **Action** from the list.
3. In the Workflow Studio main header, click the more actions icon ( ).
4. Click **Change default title**.
5. On the Change default title screen, enter a title.

   a. Use any combination of the following options to create a styled title:

Text style	Example input for new title	Example output in Workflow Studio environment
Bold	A *bold* title	A **bold** title
Italic	An _italic_ title	An *italic* title

Text style	Example input for new title	Example output in Workflow Studio environment
Underline	An ~underlined~ title	An underlined title
Strikethough	A ~~strikethrough~~ title	A strikethrough title
Title (bold and colored)	A #titled# title	A titled title

b. Add dynamically generated text for your title from an input, output, action, or action step by clicking the data pill picker ( ) and selecting the input, output, action, or action step that you want to include in your title.

> **Note:**
> The value that is associated with the **Label** field for an input or output appears in the title.

6. Click **Submit**.

Result
When you change your action or subflow's default title, the new title appears in the Workflow Studio environment.

Edit a flow

Edit an existing flow.

Before you begin
Role required: flow_designer or admin

About this task
As of the Washington DC release, flows open in a read-only state to protect them from accidental changes. When a flow is in a read-only state, you can only review, test, deactivate, or request to edit it. You must request to edit a flow before you can make changes. While editing a flow, your changes are automatically saved each time you select **Done** to close the configuration options.

> **Note:**
> Workflow Studio no longer displays a **Save** button while you edit a flow.

Procedure

1. Open the flow that you want to edit.
   If necessary, navigate to **All** > **Process Automation** > **Flow Designer**, and select the flow you want to edit.
   Workflow Studio opens the flow in a read-only state, which only provides options to request to edit, review, test, or deactivate it.
2. Select **Edit Flow**.
   Workflow Studio checks whether another person already has the flow open for edit. If the flow is already being edited, you see a message displaying the name of the user editing the flow. If the flow is available for editing, you open the flow in an editable state.
3. Take the appropriate actions to edit the flow.

While editing a flow, your changes are automatically saved each time you select **Done** to close the configuration options. Each time the flow is saved, the Save indicator icon is updated.

Option	Description
Change the flow name, description, or roles	In the main header, select **Properties,** en ter the values you want into the appropriate fields, then select **Update**. **Note:** You can't change the application scope of a flow after you've saved it.
To edit the trigger	In your flow, select the trigger description, fill in the fields as desired, then select **Done**. **Note:** Modifying triggers can result in the dele tion of referenced action configurations.
To edit an existing action	In your flow, select the action description, fill in the fields as desired, then select **Done**.
To add a new action	To add an action at the end of a flow, select the plus icon in the ACTION section. Proceed as you would for adding an action to a new flow. To insert an action into an existing flow, point to the vertical line between the action icons where you want to insert the action. When the plus icon appears, select it. Add the action just as you would add it to a new flow. **Important:** Workflow Studio displays an asterisks character beside any action, flow logic, or subflow that is missing any required field values. Open the action, flow logic, or subflow to add the required field val ues.
To undo the last edit	Select **Undo last action** to revert your last change.  Workflow Studio stores your last 20 configu ration changes. You can't undo changes that create records. For example, converting ac tions into a subflow creates a subflow, and therefore can't be undone. You can select **Un do last action** multiple times to revert multiple changes.

Option	Description
	Important: The undo option is only available during your current user session. Closing the flow tab or the Workflow Studio browser tab ends your current user session and clears out your undo history.
To redo the last undo	Select **Redo last action** to reapply the last reverted change. Workflow Studio stores your last 20 configuration changes. You can select **Redo last action** multiple times to reapply multiple changes. **Important:** The redo option is only available during your current user session. Closing the flow tab or the Workflow Studio browser tab ends your current user session and clears out your redo history.

4. To save your changes, close the flow tab.
   Workflow Studio automatically saves changes as you add and edit items. It also saves when you test or activate a flow.

Delete a flow

Delete a flow that you no longer need.

Before you begin

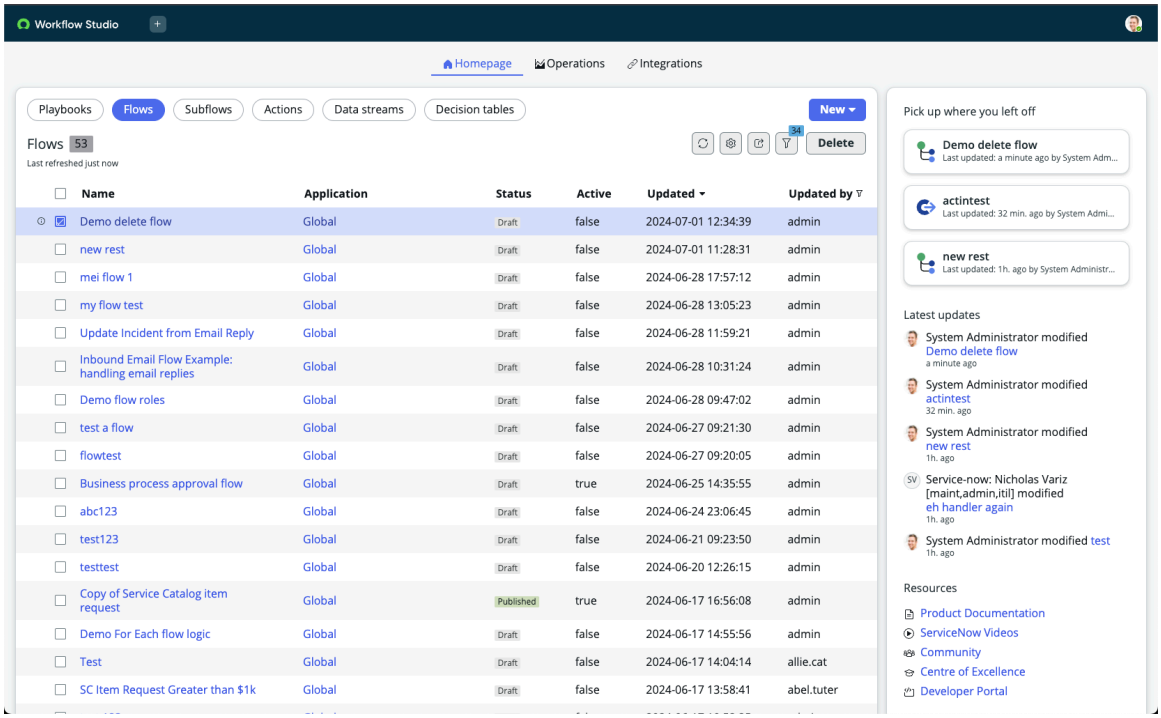
Role required: flow_designer or admin

About this task

You can only delete flows that are in the same application scope as the current session. You can't delete flows that are read-only protected. If your instance uses domain separation, make sure that you are in the Global scope. For more information, see Domain separation and Workflow Studio.

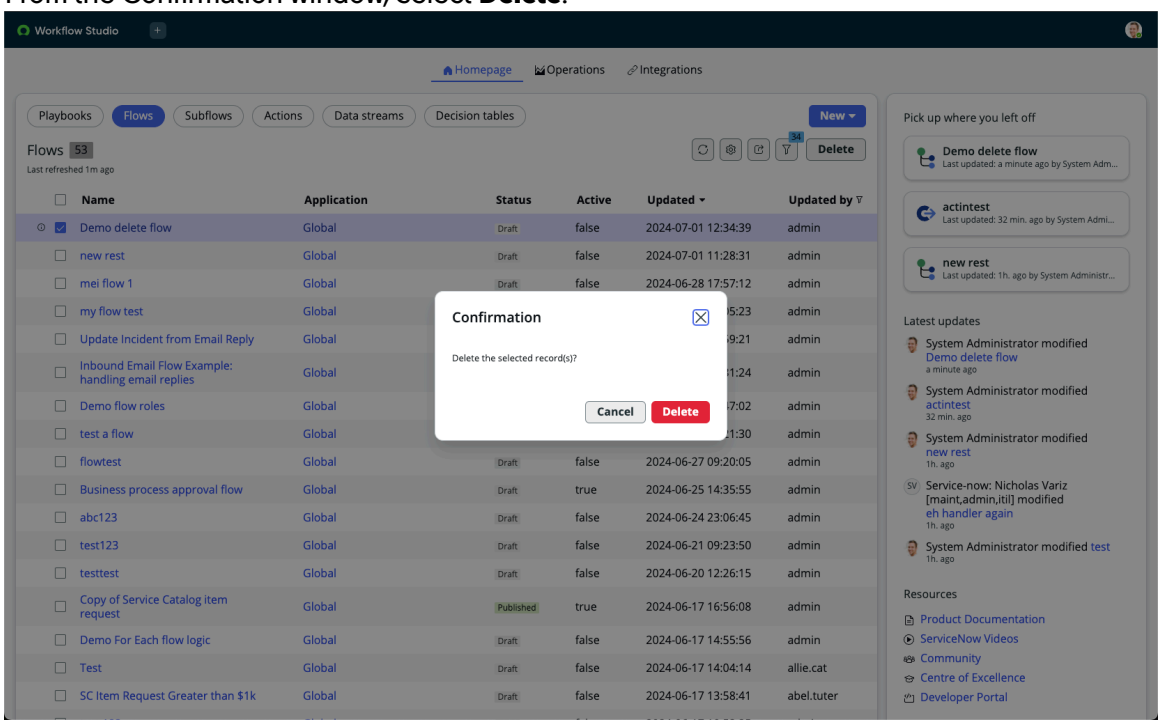
Procedure

1. **Optional:** If needed, change application scope to that of the flow you want to delete.
   For more information about changing application scope, see Exploring Next Experience pickers.
2. Navigate to **All** > **Process Automation** > **Workflow Studio**.
3. On the homepage, select **Flows**.
4. From the list of flows, select the box next to the flow you want to delete.



Tip:
You can filter the list of flows by name or application. For example, selecting the Filter icon on the **Name** column, selecting the **contains** operator, entering the string `demo`, and selecting **Apply** displays all flows that have the string demo in the name.

5. Select **Delete**.
   If the delete option is unavailable, you may be in the wrong application scope, or the flow may be read-only protected.
   Workflow Studio displays a confirmation window.
6. From the Confirmation window, select **Delete**.



Result
Workflow Studio deletes the flow and removes it from the list of available flows.

View activated flows for a table

View flows with record-based triggers that run on a specific table.

Before you begin
Role required: admin or flow_designer and any roles required to view the data table.

Procedure

1. Navigate to the list view of the data table for which you want to view activated record-based flows.

   Example
   For example, navigate to `incident_list.do` to see a list of Incident records.
2. Right-click any column heading from the list view, and select **Configure** > **Flow Designer Flows**.
   The Flows table opens and displays all record-based flows that have been activated to run on the specified table.

Flow and subflow stages

Communicate the current stage of a request, flow, or subflow with an end user.

When configuring stages in Workflow Studio, you can:

- Add stages to a flow or subflow
- Change stage labels and names.
- Configure the estimated duration for a stage.
- Import a copy of a pre-defined stage set from the Stage Sets table. To learn more about stage sets, see Workflow stage sets. Any changes made to the copy do not affect the original stage set record.

You can view the stages of a flow or subflow from its execution details.

Displaying stages in a stage field

A stage field stores and displays the stage state and details about a specific record as a flow or subflow runs. For example, the Service Catalog table uses the **Stage** field to indicate the progress of a request as it is processed.

Stage fields display:

- Stage details for the specific record that triggered the flow or that was used as a subflow input. For example, the current state of a specific Service Catalog requested item.
- Stage names and states defined in the associated flow or subflow. If the associated flow calls another flow, stages set on the child flow do not display.
- Stage details from flows or subflows that have started. If a flow or subflow is not running for a record, that record will not have any stage field details.

State icons in stage fields cannot be modified. Limit the number of stages and the length of each stage name to prevent wrapping text and icons onto multiple lines.

Note:
Only add one stage field per table. If there is more than one stage field, the system only displays stages from the first stage field defined in the table dictionary entry.

Stage field and trigger types

Associating a flow to a stage field depends on the flow trigger type.

Flow trigger type	Requirements
Record	For a stage field to report stages on a record-based flow, a stage field must be present on the same table as the triggering record. When a flow has stages, Workflow Studio communicates the status of each stage back to the triggering table and displays the current stage state as an icon. If more than one stage field exists on the table, only the first stage field defined in the table's dictionary definition is used. **Note:** Avoid creating stages for multiple flows that trigger from the same table. A stage field only displays the stages of the final flow to run. Add different conditions to each flow to ensure that the stages of one flow do not overwrite another flow.
Service Catalog	If using the Service Catalog trigger, the flow must be associated with the Service Catalog item through the **Flow** field. Remove any workflows associated with the item by clearing the **Workflow** and **Execution Plan** fields. The **Stage** field displays the current stage state on any list view of the Requested Items [sc_req_item] table.

Note:
While you can add stages to a flow that has a scheduled trigger, the stages are never displayed to an end user because there is no associated trigger record for the stage field. Only add stages to flows and subflows that have a trigger or input record.

Stage states

During flow or subflow execution, each stage can be in one of six states.

State	Description
Pending	This stage has not yet started.
In progress	This stage is executing.
Skipped	This stage was skipped and did not run. Typically, this state is reached when a conditional flow logic block is not executed.
Complete	This stage is complete.
Cancelled	This stage was cancelled.
Error	This stage has reached an error condition.

State	Description
	When designing a flow or subflow, you can manually set its to report an Error state. To set an Error state: • The flow or subflow must have at least one stage defined. • The Error can only be set within a stage. When an Error condition is reached, the current stage is set to Error. • The Error can only be set within a conditional flow logic block.

Each stage can have its own custom state labels. For example, suppose that you have a flow with two stages. Stage 1 could have the Pending state with the label Waiting, and Stage 2 could have a Pending state with a label of Not yet started. Workflow Studio provides options to generate either the default states or approval states.

System properties

You can use these system properties to configure how a flow or subflow displays approval details.

com.glide.hub.flow_engine.stage_display.show_approvers

Show or hide the list of approvers assigned to a stage from a stage field. Set the value to true to show the list of approvers assigned to a stage. Set the value to false to hide the list of approvers assigned to a stage.

- Type: true | false
- Default value: true
- Location: Add to the System Properties [sys_properties] table

com.glide.hub.flow_engine.stage_display.show_approvers_limit

Specify the maximum number of approvers to display in a stage field as an integer value. Setting this value above 10 risks causing rendering errors in a list view. The stage field for one record can become so big that the list cannot display additional records in the list.

- Type: integer
- Default value: 5
- Location: Add to the System Properties [sys_properties] table

General guidelines

Follow these general guidelines when creating flows or subflows with stages.

Avoid defining stages that depend on a For Each flow logic

Flow Designer prevents you from adding stages within a **For Each** block. You can only add stages before or after a **For Each** block.

Avoid creating stages for the same records in different flows or subflows

A stage field always displays the stage information provided by the last flow or subflow to run on a table's record. If multiple flows or subflows run on the same records, then the stages defined in one flow or subflow can in theory overwrite the stages from another flow or subflow. To avoid multiple flows or subflows overwriting each other's stages, define unique trigger or start conditions for each flow or subflow.

Avoid updating stage fields from outside a flow or subflow

If you manage stages with a flow or subflow, avoid directly updating record stage fields from outside the flow or subflow. Manually updating the value of a stage field may produce unexpected or undesired results.

Ensure that each flow on a table has unique trigger conditions

Adding unique trigger conditions to each flow ensures the flows only run under those conditions and prevents the stages from one flow overwriting the stages of another flow. Specifying unique trigger conditions makes it easier to troubleshoot flows by limiting the number of flow executions that can produce record changes.

Use error stages to communicate with the user

The flow error state does not affect flow execution. A flow continues running even if it reaches an error stage. Use a conditional flow logic block to set the error stage and communicate to the user that the state of the current stage is Error. For example, if an approval is not approved within the required limit, you may want to communicate an error to the user.

Use the error stage to stop processing a flow

Use a conditional flow logic block to identify when a flow enters the error stage. Use the flow logic to stop processing the flow or take some kind of remediation action. For example, you may want to change the record state or assignment when a flow reaches an error state.

Configure stages and add them to a flow

Configure when stages display to a user, define stage state labels, and add stages to a flow or subflows within Workflow Studio.

Before you begin

This task assumes that you are familiar with flow and subflow stages, stage fields, and stage sets. For more information about stages, see Flow and subflow stages.

Role required: flow_designer or admin

Procedure

1. **Optional:** If not already present, create a stage field on the table that triggers the flow.

   (Optional) A stage field stores and displays the stage state and details about a specific record as a flow or subflow runs. The Service Catalog Requested Items [sc_req_item] table has a stage field by default.

   For a stage field to report stages on a record-based flow, a stage field must be present on the same table as the triggering record.

   To add a field to a table, see Add and customize a field in a table.

2. **Optional:** If adding stages to a flow with a Service Catalog trigger, select the flow in the **Flow** field of the Service Catalog Item [sc_cat_item] table.
   If using the Service Catalog trigger, the flow must be associated with the Service Catalog item through the **Flow** field. Remove any workflows associated with the item by clearing the **Workflow** and **Execution Plan** fields. This enables a request for a catalog item to initiate a flow specific to that catalog item.
3. Open the flow or subflow in Workflow Studio.

Note:
While you can add stages to a flow that has a scheduled trigger, the stages are never displayed to an end user because there is no associated trigger record for the stage field. Only add stages to flows and subflows that have a trigger or input record.

4. Create and configure stages.

   a. Click ... and select **Flow stages**.
   The Flow stages menu opens.

   b. **Optional:** From **Add stages from a template**, select a stage set to import existing stages from the Stage Sets table, and select **Add stages**.
   To learn more about stage sets, see Workflow stage sets. Any changes made to the copy do not affect the original stage set record.

   In general, use stage sets when you want to reuse stages across multiple flows or subflows.

   c. Click **Add new stage** to manually create and configure stage options inline.

Field	Description
Name	Display name for the stage. Must be unique.
Value	Internal name for the stage. Must be unique.
Duration	Estimated duration displayed to the user. If you want to update stages based on flow run times, consider adding one or more Wait for a duration flow logic blocks. For more information, see Wait for a duration flow logic.
Always Show	If selected, the stage always displays in the stage field, even if the stage is set within a flow logic block that might not execute.

   In general, create stages inline when you do not plan to reuse stages in other flows or subflows.

   d. Order stages from top to bottom in the order you want the stage field to display them.
   The stage at the top appears first in the stage field, and the stage at the bottom appears last in the stage field.

   Note:
   If you create stages inline, the Flow Stages menu displays them in the order they were created, not in the order they appear in the flow.

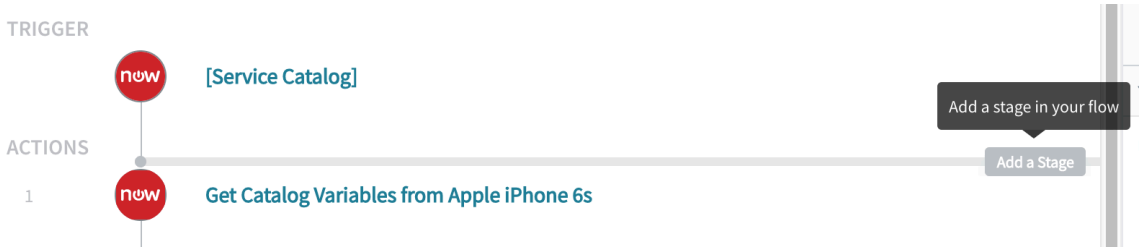
5. Add stages to the flow or subflow.

   a. Point to a location in the flow or subflow.

   b. Click **Add a Stage**.

   c. Select an existing stage.
   Stages can be applied at the beginning of any Workflow Studio action or flow logic block, or within an If block.

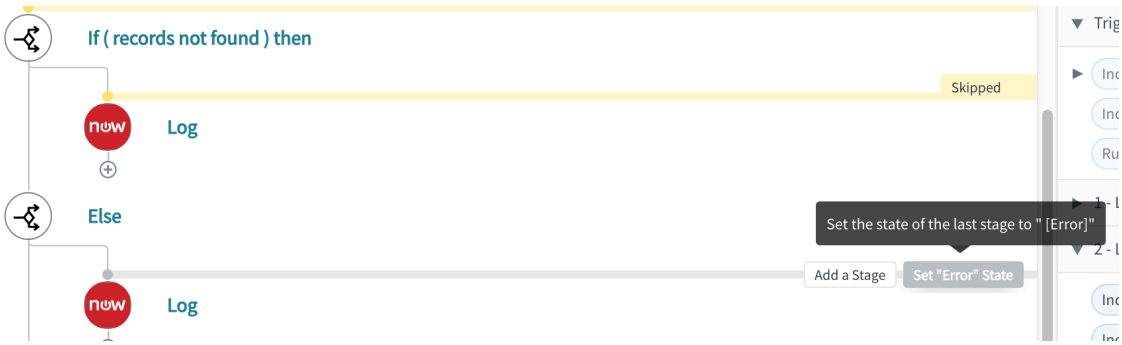
Important:
Stages are unavailable from within a **For Each** flow logic block. You can only add stages before and after a **For Each** flow logic block.



6. **Optional:** Create stages inline.
   a. Point to a location in the flow or subflow.
   b. Select **Add a Stage**.
   c. Select **+**.

> **Important:**
> When you create stages inline, stage fields display the stages in the order they were created, not in the order they appear in the flow. To configure or change the order of stages you create inline, open the More Actions menu ⋯.

7. **Optional:** Set a stage to the Error state within a conditional flow logic block.
   a. Point to a location.
   b. Select **Set "Error" State**.
   Note the following requirements to set a stage to the Error stage:
   - The flow or subflow must have at least one stage defined.
   - The Error can only be set within a stage. When an Error condition is reached, the current stage is set to Error.
   - The Error can only be set within a conditional flow logic block.



When the flow reaches the indicated point, the currently executing stage is set to Error in the stage field.

Result

When the flow runs, the stage details appear in any field of type Workflow. In a flow with a record-based trigger, the Workflow field of the triggering table displays the current stage state. In a flow with a Service Catalog trigger, the **Stage** field of the Requested Items [sc_req_item] table displays the current stage state.

Flow error handler

Enable flows to catch errors. Run a sequence of actions and subflows to identify and correct issues. For example, have flows log output values, send notifications, and run corrective subflows when they produce an error.

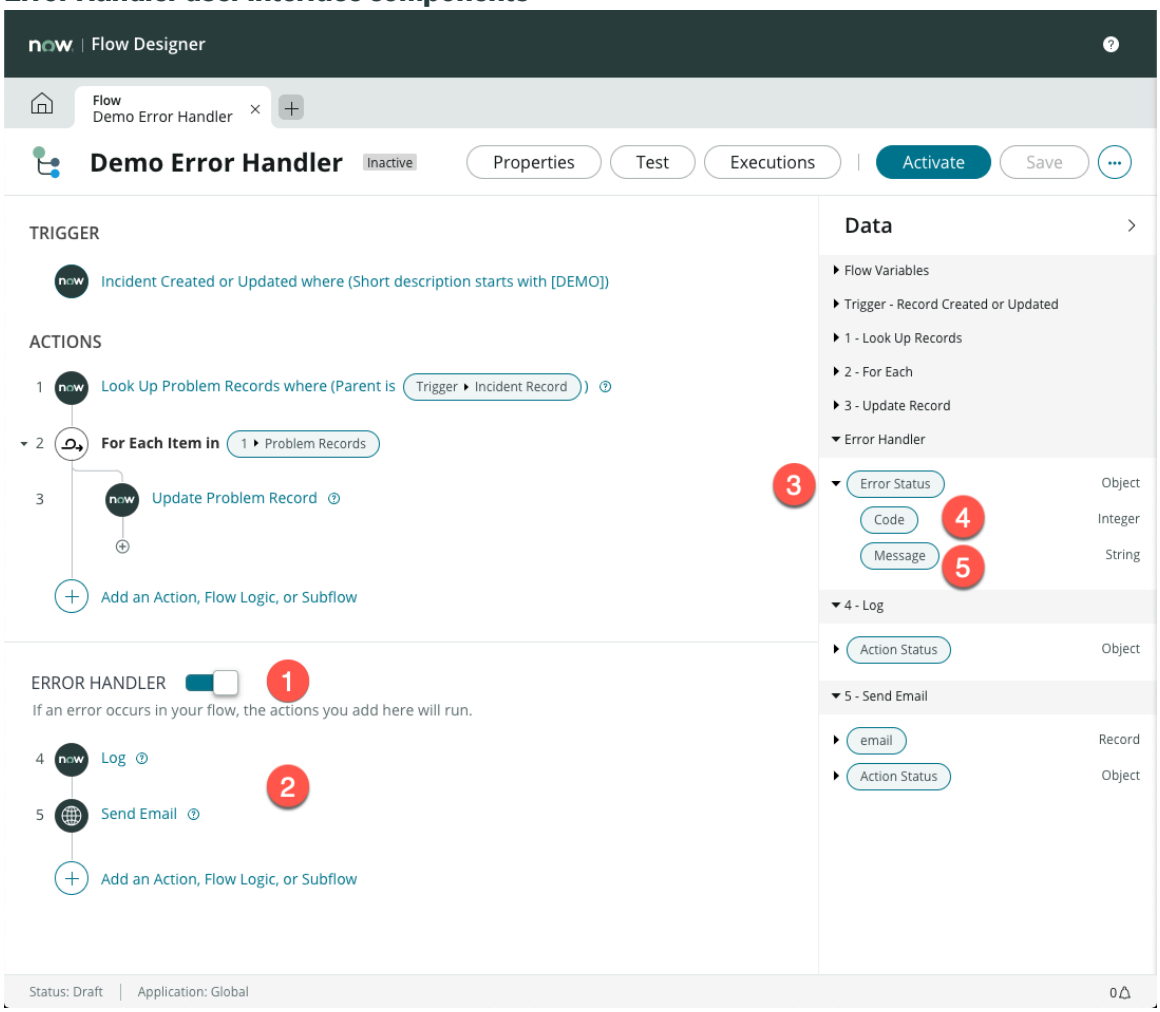
Benefits

Enable a flow error handler to gain these benefits.

- Automate the identification of flow errors as they happen. Capture and push flow error information rather than manually search for issues caused by flow errors.
- Automate the resolution of flow errors. Run actions and subflows to update records rather than manually updating records affected by flow errors.
- Build your own action error handling logic. Specify when actions return an error rather than always return an error.

Error Handler components

Error Handler user interface components



Flow error handling consists of these components.

1. **Error Handler switch**

   Option to enable or disable flow error handling. When enabled, the flow displays the Error Handler section.

2. **Error Handler section**

Section of flow that runs when the flow catches an error. Use this section to automate the identification and resolution of flow errors. You can add up to 10 items in this section.

Note:
The 10-item limit includes any combination of actions, flow logic, or subflows.

3. Error Status

Object data pill containing details about the error caught by the flow.

4. Error Status > Code

Integer data pill indicating whether the flow produced an error. By default, a value of 1 indicates that the flow produced an error. A value of 0 indicates that the flow ran successfully. You can define your own error codes when you create a custom action.

5. Error Status > Message

String data pill containing the error message produced by the action, step, or system operation.

Available error states

Enabling the Error Handler changes the states reported in the flow execution details. The Error Handler can produce these flow states.

Completed (error caught)

State generated when the flow caught an error and successfully ran the items in the Event Handler section. The flow generates this state even when the Event Handler section is empty. This state is only available when you enable a flow Error Handler. This state is only visible from a flow execution details page. Flow context records instead display the state as **Complete**.

Completed (error skipped)

State generated when a custom action continues running after a step failure. When an action generates this state, it passes it to the parent flow. This state is only available when you enable a flow Error Handler. This state is only visible from a flow execution details page. Flow context records instead display the state as **Complete**.

Error

State generated when an error remains uncaught.

- An error occurs in the flow while the Error Handler is disabled
- An error occurs in the Error Handler section

When an error occurs

When an error occurs in a flow with an active error handler, the flow stops running further actions and flow logic and instead runs the items in the error handler section. If the items in the error handler section run, the flow stops with the Completed (error caught) state. If the error handler itself generates an error, the flow stops with the Error state.

A flow cannot rerun items that generate an error or resume from the step that generated an error. You can use Try flow logic to continue running a flow that encounters an errors within the try block.

Flow and action error handling resources

For more information about using error handling in actions and flows, see the ServiceNow® Community post Flow and Action Error Handling Overview: Why and how to test for errors - Worfklow Automation CoE.

- Flow and Action Error Handling Level 1: Retry and Action Error Evaluation - Workflow Automation CoE
- Flow and Action Error Handling Level 2: Flow Logic - Workflow Automation CoE
- Flow and Action Error Handling Level 3: Flow Error Handling - Workflow Automation CoE
- Flow and Action Error Handling Level 4: Good Practices and Summary - Workflow Automation CoE

General guidelines

Follow these general guidelines to achieve the benefits offered by flow error handling.

Avoid adding error handling items to the main section of the flow

A flow normally stops running when an action or subflow returns an error in the main section. A stopped flow cannot run any actions or subflows past the point where it returned an error. Adding error handling actions and subflows to the Error Handler section ensures they run them when there is an error.

Capture Error Status information

The Error Status object contains information about the action that produced an error. You can use this information to identify the cause of the error as well as record data that may need correction.

Suppress subflow error messages

You can enable the Error Handler for a subflow to prevent its errors from cascading to a parent flow. Leaving the subflow Error Handler section empty ensures that it always generates the **Completed (error caught)** state.

Use subflows to avoid the 10-item limit

Rather than force your error-handling-process to fit within a 10-item limit, call subflows, which can contain many more items. You can also use the subflow outputs to trigger automation in other flows.

Use subflows to take corrective actions

Rather than recreate the same sequence of actions in multiple flows, create reusable subflows to correct errors to your record data. When a flow error leaves your record data in an undesired state, use subflows to correct these records. You can use the error handler to identify such record data as a subflow output.

Related topics

Action error evaluation

Add an error handler to a flow

Run a sequence of Workflow Studio actions and subflows to identify and correct issues that are caused by flow errors.

Before you begin

Role required: flow_designer or admin

About this task

A flow error handler allows your flow to catch an error and run a set of actions, flow logic options, and subflows.

Important:
A flow error handler cannot resume or restart a flow that produces an error.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Create a flow or open an existing flow.
   For more information about creating a flow, see Create a flow in Workflow Studio.
3. Enable the ERROR HANDLER switch.
   Workflow Studio adds an error handler section to the flow and the Data pane.
4. Add actions, flow logic options, or subflows to the Error Handler section.
   You can add up to 10 actions, flow logic options, or subflows to this section. Consider using a subflow to capture the error information or to correct issues with the record data.

   You can add flow logic options to the error handler. These options are described in the following table.

   Supported flow logic options

Flow logic option	Description
If	Selectively apply one or more actions only when a list of conditions is met. For more information, see If.
Wait for a duration of time	Use this flow logic option to pause the flow for a specified time period and resume the flow execution after the time period elapses. For more information, see Wait for a duration of time.
End Flow	Use this flow logic option to stop a flow within Workflow Studio. For more information, see End Flow.
Dynamic Flow	Identify and run a flow or subflow dynamically by using runtime data. Build templates to provide expected inputs for dynamically called flows or subflows. For more information, see Dynamic Flow.
Set Flow Variables	Assign a value to one or more flow variables. Change or update a variable's value during a flow. For more information, see Set Flow Variables.

   The Error Status object contains the information about the flow error. You can also use the Action Status object that is returned by each action to build the conditional logic. Both of these objects are available from the Data pane.

5. **Optional:** Add stages in the Error Handler.
   For more information on how to add stages, see Configure stages and add them to a flow.
6. Add a custom action to the main body of the flow that throws an error.
   To learn how to create a custom action, see Create a custom action to throw an error.

7. Test your flow to ensure that the error handler works as expected.
   For more information about testing a flow, see Test your flow.
8. When your flow error handling is working as expected, select the delete icon next to your custom action that throws an error to remove it from your flow.

Result

Your flow runs the actions, flow logic options, and subflows that you specify when the flow produces an error. The flow execution details display the Completed (error caught) state for both the flow and the action that returned an error.

Create a custom action to throw an error

Create an action that intentionally throws an error to test flow error handling.

Before you begin

Role required: flow_designer, action_designer, or admin

About this task

This custom action throws an error when the action input value is set to 1. Any other input value allows the action to run without throwing an error. You can add this custom action to a flow to test flow error handling.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
   The system displays the Workflow Studio landing page.
2. Select **New** > **Action**
   The system displays the Action Properties dialog.
3. Enter these sample values.

Field	Value
Name	Throw an error
Application	Global
Accessible From	All application scopes

4. Select **Submit**.
   The system displays the Workflow Studio interface.
5. From the Action Outline, select **Inputs** > **Create Input**
   The system displays a new action input.
6. Configure the action input with these values.

Field	Value
Label	Error Code
Type	Integer
Mandatory	True

7. From the Action Outline, select **Add a new step**.
   The system displays a list of available steps.

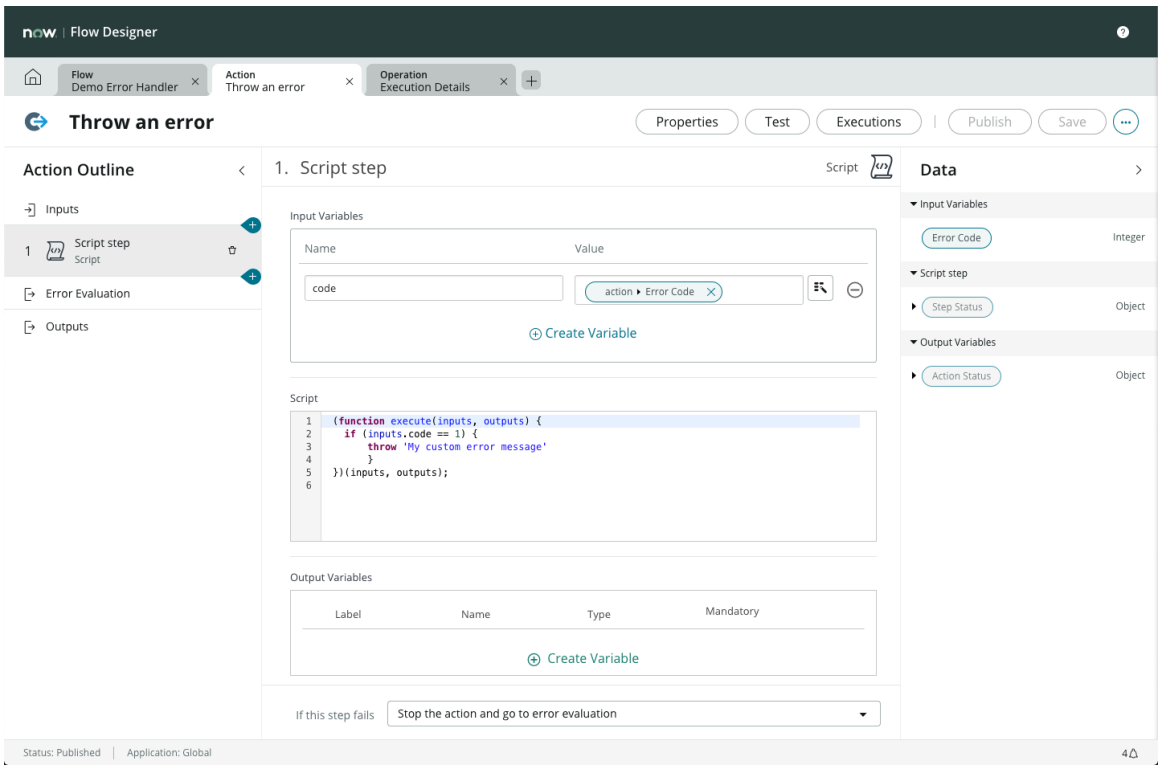
8. Select **Script step**.
9. From the **Input Variables** section, select **Create Variable**.
10. Configure the variable with these values.

Field	Value
Name	code
Value	Select the data pill [action->Error Code]

11. In **Script**, enter this JavaScript code.

```
(function execute(inputs, outputs) {
  if (inputs.code == 1) {
      throw 'My custom error message'
      }
})(inputs, outputs);
```

12. Click **Save**.



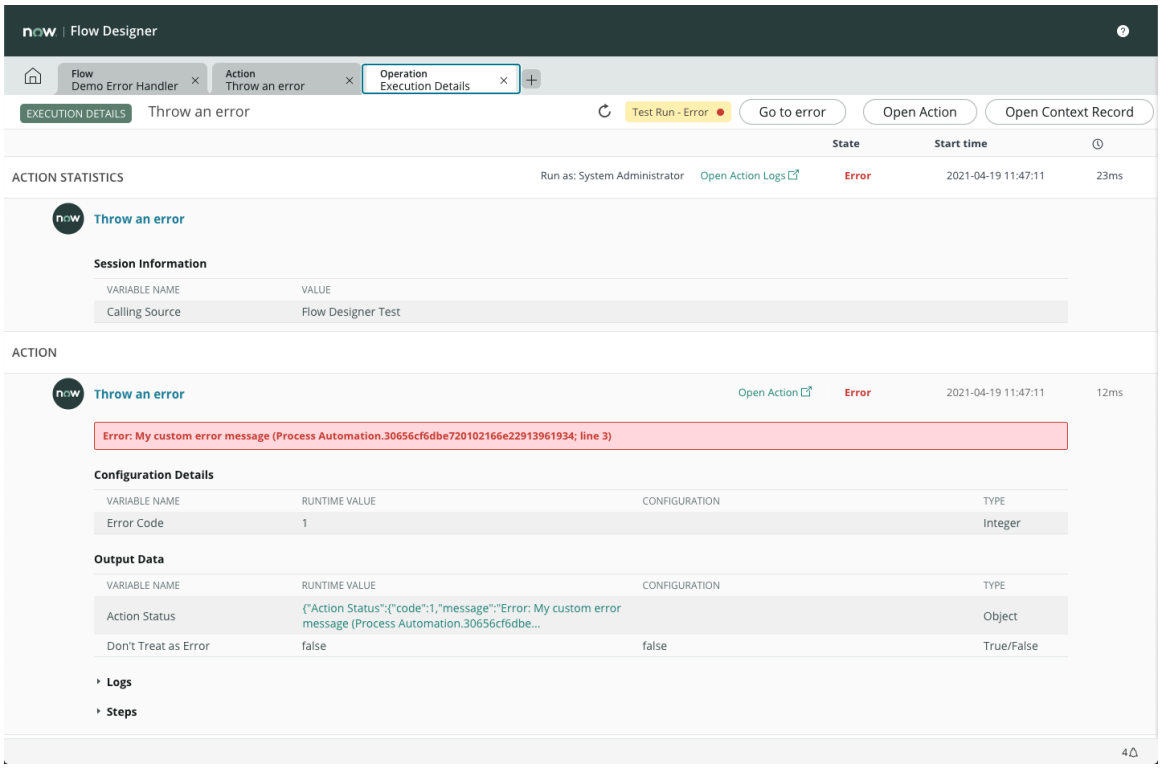
13. Select **Test**.
    The system displays the Test Action dialog.
14. Enter the following test value:

Input	Value
Error Code	1

15. Select **Run Test**.
    The system runs the action with the test values provided.
16. Select **Your test has finished running. View the action execution details.**

The system displays the action execution details.

17. Verify that the action ran the Script step and threw your custom error message.
The **Action Status** object should list an error on line 3 and display the text of your custom error message.



18. Close the action execution details.
19. Select **Cancel** to stop testing the action.
20. Select **Publish** to make your custom action available to your flows.

Result

You have a custom action that throws an error when you set the action input **Error Code** to 1.

What to do next

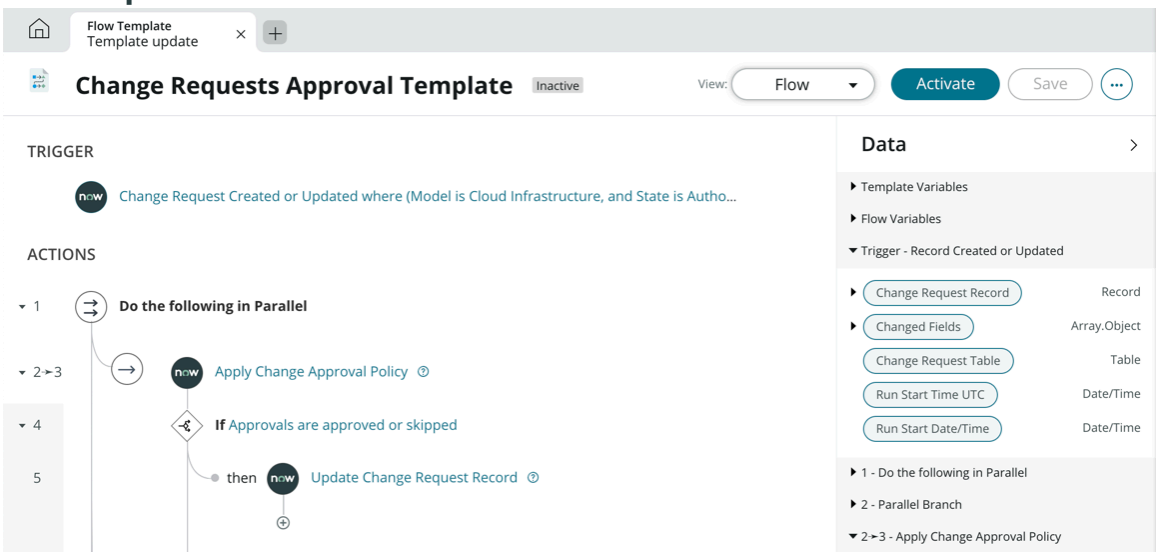
Add this action to a flow to test the contents of the Error Handler section.

Flow Template Builder

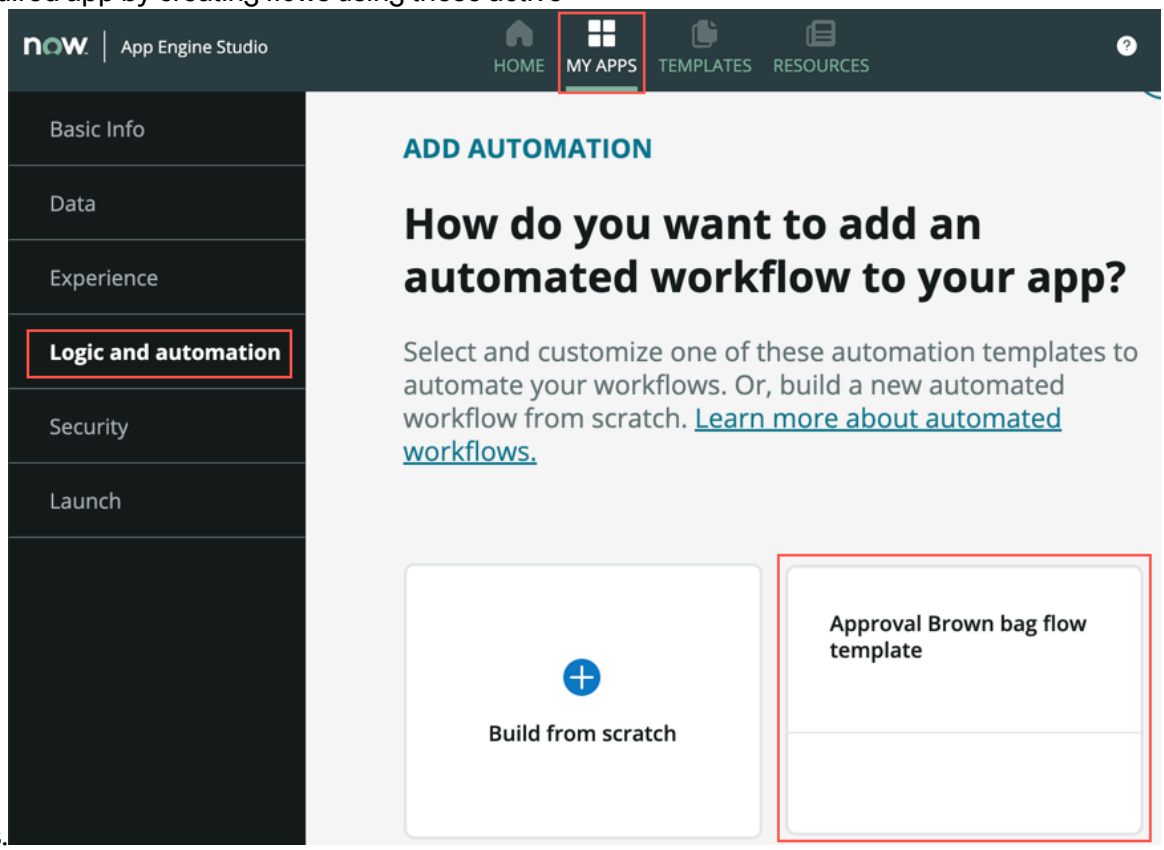
Enable citizen developers to create their own flow templates. Flow Templates guide flow authors to create flows for common use cases. Use the flow template builder to define flows, actions, and flow template variables.

Template authors can create templates from the flow with the required configurations, in Workflow Studio. Template authors can view or edit an existing flow template in Flow Template Builder.

Flow Template Builder UI



In App Engine Studio, administrator can add automations in the required app by creating flows using these active templates.



Create a template using Flow Template Builder

Create template from a flow in Workflow Studio to guide flow authors through the creation of a flow with the same configuration and customized template input values for the components.

Before you begin

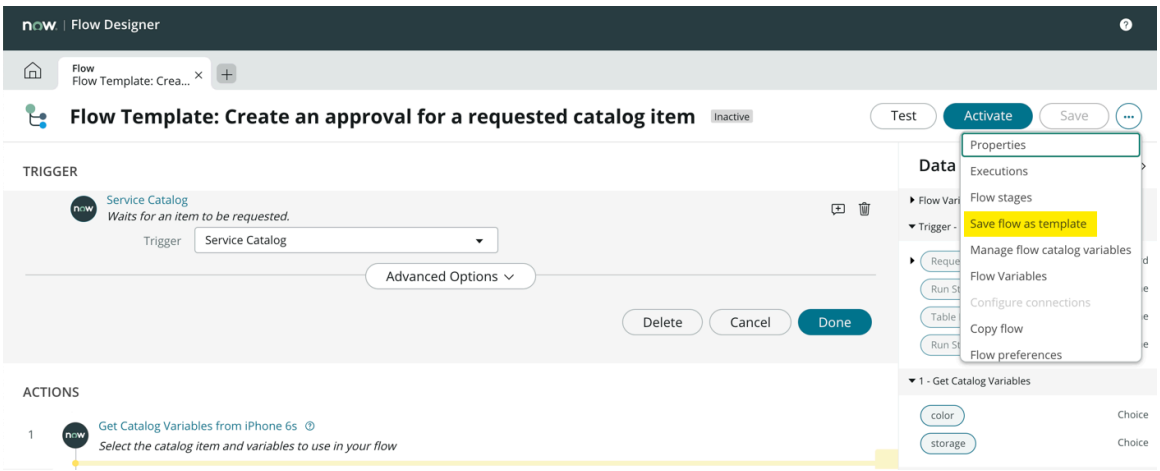
- Activate the App Engine Studio (sn_app_eng_studio) and Flow Template Builder (sn_flow_template) plugins.
- Create a flow in Workflow Studio as per your requirement.

- Role required: App Template Author or admin

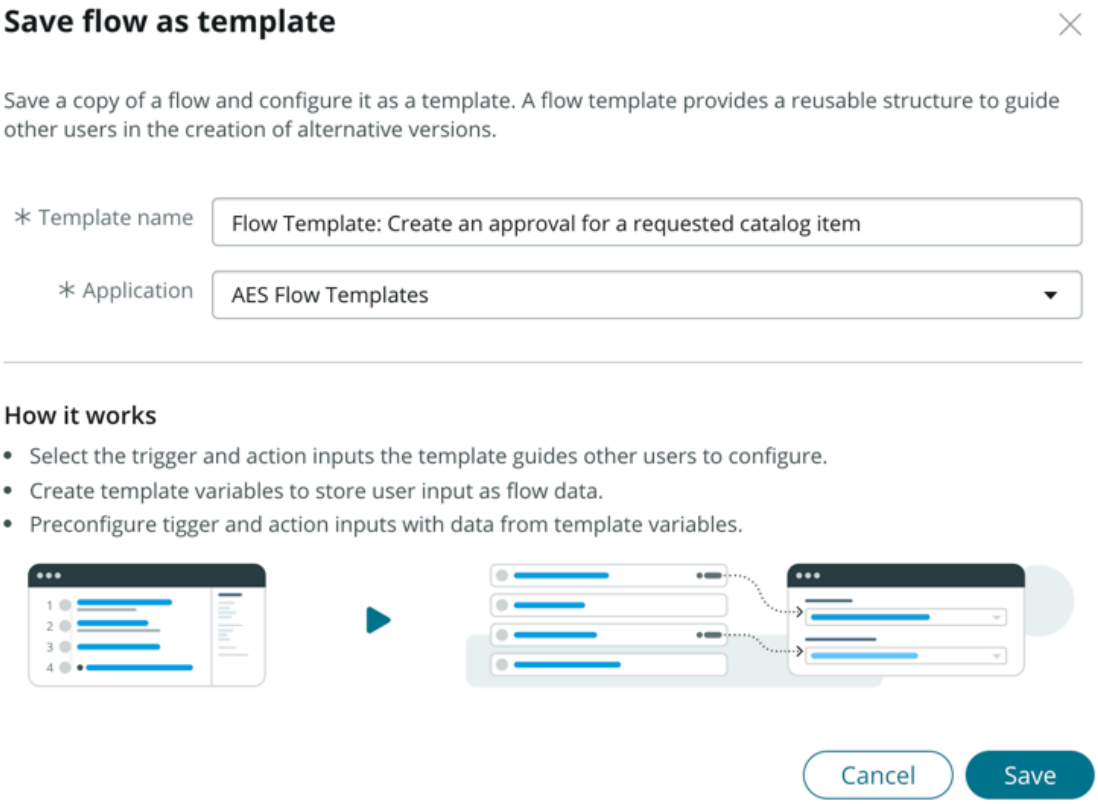
Note:
Users with the App Template Author cannot create flow templates in the **Global** scope. They can create flow template in only those scopes to which they have access.

Procedure

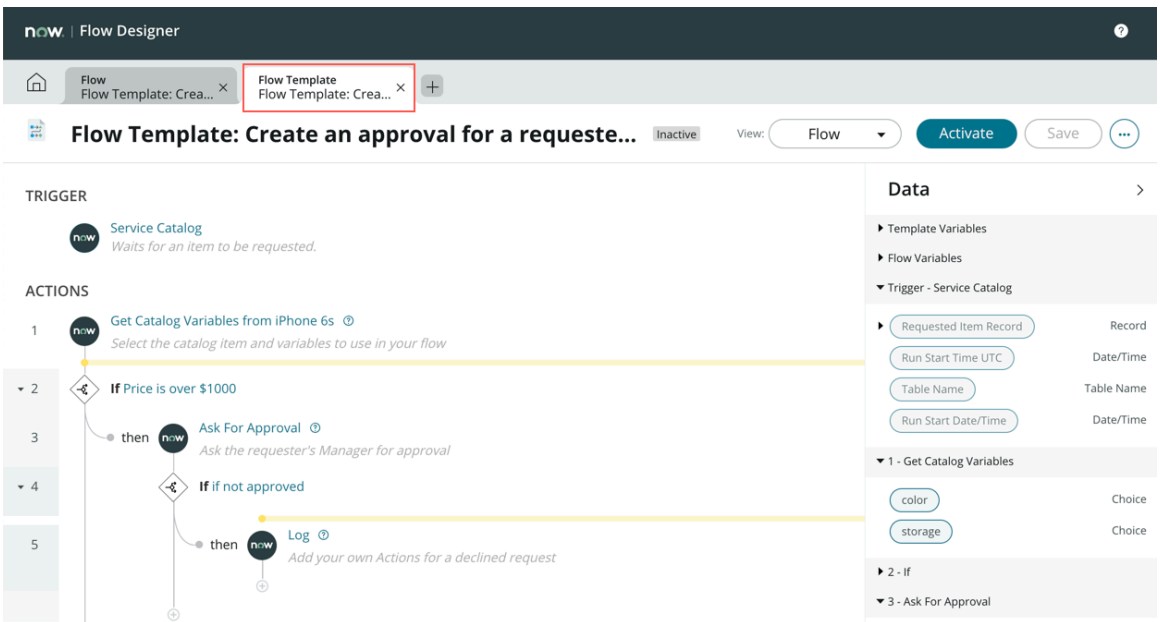
1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. In Workflow Studio, open the required flow.
3. Click the more actions icon ( ) and select **Save flow as a template**.



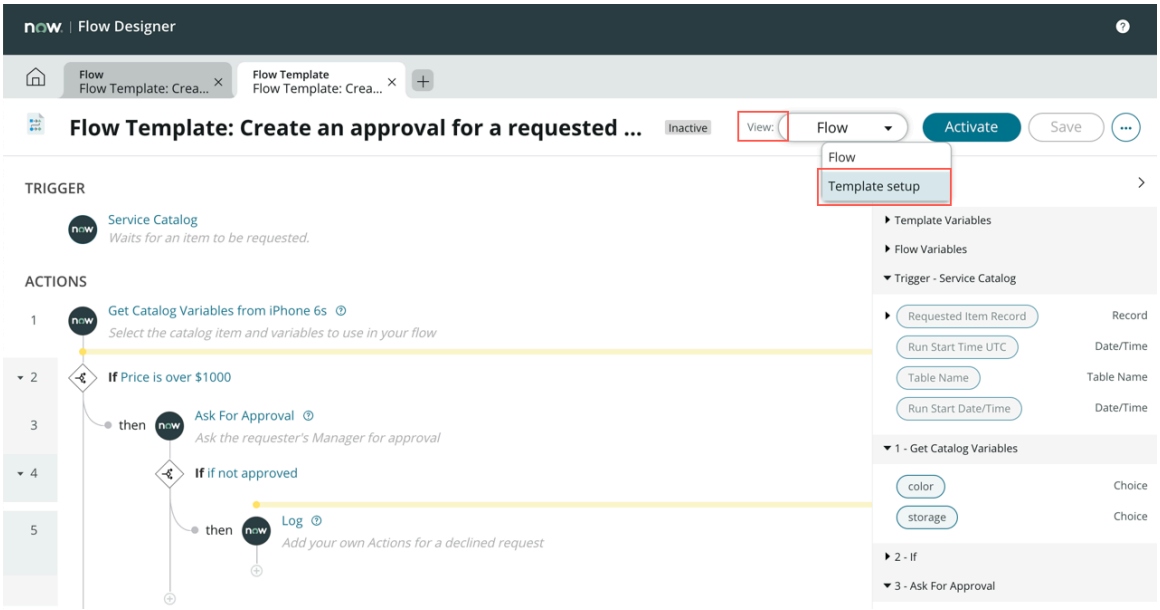
4. In the Save flow as template dialog, enter **Template name** and select **Application** in which you want the template.



5. Click **Save**.
The template is created and is displayed in Flow Template Builder.



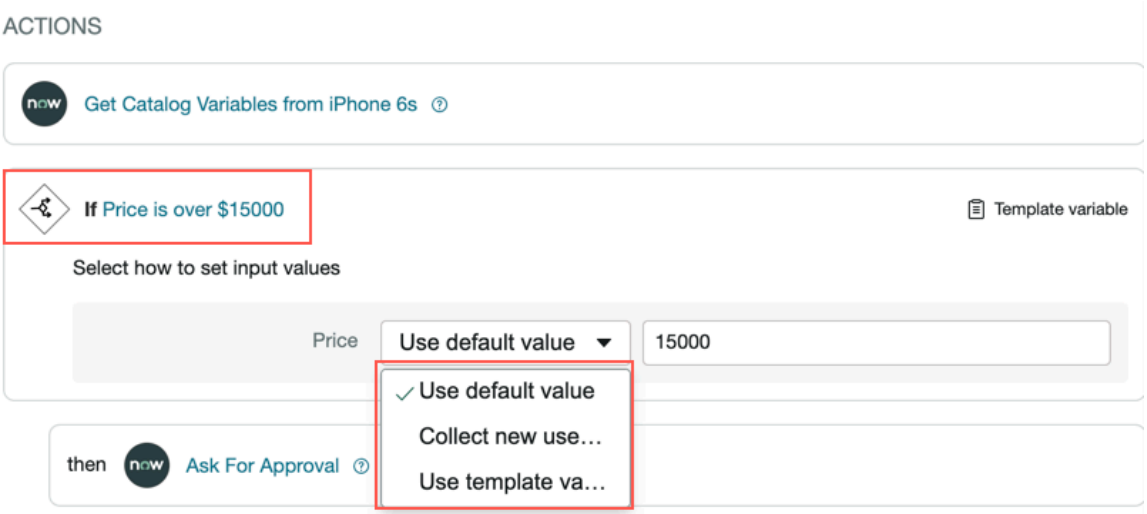
6. In **View**, select **Template setup**.



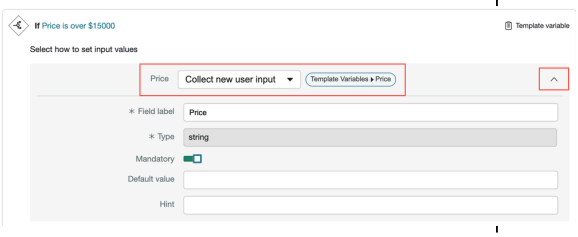
Template is loaded in Flow Template Builder.

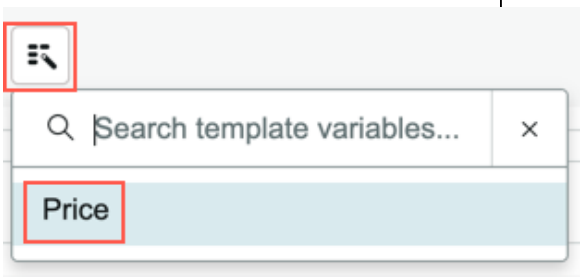
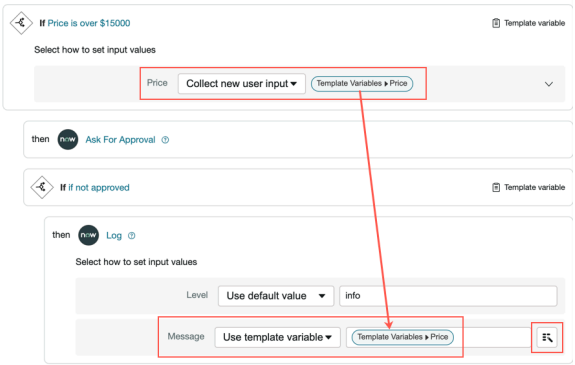
When **View** is changed to **Flow**, the template is displayed in the Workflow Studio UI.

7. Click the required action and select the required inputs.



Choice	Description
Use default value	Uses the default value provided in flow.
Collect new user input	Creates a template variable. Expand the template variable to configure the user input as per your requirement.



Choice	Description
	Note: Input variable once created, cannot be deleted.
Use template variable	Uses the template variable that has been collected in a previous action. Click the data picker to use the previously collected user input.  In this example, **Price** is collected as a user input and this user input is used in the **Message** input of the **Log** action. 

Note:
Supported template variable data types:

- String
- List
- Choice
- Reference
- Table Name
- URL
- Multi Line Small Text Area
- Two Line Text Area
- Price
- Email
- Integer

8. Click **Save**.
   The flow template is created.

   To verify if the template record is created, type `sys_app_template.list` in the left navigator pane and search for the template you had created.

9. Click **Activate**.

Note:

- While configuring the properties in Flow template properties dialog, ensure that you select the **Icon** first before you enter other fields and click **Update**.
- **Icon** in the Flow template properties supports only .SVG files.

Create a flow from a template in App Engine Studio

Create a flow from an existing App Engine Studio automation template. Follow the template guidance to provide values for template inputs that accept dynamic data.

Before you begin

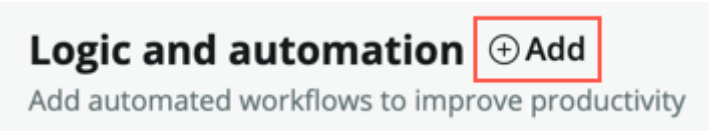
- Create a template using Flow Template Builder and activate it.

Note:
If the template is modified, the template must be activated again for the changes to be reflected in App Engine Studio.

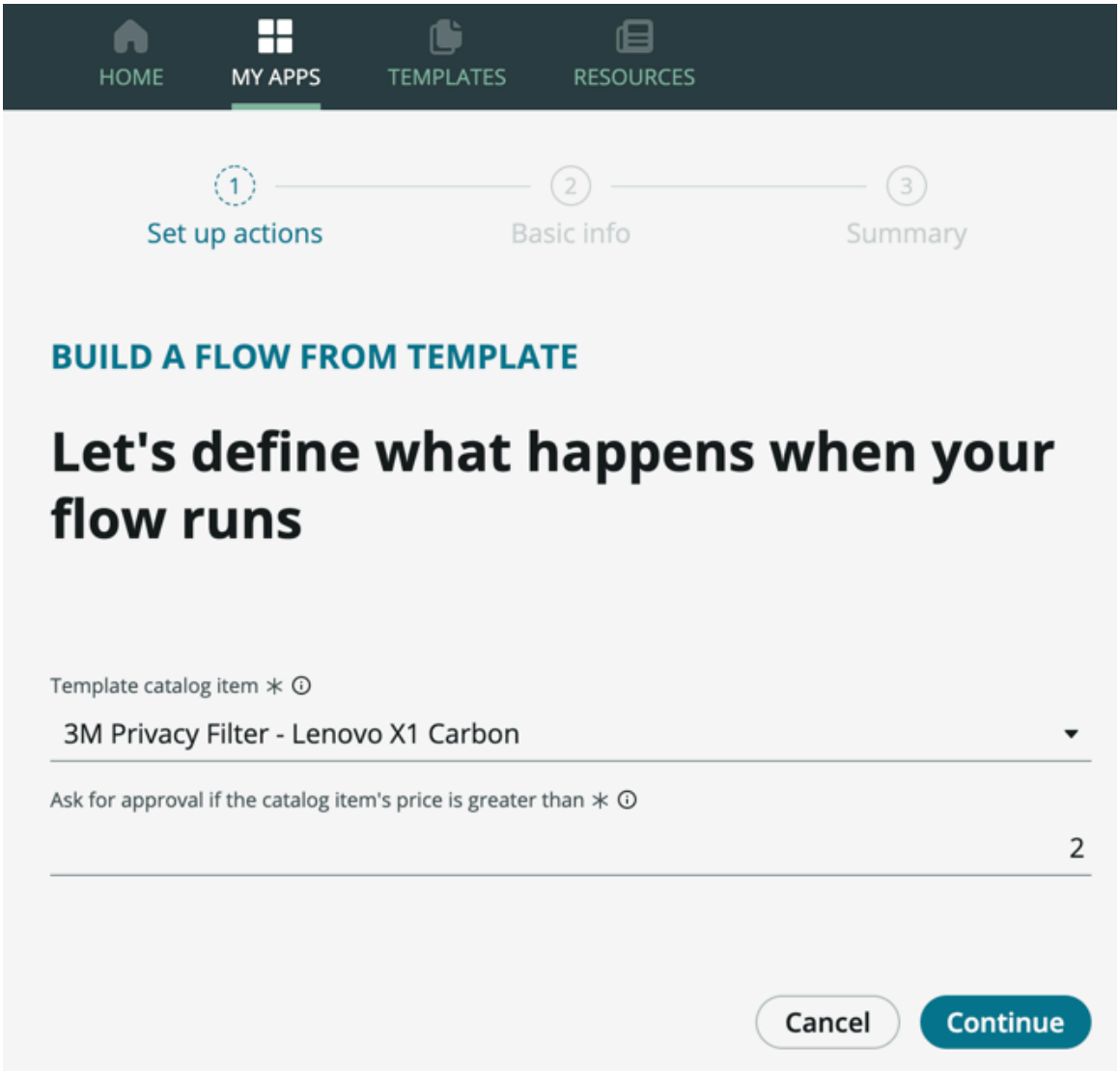
- Role required: Template Runner

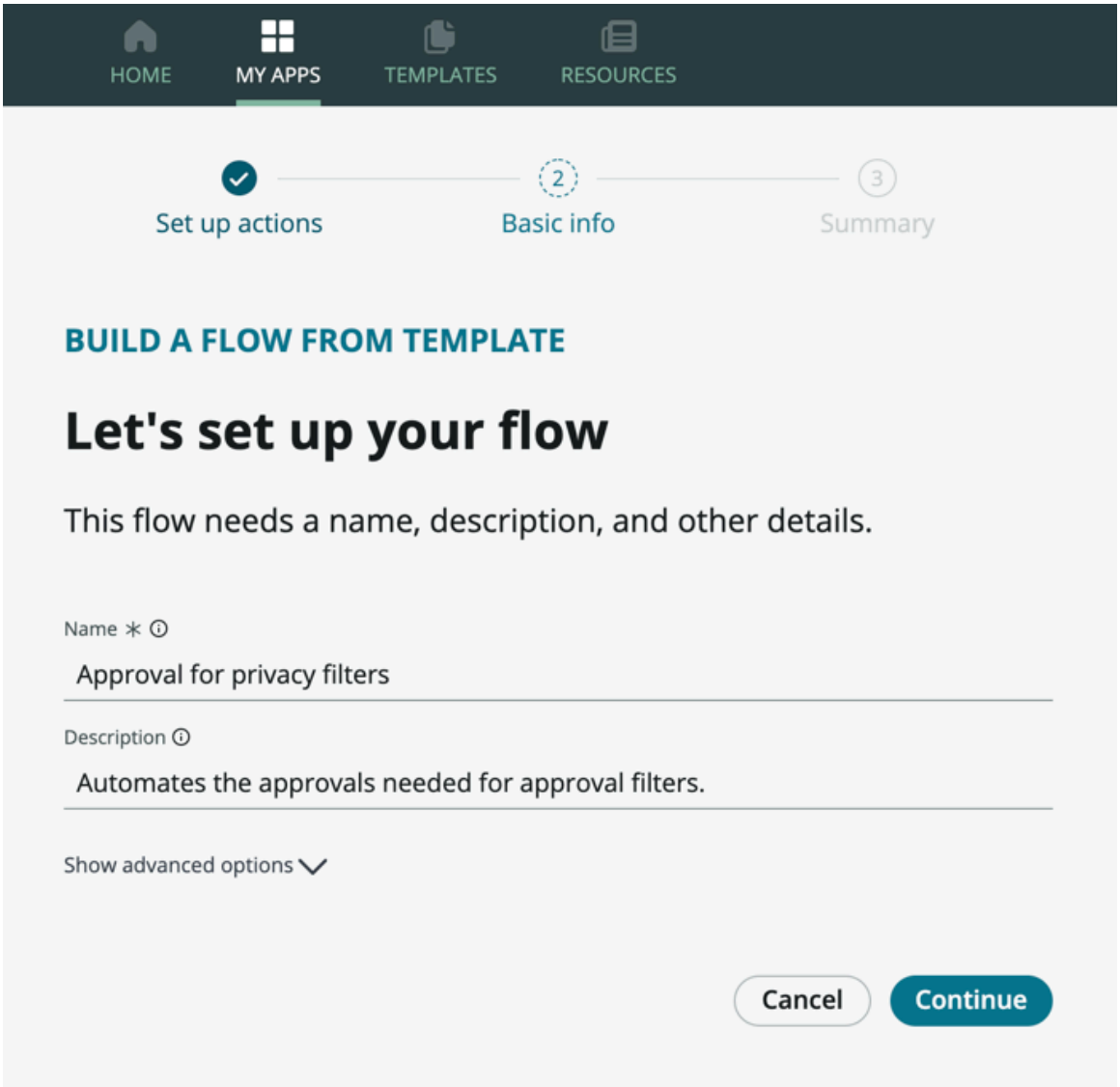
Procedure

1. Navigate to **All** > **App Engine** > **App Engine Studio**.
2. In **My recent apps**, select the required app.
   If you haven't created an app, you can create it.
   The app is opened in App Engine Studio.
3. Click **Logic and automation**.
4. Click **Add**.



5. Select the required flow template.
6. In the template dialog, click **Begin**.
7. In the templates wizard, provide the inputs to create flow using the template.





After providing the required inputs, a confirmation message is displayed that the flow is created.

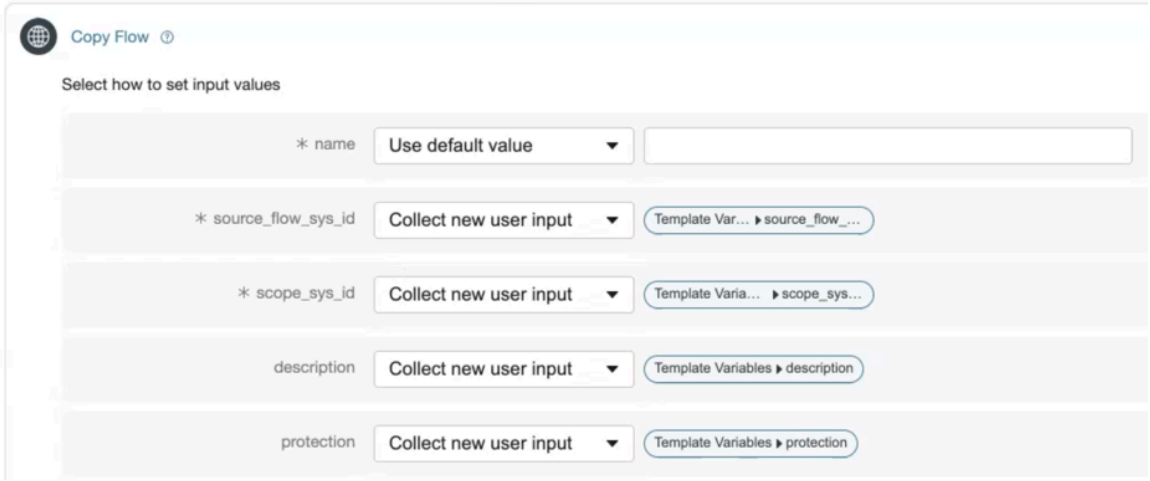
8. To edit the flow in Workflow Studio, click **Edit this flow**.

The flow is opened in Workflow Studio.

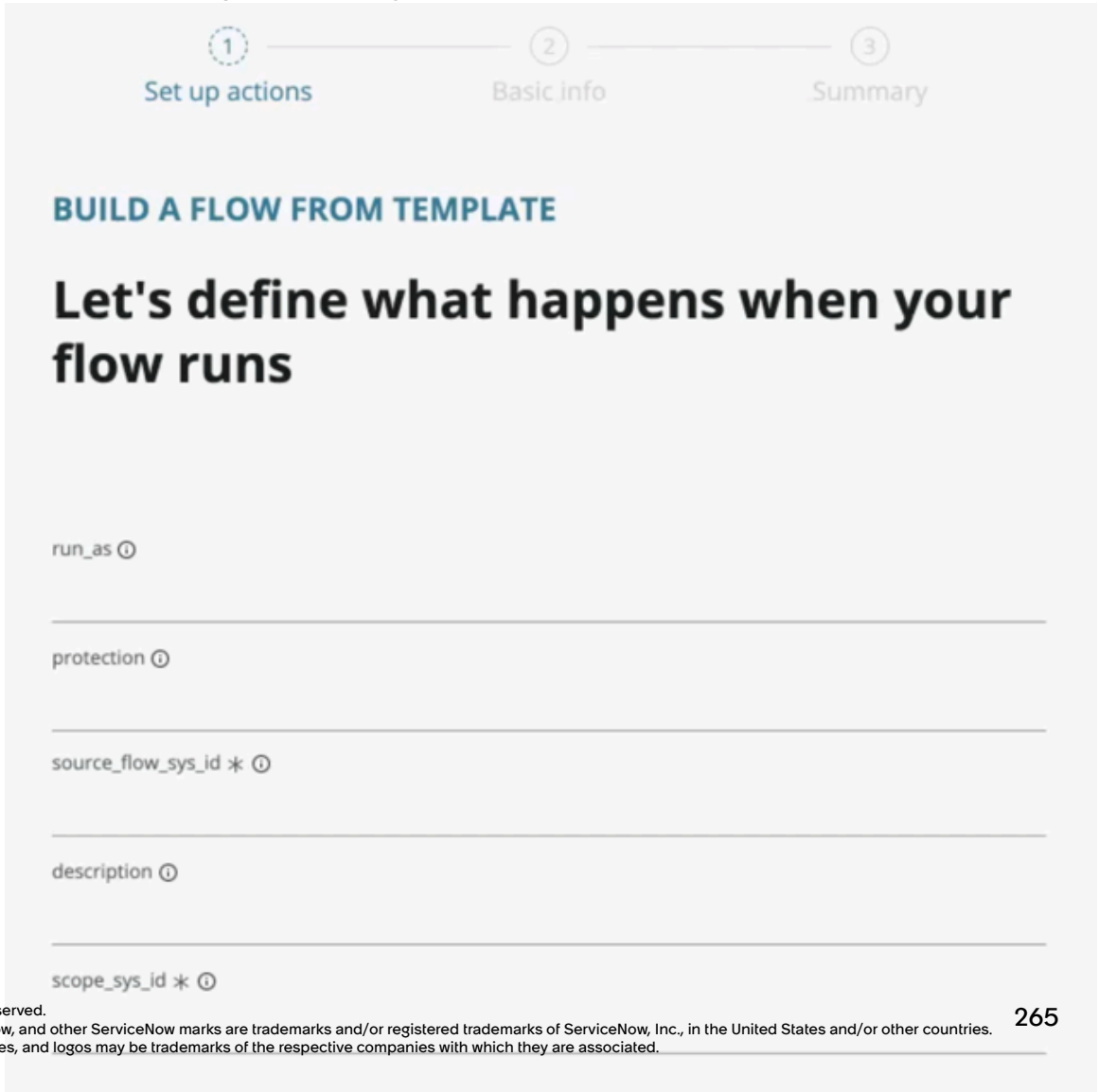
Note:

- Avoid editing flows that are created from a template. If you intend to edit the flow, ensure that you test the flow before publishing it.
- In App Engine Studio, template inputs are not displayed in the same order as you had created in Workflow Studio. In this example, order in which fields appear in App Engine Studio is different from the order in which inputs are configured in Workflow Studio.

Input fields configured in Workflow Studio



Input fields displayed in App Engine Studio



Flow variables

Similar to Workflow scratchpad variables, create variables that you can use and modify directly in your flow. Access flow variables as data pills directly in the Data panel.

Use flow variables to set and retrieve values throughout a flow. Flow variables are similar to subflow inputs and outputs. Both define data available to a flow or subflow. The main difference between them is when they are accessible. Flow variables are accessible throughout a flow. Inputs are only accessible at the start of a subflow, and outputs are only accessible when a subflow completes.

Creating flow variables

Create variables with the **Flow Variables** option on the More Actions menu. You can create several variables at a time by choosing a name and data type for each one. Flow variables appear as data pills in the Flow Variables section of the Data panel.

Assigning values to flow variables

Assign values to variables with the Set Flow Variables flow logic. Set Flow Variables has the following inputs:

- The name of the variable.
- The data value for the variable.

You can assign values to all of your variables with a single use of Set Flow Variables. Unlike other data pills, the values assigned to flow variables are mutable and can be changed at any time. Using Set Flow Variables overrides the current value of the variable. If no value is assigned to a variable, the default value is **null**.

Flow variable values are set in the order in which they're assigned from top to bottom. If you set the value of the same variable multiple times, the flow only uses the last value set. For example, these three variable definitions result in the variable having the runtime value of `last value set`.

Last value set defines flow variable value

Order	Variable	Configuration
1	variable	first value set
2	variable	second value set
3	variable	last value set

Variable values can reference any data pill from earlier in the flow, including other variables. If you set variable values by reference to other data pills, you must maintain the order of the variable assignments. The referenced value must always come before the variable that uses the referenced value. Changing the order may produce null values. For example, these variable definitions only produce the expected runtime values when you maintain the order of the variable definitions.

Setting variable values by reference

Order	Variable	Configuration	Runtime Value
1	variable1	One	One
2	variable2	{variable1}, Two	One, Two

Setting variable values by reference (continued)

Order	Variable	Configuration	Runtime Value
3	variable3	{variable1}, {variable2}, Three	One, Two, Three

Flow execution details

A summary of the Set Flow Variables flow logic appears in the execution details. The details show the name, type, configuration, and runtime values for all the variables set with the action. Execution details also provide information about the variables when they're used in actions or flow logic. In that case, it shows the type, configuration, and runtime values.

Supported data types

Workflow Studio supports the following data types for flow variables:

- Date/Time
- Decimal
- Floating-point number
- Integer
- JSON
- Reference
- String
- True/False

Create a flow variable

Create a flow variable to store and retrieve a value throughout a flow.

Before you begin

Role required: flow_designer or admin

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Click **New** > **Flow**.
3. On the Flow Properties form, fill in the fields.
   For more information, see Create a flow in Workflow Studio.
4. Click **Submit**.
5. Click the more actions icon () and select **Flow Variables**.
6. In the upper-right side of the Flow Variables form, click the plus icon () to create a variable.
7. On the form, fill in the fields.

Flow Variables form

Field	Description
Label	Unique label for the variable. It can consist of any text. The label is visible to the user.
Name	Unique name for the variable. Displays the name used to identify the variable in script calls. The name can only consist of alphanumeric and underscore characters. The

Field	Description
	system automatically converts the label into a valid name by removing or replacing any special characters. The name is not visible to the user, it's what the system uses behind-the-scenes.
Type	Data type for the variable.

8. Click **Save**.

Result
The Data panel displays the variable in the Flow Variables section. It appears as a data pill, with its name and type.

What to do next
To assign a value to the variable, use the Set Flow Variables flow logic. If you don't assign a value, the default value is null.

Inline scripts

Enable users with coding experience to write inline scripts that set and modify input values during the configuration of an action or flow. Use inline scripts to modify input values that require small format conversions, data transformations, or math operations.

You can also modify input values without scripting by using transform functions. For a list of available options, see Transform functions.

Workflow Studio displays a Script button when you configure these components.

- Action inputs when you configure the action for a flow
- Action outputs when you configure the action for a flow
- Flow logic inputs when you configure the flow logic for a flow
- Flow logic outputs when you configure the flow logic for a flow
- Step inputs when you configure the step for an action.
- Subflow inputs when you configure the subflow for a flow.
- Subflow ouputs when you configure the subflow for a flow.

Inline scripts must return values in the same data type as the input expects. For example, an inline script for a Record input must return a GlideRecord object and an inline script for a Date input must return a date-time value. Always test actions and flows containing inline scripts, and verify that there are no runtime errors in the flow execution details.

Script writers should be familiar with ServiceNow AI Platform table structures and field types. In addition, they should know how to work with record and system data using the ServiceNow API.

Benefits

Inline scripts offer these benefits.

- Enable simple data conversion or transformation without having to create custom actions or flows.
- Identify which input data a script affects.
- Restricted access to scripting features to users or groups who are knowledgeable with the available ServiceNow APIs.

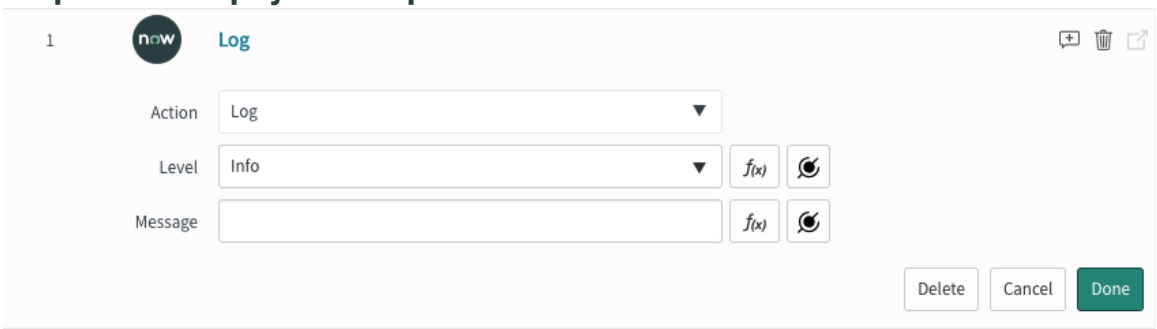
Access to inline script

You can grant users access to online scripting by either granting them the flow_designer_scripting role or the **Allow Scripting** delegated development permission. Both the role and the developer permission display a script button field for each Workflow Studio input.

Script button

When you enable a user to create inline scripts, Workflow Studio displays a script button next to flow and step inputs.

Script buttons display beside inputs



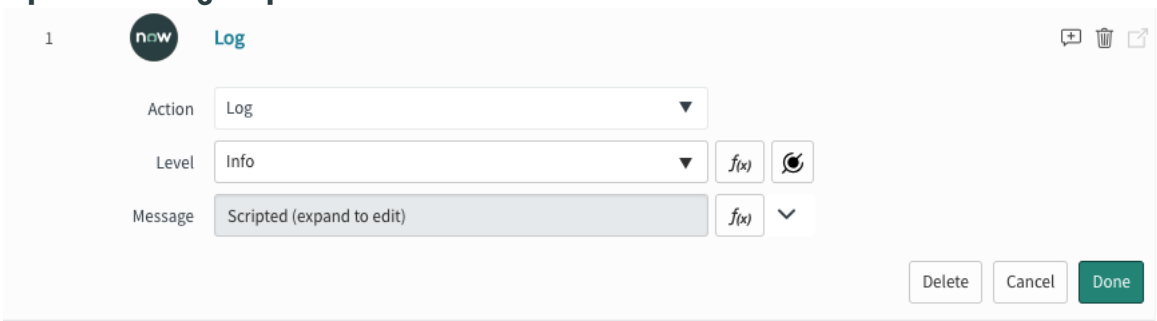
Clicking the **Script** button opens the Script editor, which replaces the standard input interface. Enter a script to compute the input value.

Ensure your script includes a `return` statement with the results of your script. For example, `return shortDesc;` returns the value of the `shortDesc` variable.

Script editor for input

Clicking the **Collapse Script** button hides the Script editor and displays a read-only version of the input. Clicking the **Expand Script** button displays the Script editor and allows you to edit the script.

Input containing script



Workflow Studio data object

Script writers can use the `fd_data` object to access data from previous actions and steps. Use the `fd_data` object to dot-walk to a specific output of the flow. You can use the script editor type ahead suggestions to select a specific output value. For example, enter `fd_data` and select `_2__for_each.item` from the type ahead suggestions to create a dot-walk reference to `fd_data._2__for_each.item`. This reference accesses data from the second output of the flow, which in this example is a For Each flow logic item.

Important:
The fd_data object always requires a dot-walk reference to a specific flow output. Without a dot-walk reference, the inline script cannot access Workflow Studio data.

Workflow Studio updates the data object each time you save your action, flow, or subflow. If the type ahead suggestions do not include your target, try saving the action, flow, or subflow, and then retyping the period character to refresh the list of suggestions.

Flow Designer data object options

Data Target	Reference Syntax	Example
Action input	`fd_data.action_inputs.`*`input-name`*	`var task = fd_data.action_inputs.task;`
Flow variable	`fd_data.flow_var.`*`flow-variable-name`*	`var shortDesc = fd_data.flow_var.short_descripti`
Output of prior action, flow logic, or step	`fd_data._`*`output-target-step-number__output-target-name`*	`var taskRecord = fd_data._2__create_task;`
Subflow input	`fd_data.subflow_inputs.`*`input-name`*	`var reqItem = fd_data.subflow_inputs.requested`
Trigger	`fd_data.trigger.`*`trigge`*	`var incident = fd_data.trigger.current;`
	`data-pill`	

General guidelines

Follow these general guidelines to create reusable and maintainable inline scripts.

Write inline script for small non-reusable logic

Use inline script format or modify the data for specific inputs and use cases. For standard and reusable input data conversions and formatting operations, select a transform function instead.

Review available transform functions

Workflow Studio provides a list of standard transform functions for data conversions and formatting operations. Rather than write and maintain a custom script solution, select an existing transform function if one is available.

Call script includes from inline script

Call a script include from your inline script to reduce the amount of code you write and also to maintain common code in a single location. Use the class constructor to call your script include. For more information about creating a script include, see Script includes.

```
var si = new MyScriptInclude();
si.functionOne();
```

Create custom actions or subflows for reusable code rather than inline script

Create custom actions or subflows for reusable or complex data logic such as changing the data type of source data. You may also want to provide custom actions or subflows for flow designers who are not comfortable with code.

Avoid duplicating action and flow functionality

Avoid writing inline script that duplicates action and flow functionality. For example, rather than write inline script to perform record operations, use the create and update record baseline actions.

Avoid data type changes

Avoid runtime errors by verifying that your inline script provides information in the same data type as the input or output expects.

Create variables by declaring them with the var keyword

Use the `var` keyword to declare variables so that they remain within the proper JavaScript scope. When you create a variable by assigning it a value, JavaScript may attach it to the global object, which can result in variable values persisting outside of the local scope and causing errors.

Process records outputs with For Each flow logic and the flow data object

Inline script can only access the **records** output of a Look Up Records action from For Each flow logic. Add a Look Up Records action to the flow to generate the records output. Add a For Each flow logic to the flow to process each record in the records output. Create an inline script reference to the For Each flow logic using the fd_data and item objects. For example, this reference assumes that the For Each flow logic is the second item in your flow outline `fd_data._2__for_each.item`.

Use type ahead suggestions to generate references to flow and action data.

Create references to flow and action data using the fd_data object. The script editor displays type ahead suggestions for existing flow and action data when you type `fd_data`. Select a suggestion to build references to flow and action data.

Note:
Refer to record data in a For Each flow logic using the **item** object.

Scope loop counters

Script loops don't have a maximum number of iterations, so loops execute infinitely when there is not a valid exit condition.

To make sure that there is a valid exit condition, use scope loop counters in inline scripts or in script steps within an action. Add `var` to `for (i=0; i< length; i++)` and get `for (var i=0; i< length; i++)`

Licensing considerations

Inline scripts that call integration APIs are subject to Integration Hub licensing.

Code editor

The code editor provides text editor support for inline 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.

Building subflows

Workflow Studio is the default ServiceNow AI Platform process automation builder used to create subflows. Workflow Studio replaces the Workflow Editor.

Unlike flows, subflows lack a trigger. Use a subflow when:

- You only want to start a flow by calling it from another flow or script.
- You want to create a set of reusable operations for use in multiple flows.
- You want to specify the inputs available to the subflow when it starts.
- You want to specify the outputs available to the parent flow after the subflow ends.

All subflows consist of properties, one or more inputs, one or more outputs, a sequence of actions and flow logic, and the data collected or created.

Subflow properties

The subflow properties specify the subflow name, application, category, description, in-flow annotation, roles, and status. Flow designers can update the subflow name, category, description, in-flow annotation, and roles at any time, but can only set the application during subflow creation. The subflow status is set when you save or publish a subflow.

Subflow inputs

Subflow inputs specify the data available to the subflow when it starts. Each input you define for a subflow becomes a configuration option in the Workflow Studio interface. To use the subflow in a flow, you must define a value for each mandatory input. The more inputs a subflow has, the more data you must define and the more familiar you must be with the underlying data model to use the subflow effectively.

Inputs provide advanced options based on their data type. All inputs have advanced options to add a hint or provide a default value. Use advanced options to guide flow designers through adding and configuring a subflow to a flow. For example, create a choice input to provide flow designers with a pre-defined list of configuration options to choose from. For more information about the configuration options available to particular data types, see field types.

Subflow outputs

Subflow outputs specify the data available to the parent flow after the subflow completes. Subflow outputs are defined as variables with a name and data type. Subflow designers assign values to an output using the **Assign Subflow Output** flow logic. Output values can be based on the subflow logic conditions, action results, or a manually set value. For example, an output may have one value when a condition is met and another value when a condition is not met. During runtime, the value of the output is determined by the condition that is met.

Consider the following example of a subflow with two conditions that both result in a value for a single output variable. The value of the variable depends on which condition is met during runtime.

Outputs

[Manager ID] [String]

Actions

- 1 Look Up [User] Record where (Created on Today)
- 2 If (**[1->User Record->Title]** contains Manager) then, Assign Subflow Outputs [Manager ID] to **[1->User Record->User ID]**
- 3 Else, Assign Subflow Outputs [Manager ID] to **[1->User Record->Manager->User ID]**

In this case, if the user's title contains "Manager" then the user ID is assigned as output. Otherwise, the subflow looks up the user's manager and assigns the user ID of the manager as output.

Workflow Studio allows you to define a value for the same variable multiple times. However, if a variable is given two or more possible values without conditional logic, only the last value defined in the subflow is applied to the output at runtime.

Outputs

[Manager ID] [String].

Actions

- 1 Look Up [User] Record where (Created on Today)
- 2 Assign Subflow Outputs [Manager ID] to **[1->User Record->User ID]**
- 3 Assign Subflow Outputs [Manager ID] to **[1->User Record->Manager->User ID]**

In this example, action three overwrites the value of action two and **[1->User Record->Manager->User ID]** is applied to the [Manager ID] output at runtime because it was the last value defined. Typically, subflows should only include multiple values for one variable if conditional flow logic is used.

Subflow execution details

Process analysts can view subflow execution details from multiple locations.

Parent flow execution details

Workflow Studio displays subflow execution details within the parent flow execution details. The parent flow execution details list each subflow as inline elements. You can expand a subflow step to see its execution details.

Subflow execution details

The system generates flow execution details for each subflow run. View subflow execution details directly from the list of flow executions.

Actions

Within **Actions**, flow designers can add actions, flow logic, flows, or other subflows.

An action is a reusable operation that enables process analysts to automate ServiceNow AI Platform features without having to write code. For example, the **Create Record** action allows process analysts to generate records in a particular table with particular values when certain conditions occur. ServiceNow core actions like Create Record require some familiarity with ServiceNow AI Platform tables and fields. Action designers can create application-specific actions to pre-set configuration details. For example, creating a Create Incident Task action ensures that the process analyst uses the correct table and field configuration each time the action is used. You can add application-specific actions by activating the associated spoke.

Flow logic

Subflows can contain flow logic to specify conditional or repeated actions, or to assign output variables to subflow data. The system provides these flow logic options.

Available flow logic

Flow logic	Description
For Each	Applies actions to each record in a list of records. Flow designers must specify l list of records from the subflow data. **Note:** You can nest a **For Each** flow logic block inside of another flow logic block to repeat an action over a series of records. However, avoid nested For Eac loops that process many records. Nested loops may cause the flow to run until it is stopped by the flow transaction quota rule, which prevents flows from running longer than an hour. For more information about transaction quotas, see Transaction quotas.

Available flow logic (continued)

Flow logic	Description
If	Applies actions when a list of conditions is met. Flow designers can specify the conditions with subflow data. Once an If condition is added, you can add an Else or Else If flow logic option to define behavior when conditions are not met.
Assign Subflow Outputs	Assigns an output variable to subflow data. Only outputs defined in **Inputs & Outputs** can be assigned a value. Assigning outputs enables you to assign a different output variable for each logical path in the subflow.

More Actions

Click the **More Actions** ( ) button to access additional options for the subflow.

Copy action

Create a copy of the open subflow in an application you specify.

Configurations

Enable or disable the **Show draft actions**, **Show triggered flows**, **Show store spokes**, and **Show inline script toggle** options.

Code Snippet

Generate a code snippet for the action.

Manage security

Enable or disable the **Callable by Client API** option.

Manage natural language title

Create or edit a subflow title with styled or dynamic text. For more information, see Manage natural language titles.

Testing subflows

You can test a subflow alone, or when added to a flow. When testing a subflow alone, you must define the inputs that the subflow uses in its actions. Because a subflow does not have a trigger, testing a subflow runs the actions using the defined input values.

Note:
Because testing a subflow creates or changes records on the instance, flow designers should always test subflows on a non-production instance containing relevant demonstration data.

Roles

To access subflows, a user must have the flow_designer or admin role.

General guidelines

General guidelines that apply to flows also apply to subflows.

Reasons to use a subflow instead of a flow include:

Reuse business logic

Create a set of reusable operations as a subflow that can then be used in multiple flows.

Reuse business logic

Create a set of reusable operations as a subflow that can then be used in multiple flows.

Configure different input values for each call

Configure a subflow's input values differently each time you call it. For example, design a subflow to accept different record types as an input run. Reuse this generic record subflow instead of writing a specific flow for each record type.

Improve performance and readability of large flows

Use subflows when a flow exceeds 25 actions. 50 is the maximum number of actions specified by the sn_flow_designer.max_actions system property, but limit a flow to 25 actions for the best performance.

Limit subflows to 20 inputs

The more inputs your subflow has, the more resources it takes to open and run it. Processing more than 20 inputs risks the subflow being slow to open and run.

Pass inputs and outputs with subflows

Call subflows if you want to pass inputs and outputs. Use subflows if you want to specify the inputs available to a subflow when it starts, or if you want to specify the outputs available to the parent flow after a subflow ends.

Trigger multiple flows on a single event vs. using parallel subflows

- Use parallel subflows if there are interrelated outputs or if some action must be taken when all are available. If not, then it's simpler to trigger multiple flows.
- To configure parallel subflows, launch each subflow without a wait and then use wait for condition to wait for each subflow to be terminal (complete, error, canceled)

Use dynamic flows if you have multiple subflows with similar functionality

Dynamic flows let you compartmentalize your processes by applying a template to handle the inputs of multiple similar subflows. Compartmentalization lets you distinguish between subflows that perform similar functions, such as subflows for IntegrationHub spokes.

Avoid the 10-item limit in the error-handling-process

Rather than force your error-handling-process to fit within a 10-item limit, call subflows, which can contain many more items. You can also use the subflow outputs to trigger automation in other flows.

Take corrective actions

Rather than recreate the same sequence of actions in multiple flows, create reusable subflows to correct errors to your record data. When a flow error leaves your record data in an undesired state, use subflows to correct these records. You can use the error handler to identify such record data as a subflow output.

Create a subflow in Workflow Studio

Reuse an entire flow's content as a subflow. Define the input data the subflow uses and the output data it generates. Call subflows from other flows or script.

Before you begin

Set up an application in Guided Application Creator to store Workflow Studio content.

Role required: flow_designer or admin

About this task

Users with the flow_designer or admin role should know the application table structure and be aware of any existing business logic associated with the target tables of a flow or subflow. Be sure to disable any conflicting business rules or workflows before creating a flow or subflow.

Creating a custom application to contain your Workflow Studio content allows you to deploy it using the application repository or the ServiceNow Store.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Click **New** > **Subflow**.
   The Subflow properties screen displays.
3. Fill in the following fields:

Field	Description
Subflow name	Name of the subflow.
Description	Description of the subflow.
Application	Application scope to create the subflow in.
Domain	Domain scope of the subflow. For more information about domain separation, see Domain separation explained.
Additional properties	
Accessible From	Accessible from all application scoped or only within the specified application scope.
Category	Defined category within the application scope for a subflow. Beginning with the Xanadu release, you can create a custom category to organize your subflows.
Protection	Select whether the subflow is read-only. You can only select a value when you create the subflow in an application scope you own. The default value is None.
Subflow annotation	Help text that appears under the subflow title in Workflow Studio to help flow authors understand what the subflow does when used in a flow.
Run as	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 the flow. Settings for the Run as option in a flow don't apply to child subflows. Running as the initiating user also ensures the actions taken during flow execution are limited by the user's ACL restrictions. Flows run by the initiating user will also respect user-specific settings like date/time format. **Note:** When choosing the option to run as the user who initiates the session, ensure that your security restrictions do not prevent your users from making any changes the flow executes.

Field	Description
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**.
Flow priority default	Priority level at which you want the system to run this subflow by default. Options include: ○ Low ○ Medium (Default) ○ High To learn about flow priority levels, see Flow priority.

4. Create subflow inputs to specify the data available to the subflow when it starts running.
   Each input you define for a subflow becomes a configuration option in the Workflow Studio interface. To use the subflow in a flow, you must define a value for each mandatory input. The more inputs a subflow has, the more data you must define and the more familiar you must be with the underlying data model to use the subflow effectively.

   a. Click **+** to open the Inputs & Outputs pane.

   b. Click **+** to add a new input.

   c. Define the name and type for the input.

   Note:
   Subflow input names can't include any of the following reserved system names:
   - `sys_id`
   - `sys_created_by`
   - `sys_created_on`
   - `sys_updated_on`
   - `sys_updated_by`
   - `sys_mod_count`

   d. To make the input a mandatory configuration option, select the **Mandatory** flag.

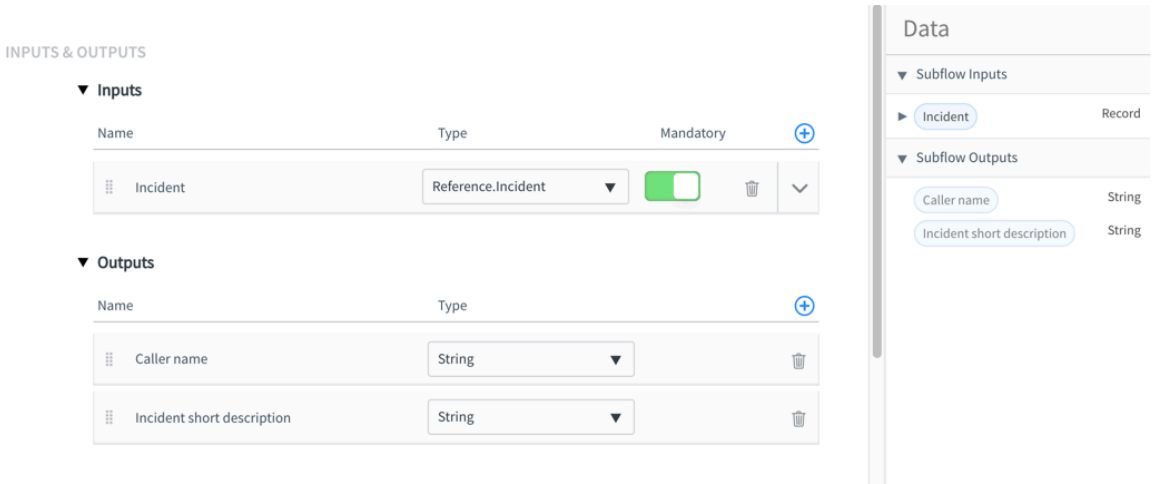
   e. Click ⌄ to view the advanced options and define values.
   Inputs provide advanced options based on their data type. All inputs have advanced options to add a hint or provide a default value. Use advanced options to guide flow designers through adding and configuring a subflow to a flow. For example, create a choice input to provide flow designers with a pre-defined list of configuration options to choose from. For more information about the configuration options available to particular data types, see field types.

5. Create subflow outputs by defining the names and data types.
   Subflow outputs specify the data available to the parent flow after the subflow completes.

a. Click **+** to add a new output.

b. Define the name and the data type.
Output values are assigned in later steps.

> **Note:**
> Subflow output names can't include any of the following reserved system names:
>
> - `sys_id`
> - `sys_created_by`
> - `sys_created_on`
> - `sys_updated_on`
> - `sys_updated_by`
> - `sys_mod_count`



6. 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**. 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.

Option	Description
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.

7. Assign subflow outputs to a value.
You can assign a subflow output to multiple values, enabling you to create conditional outputs based on flow logic.

a. Under **Actions**, click **+** and select **Flow Logic**.

b. Click **Assign Subflow Outputs**.

c. In the **Name** field, select an output you created in the Inputs & Outputs section.
You can assign values onlyl to outputs that have already been given a name and data type.

d. In the **Data** field, enter a value or select a data pill from the data panel.

e. Click **Done**.

What to do next

Test the subflow, and publish it when it is ready to be added to a flow or called from a script.

Note:
You can only test or publish subflows that contain at least one action.

Copy a subflow

Copy a subflow to give it a new name and move it to another application scope.

Before you begin

Role required: admin or flow_designer

About this task

You can copy a subflow to give it a new name or move it to another application scope. The new subflow has the same subflow properties, inputs, actions, flow logic, and outputs as the source subflow.

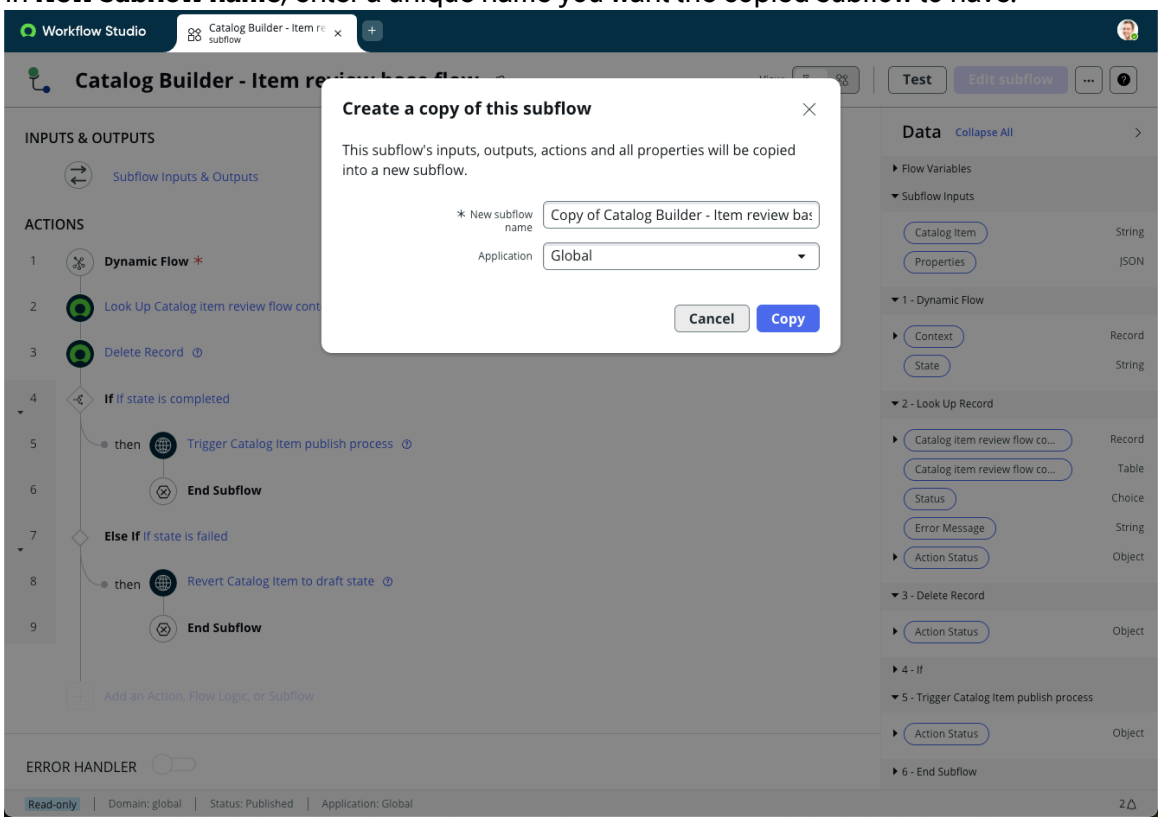
You can't copy subflows that have a protection policy. You must have write access to a subflow to copy it.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Subflows**.
3. Select the subflow that you want to copy.
4. Click the more actions icon (...) and select **Copy subflow**.

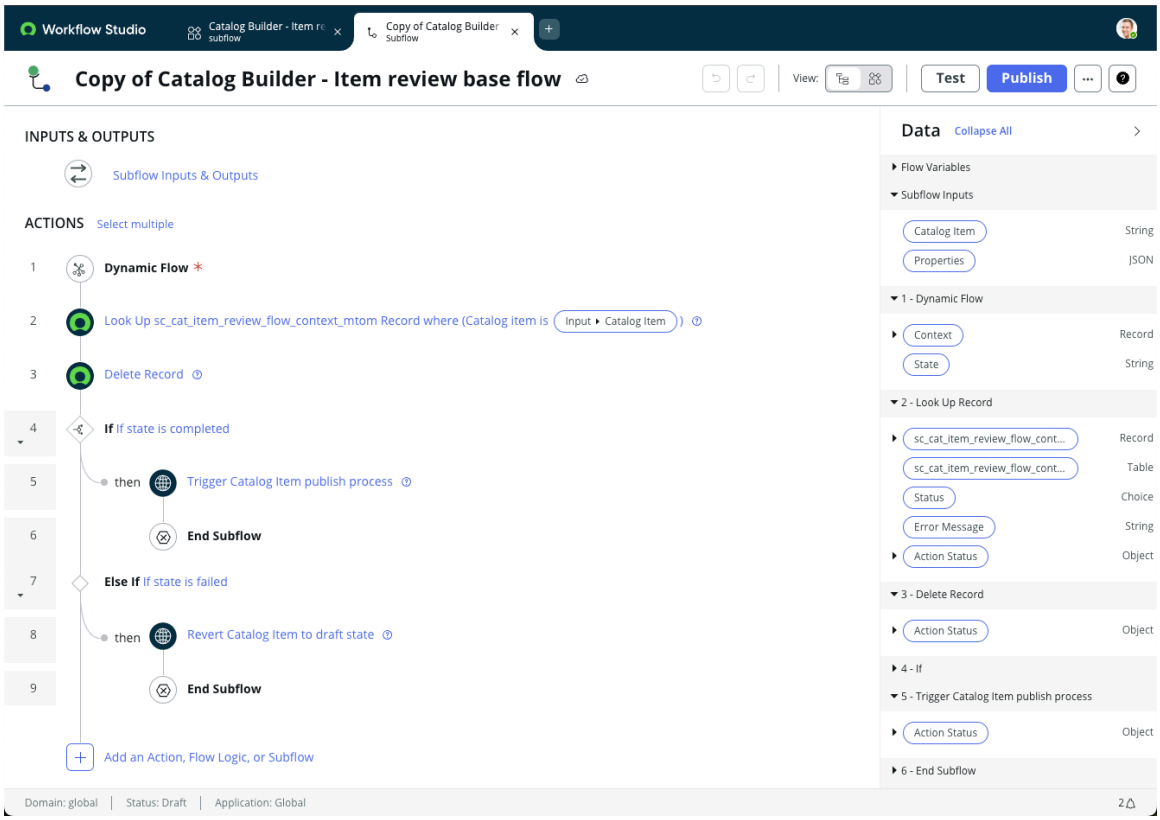
> **Note:**
> If the **Copy subflow** option is not visible, then you don't have permission to copy the subflow. This could be because the subflow has a protection policy or because you lack the necessary user role or developer permissions.

5. In **New subflow name**, enter a unique name you want the copied subflow to have.



6. **Optional:** From **Application**, select the application scope where you want to copy the subflow.
7. Select **Copy**.

Result



Workflow Studio opens the new subflow.

Create a decision table in a subflow

Create a decision table structure while you author your flow in Workflow Studio. Use data from the subflow to create inputs, conditions, and results for the decision table, all in a convenient modal.

Before you begin

Role required: admin, decision_table_admin, flow_designer, or delegated developer permissions

About this task

Creating a decision table in-line in a subflow only creates the structure of the table with inputs, conditions, and results. To complete the table by adding the actual decision rules, you must open and edit the decision table in its own tab.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Subflows**.
3. Select a subflow.
4. Under Actions, select **Flow Logic**.
5. Select **Make a Decision**.
6. In the **Decision Label** field, enter a unique label for the decision.
7. In the **Decision Table** field, select the Create new record (+) button.
   In the Create decision table modal, on the Set Properties page, two editable fields are populated from your subflow.

Set Properties

Field	Description
Decision table name	Unique name for the table. This name is populated from the **Decision Label**.
Application	Application where the decision table lives. The application is populated based on the application that the subflow is in.

8. Select **Next**.
9. Select **Add input** to add inputs to the decision table.
   Inputs are the variables that define the type of data the decision table looks for to make decisions. When creating a decision table in a subflow, you can add inputs directly from the data in the subflow. For more information about the types of inputs you can add, see Create decision tables in Workflow Studio.

   Note:
   Some inputs must be added or adjusted when you open the decision table to populate its values.

   - If you want to add inputs of type Reference to this decision table, you must do that when you open the decision table in its own tab.
   - If you want to add inputs of type Choice in this modal, the choices themselves don't appear when you begin editing the decision table on its own. You must add them manually, because choice options are based on the scope you're in. For example, if you add a **Priority** field with **Choice** as the type, the options for priority (Low, Medium, High, and so on) aren't added automatically. You can add them when you open the decision table to populate its values.

10. Select **Next**.
11. Select **Add Result Column** to begin adding the result columns to your table.
    Results are the outputs your decision table returns when certain conditions are met.
12. **Optional:** Add any additional configurations required for the result type that you selected.
13. Select **Create and Open**.
    The decision table opens in its own integrated tab within Workflow Studio. You can edit any part of the table here, including adding Reference type filters to the inputs or results.

    Condition columns are added to the decision table based on corresponding input fields.
14. Complete the decision table by adding decision rules to the condition columns and result values.
15. Select **Save**.
16. **Optional:** In your original subflow, review and test the decision to make sure it produces the expected results.
17. Select **Done**.

Convert items to subflow

Convert consecutive items of a flow into a new subflow that preserves data pill references between the converted items. Update the original flow to replace the converted items with a call to the new subflow.

Before you begin

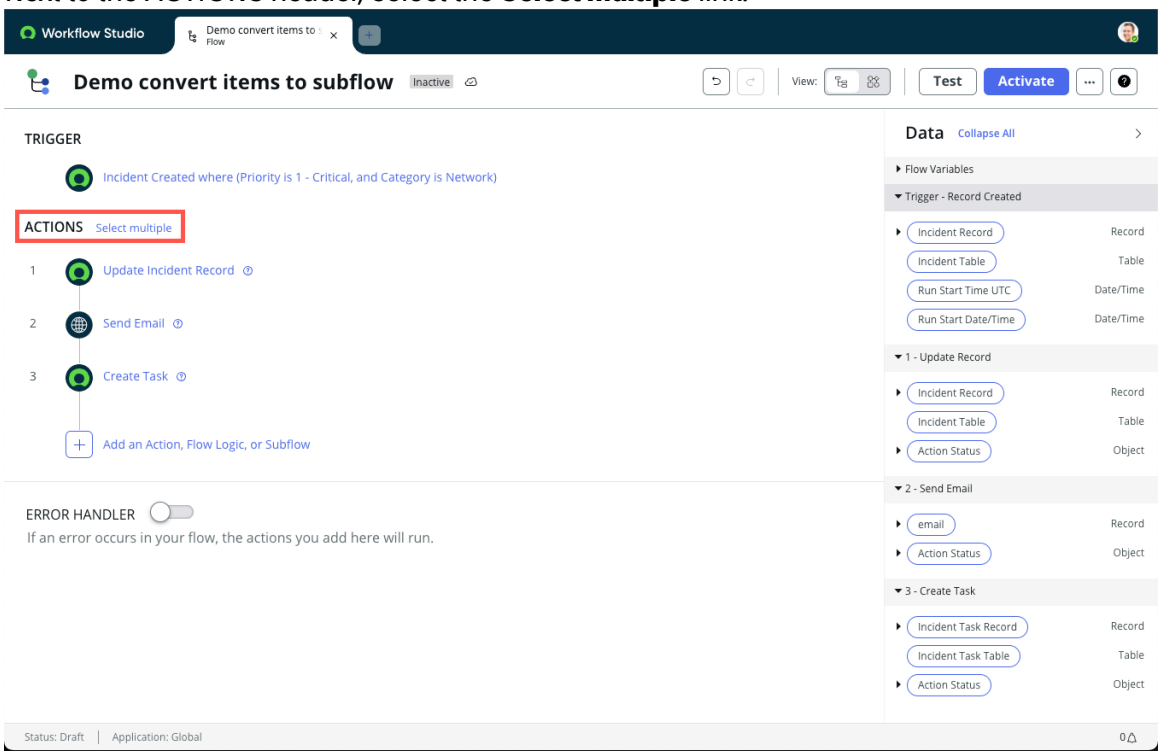
- Role required: flow_designer or admin
- Read and write access to the source flow

About this task

Converting actions into a subflow preserves data pill references and transform functions from the original flow. Workflow Studio creates the new subflow in the same application scope as the original flow.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Open the flow you want edit.
3. Next to the ACTIONS header, select the **Select multiple** link.



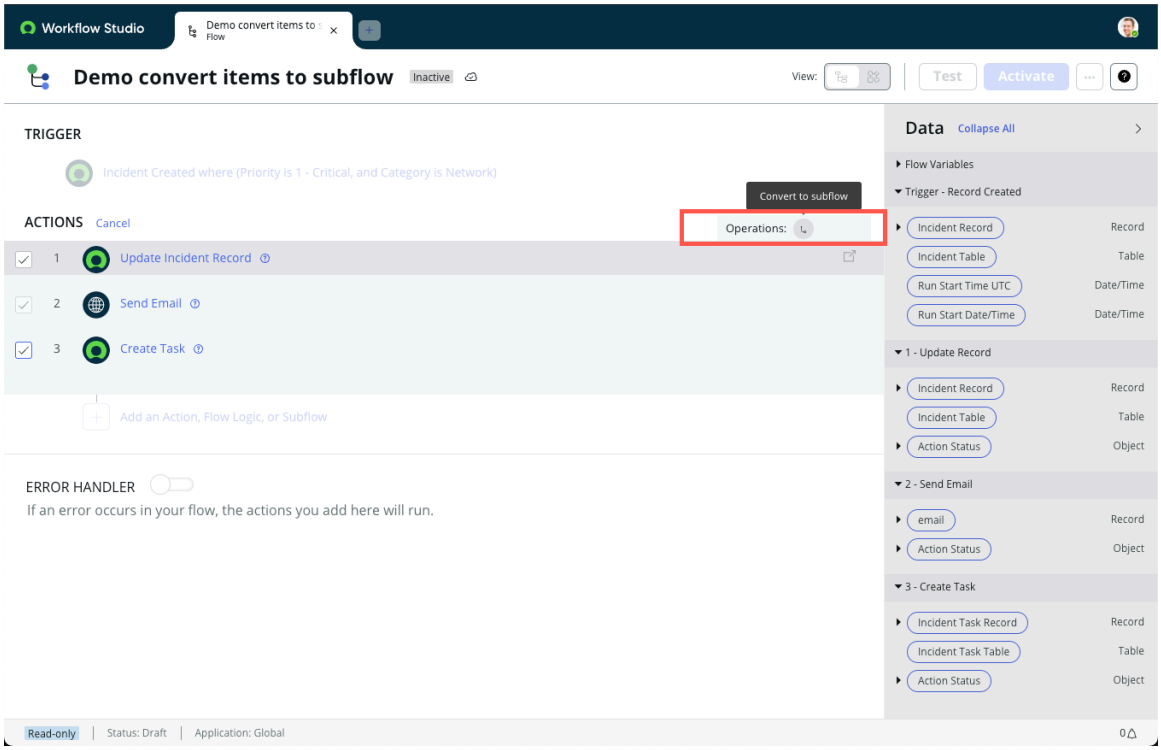
Important:
The flow must contain at least two actions in order to display the Select multiple link.

Workflow Studio displays a checkbox next to each action.

4. Select one or more consecutive actions to convert.

Important:
You can't convert a Set Flow Variables flow logic available in one flow to a new subflow. Instead you must define and set flow variables in the subflow in which you want to use them.

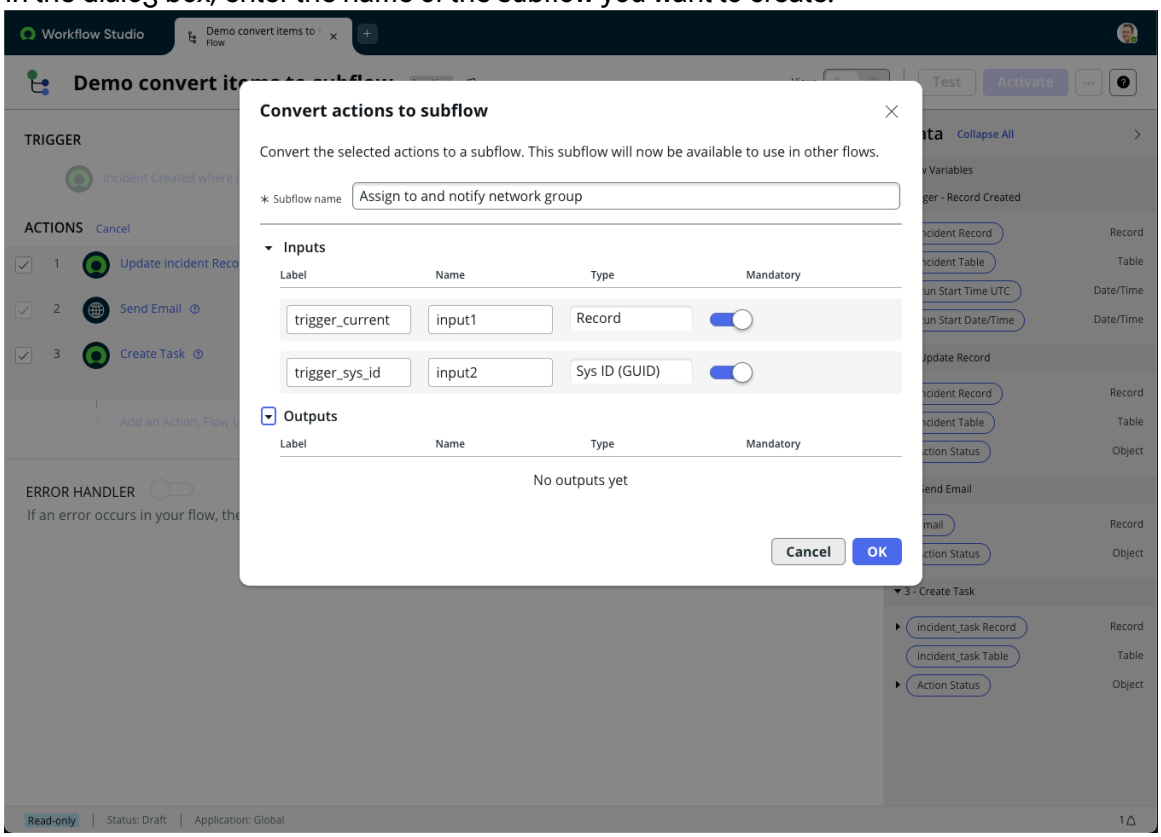
5. From Operations menu, select the Convert to subflow icon ( ).



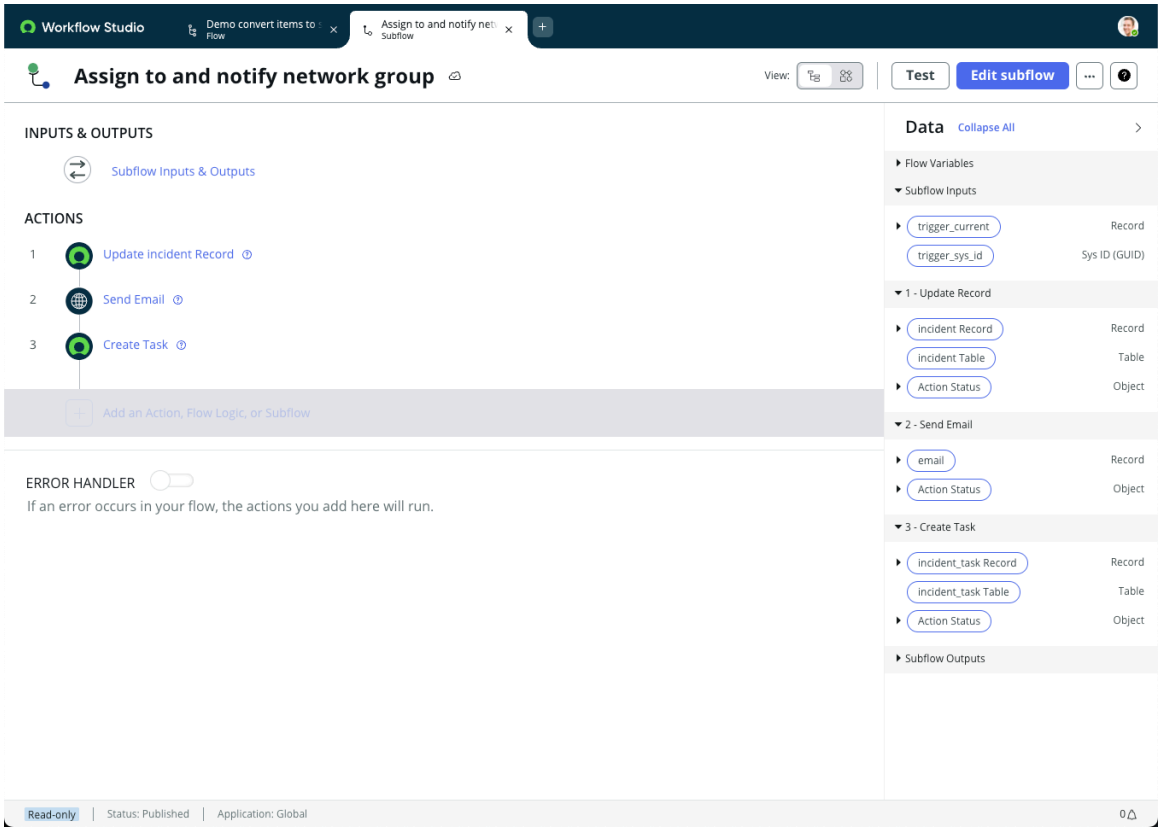
Important:
You must have write access to the flow to convert items to a subflow. A read-only protected flow doesn't support converting items into a subflow.

Workflow Studio displays the Convert to subflow dialog box.

6. In the dialog box, enter the name of the subflow you want to create.

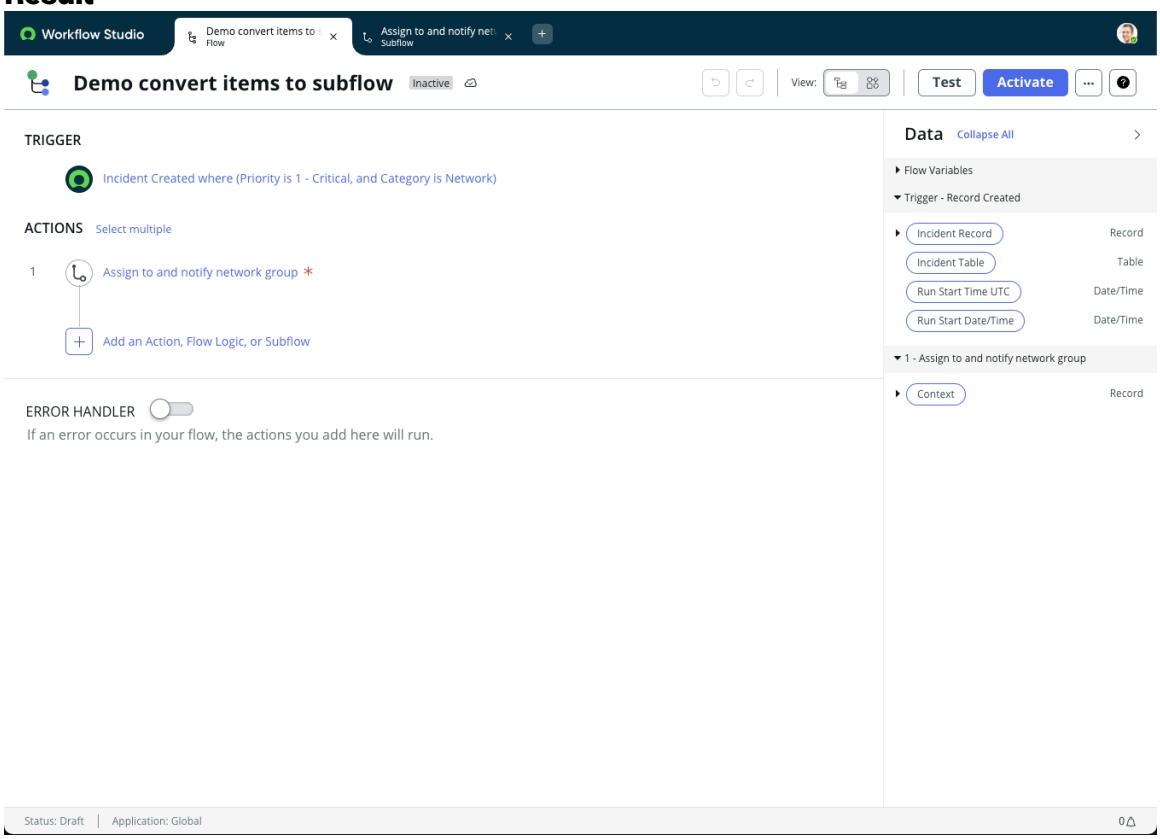


7. Select **OK**.



The selected actions are removed and replaced by the Call a subflow flow logic block. Workflow Studio creates and publishes a subflow that contains the same actions, input values, references, and transform functions from the original flow.

Result



Workflow Studio updates data pill references in the original flow to their new locations. References to actions within the new subflow include their new subflow name. References to actions within the original flow include their new sequence number.

What to do next

Review the new subflow, and update the original flow to provide the input values needed for the subflow.

Create a template value input

Enable flow authors to set field values for a record being created or updated. Use a template value input to set different field values each time you add an action or subflow to a flow.

Before you begin

A template value input can only be created in a subflow action or action step that creates or updates a record, such as Create Catalog Task, Create Task, Create Record, and Update Record.

Role required: admin or action_designer

About this task

When creating or updating a record in a subflow action or action step, you can set static or dynamic input values. A static input value is the same in every flow you add it to. For example, setting the Urgency to the static value 1 - High generates an urgent catalog task in every flow. A dynamic input value allows a flow author to change the value during flow configuration. For example, a flow author could configure one flow to create a catalog task with an Urgency to 1 - High, but another flow could create a catalog task with an Urgency of 4 - Low.

Procedure

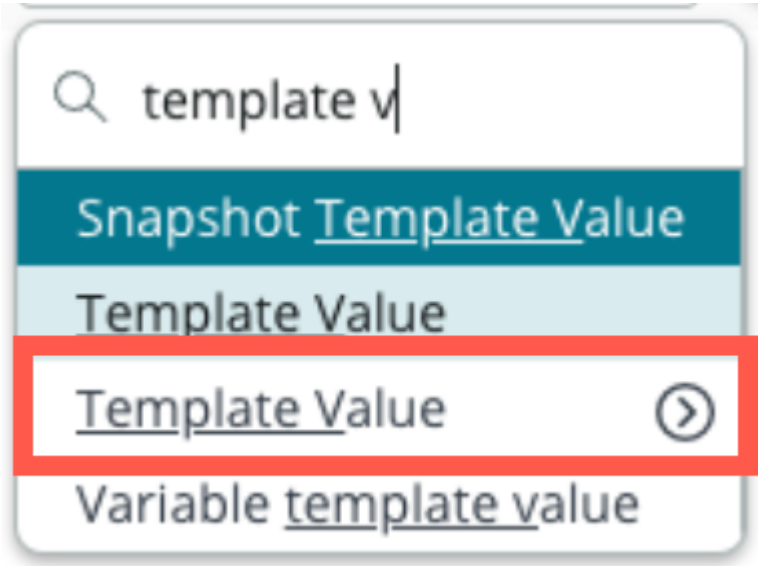
1. Open an action in Workflow Studio or a subflow in Workflow Studio that you want to create a template value for.
2. Create an input.

   a. In the **Label** field, enter a label to help flow designers understand the purpose of the field. This is the label for the field when the flow designer adds the subflow or action to a flow.

      Example
      For example, enter `Select fields`.

   b. In the **Type** field, select the Template Value option with the tables icon.

Use the tables icon to select the table containing the record you will create or update. For example, if you are adding a Create Catalog Task action in a subflow, set the Type to Template Value.Catalog Task [sc_task].



3. Add an action step or action that will use the template value.
   The action step or action must create or update a record.

   Example
   For example, add a Create Task action to a subflow that will create an incident task.
4. Drag the Template Value data pill into the **Field Values** or **Fields** field.
5. **Optional:** You can set static values in addition to dynamic template values by selecting **+ Add Field Value**.

   Note:
   Avoid setting static values that you want flow designers to set dynamic values from the template input. The flow always uses the static value from the subflow or action step over a value entered from a template. You can use static values to enforce business policies that you do not want flow designers to change.

Result

When the subflow or action is added to a flow, a flow author can set field values for the record being created or updated.

Get started with Dynamic Flow and Get Flow Outputs

Create a sample flow that dynamically calls subflows for provisioning cloud services.

Before you begin

Role required: flow_designer or admin

About this task

To understand how to use the Dynamic Flow and Get Flow Outputs flow logic together, the flow that you create in this task dynamically runs subflows that are related to provisioning cloud services. This flow does the following:

1. Triggers when a Cloud Instance Provisioning Request record is created.
2. Calls the appropriate subflow to create a Cloud Instance record.
3. Gets an output from the dynamically called subflow and updates the Cloud Instance Provisioning Request record with the output's value.

Procedure

1. Navigate to **All** > **System Applications** > **Studio**.
2. On the Select Application screen, click **Create Application**.
3. On the Guided App Creator welcome screen, click **Let's get started**.
4. In the **Name** field, enter `Cloud Instance Provisioning` and then click **Create**.
5. In the **Roles** field, enter `flow_designer` or `admin` and then click **Continue**.
6. From the list of app formats, select **Classic** and then click **Continue**.
7. Select **Create new table** > **Create table from scratch** > **Continue** to create two new tables for your application.
   - Add the following fields for the first table and name the table `Cloud Instance`.

Field label	Field type	Reference
Owner	Reference	User [sys_user]
Instance Type	String	None
Instance URL	URL	None
Build Status	String	None

   - Add the following fields for the second table and name the table `Cloud Instance Provisioning Request`.

Field label	Field type	Reference
Requested Instance Type	String	None
Requested by	Reference	User [sys_user]
Approval Status	String	None
Approved Instance	URL	None

8. After creating both tables, click **Done with tables**.
9. Select **Start** > **Create** > **Done with apps** to finish creating your application.

Create a subflow template

Create a sample subflow template for provisioning cloud services.

Before you begin

Role required: flow_designer or admin

Procedure

1. Under Business logic, process automation, and integrations, select **Go to Flow Designer**.
2. In the Workflow Studio landing page main header, select **New** > **New Subflow**.
3. On the Subflow Properties screen, enter `TEMPLATE: Create Cloud Instance Record` in the **Name** field and then select **Submit**.
4. Under Inputs & Outputs, select the plus icon (⊕) to create two inputs for your subflow template.

Label	Type
Requested by	Reference.User
Requested Instance Type	String

5. Select the plus icon (⊕) to create one output for your subflow template.

Label	Type
Instance URL	URL

6. Under your output, select **Done**.
7. Select the add action, flow logic, or subflow to end of flow icon (⊕) and then select **Action**.
8. From the list of ServiceNow Core actions, select **Log**.
9. In the data panel, drag the data pills for the `Requested by` and `Requested Instance Type` inputs you created earlier, and drop both of the pills into the **Message** field.
10. In the Subflow header, select **Save** and then select **Publish** to publish your subflow template.

Create a subflow for Jira cloud instance provisioning requests

Create a sample subflow for provisioning cloud services from Jira.

Before you begin

Role required: flow_designer or admin

Procedure

1. In the Subflow header, select the more actions icon (···) and select **Copy subflow**.
2. On the Copy Subflow screen, enter `Create Cloud Instance Record - Jira` in the **New Subflow Name** field, and then select **Copy**.
3. Under the **Log** action, add a Create Record action.
4. In the **Table Name** field, select `Cloud Instance` and then fill in the following fields.

Field	Value
Owner	Select the data pill picker (⊞) and select **Subflow - Inputs** > **Requested by**.

Field	Value
Instance Type	Select the data pill picker ( ) and select **Subflow - Inputs** > **Requested Instance Type**.
Instance URL	Enter `https://mycompany-`. Then, select the data pill picker ( ) and select **Subflow - Inputs** > **Requested by** > **Name**. Finally, enter `.atlassian.net`
Build Status	Enter `In Progress`.

5. Under the create record action, select **Add an Action, Flow Logic, or Subflow** > **Flow Logic** > **Assign Subflow Outputs**.
6. Select the plus icon to add an output.
7. For **Name**, select **Instance URL**.
8. For **Data**, select the data pill picker ( ) and then **2 - Create Record -** > **Cloud Instance Record** > **Instance URL**
9. In the Subflow header, select **Save** and then **Publish** to publish the subflow.

Create a subflow for Salesforce cloud instance provisioning requests

Create a sample subflow for provisioning cloud services from Salesforce.

Before you begin

Role required: flow_designer or admin

Procedure

1. In the Subflow header, select the more actions icon ( ) and select **Copy subflow**.
2. On the Copy Subflow screen, enter `Create Cloud Instance Record - Salesforce` in the **New Subflow Name** field, and then select **Copy**.
3. Expand the **Create Cloud Instance Record** action and replace the `.atlassian.net` value for the **Instance URL** field with `.salesforce.com`
4. In the Subflow header, select **Save**, accept the data change warning message, and then **Publish** to publish the subflow.

Create a flow that runs your subflows dynamically

Create a sample flow to run your provisioning cloud services subflows.

Before you begin

Role required: flow_designer or admin

Procedure

1. Under the Workflow Studio header, select the **Create flow, subflow, or action**( ) icon and select **Flow**.
2. On the Flow Properties screen, enter `Process Cloud Instance Provisioning Request` in the **Name** field, and then select **Submit**.

3. Under Trigger, select the plus icon) to add a trigger to your flow, and then fill out the following fields.

Field	Value
Trigger	Select **Created**.
Table	Enter **Cloud Instance Provisioning Request [x_cloud_instance_p_cloud_instance_provisioning_request]**

4. Under Actions, select the plus icon and then select **Flow Logic** > **Dynamic Flow**.
5. Fill in the following fields.

Field	Value
Flow Template	Select **TEMPLATE: Create Cloud Instance Record**
Flow	Enter `Create Cloud Instance Record -` and then select the data pill picker ( ) and select **Trigger - Record Created** > **Cloud Instance Provisioning Request Record** > **Requested Instance Type**.
Wait for completion	Enable this option to run your subflow dynamically first before other actions in your flow occur.
Requested by	Select the data pill picker ( ) and select **Trigger - Record Created** > **Cloud Instance Provisioning Request Record** > **Requested by**.
Requested Instance Type	Select the data pill picker ( ) and select **Trigger - Record Created** > **Cloud Instance Provisioning Request Record** > **Requested Instance Type**.

6. Under your **Dynamic Flow** flow logic, add the Get Flow Outputs flow logic, and then fill in the following fields.
7. For **Flow Template**, select **TEMPLATE: Create Cloud Instance Record**
8. For **Context**, select the data pill picker ( ) and select **1 - Dynamic Flow -** > **Context**.
9. Under your **Get Flow Outputs** flow logic, select the plus icon ( ) and then select **Action** > **Update Record** to add an Update Record action to your flow.
10. Fill in the following fields.

Field	
Record	Select the data pill picker ( ) and select **Trigger - Record Created** > **Cloud Instance Provisioning Request Record**.
Fields > **Approved Instance**	Select the data pill picker ( ) and select **2 - Get Flow Outputs -** > **Instance URL**.
Fields > **Approval Status**	Enter `Approved`.

11. Select **Save**.

Test your flow

Test your sample flow for provisioning cloud services.

Before you begin

Role required: flow_designer or admin

Procedure

1. In the Flow header, select **Test** to test your flow.
2. On the Test Flow screen, select the Create new record ( ) icon to create a new record.
3. For **Requested By**, select the Lookup using list icon ( ) and select any user from the list.
4. For **Requested Instance Type**, enter either `Jira` or `Salesforce`.
   The value you enter determines which subflow runs dynamically at runtime. Entering `Jira` runs the *`Create Cloud Instance Record - Jira`* subflow, and entering `Salesforce` runs the *`Create Cloud Instance Record - Salesforce`*.
5. Select **Submit**.
6. Select **Run Test**, and when the flow finishes running, select **Your test has finished running. View the flow execution details.**
   Your flow runs successfully if the values in the State column for each step in your flow shows **Completed** and each step's runtime value populates appropriately.
7. Select the tab for your **Process Cloud Provisioning Requests** flow and close the Test Flow modal.
8. In the Flow header, select **Activate** to make your flow accessible within the Cloud Instance Provisioning Application scope.

Result

When a user in your instance creates a new record in the Cloud Provisioning Request table, your Process Cloud Provisioning Requests flow runs automatically. This flow dynamically creates the proper Cloud Instance record that is based on the requested instance type. It also generates a cloud instance URL, which populates in the Cloud Instance Provisioning Request record.

Test a subflow

You can test a subflow alone, or when added to a flow. When testing a subflow alone, you must define the inputs that the subflow uses in its actions. Because a subflow does not have a trigger, testing a subflow runs the actions using the defined input values. Unless updated, subsequent tests use the same inputs defined in the initial test run.

Before you begin

Create a subflow in Workflow Studio that contains at least one action and save it. Workflow Studio only tests saved subflows that contain at least one action.

Role required: flow_designer or admin

About this task

Because testing a subflow creates or changes records on the instance, flow designers should always test subflows on a non-production instance containing relevant demonstration data.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Click the subflows tab and open a saved subflow.
3. Click **Test**.
   The Test Subflow dialog opens.
4. Define input values for the subflow to use in its actions.
   The values defined are remembered on future test runs.
5. Click **Run Test**.

   Note:
   Select the **Run test in background** option to test a subflow asynchronously in the background.

   If you select the **Run test in background** option, the execution details are displayed only after the execution is completed asynchronously in the background. Also, the execution details are associated with the subflow only after execution is completed.

6. After the flow executes, click **Subflow has been executed. To view the subflow, click here**.
   The Execution Details open.

What to do next

Review the Flow execution details.

Once the subflow behaves as desired, you can publish the subflow and add it to a flow.

Publish a subflow

Publish a subflow to make it available to other users and to add it to activated flows.

Before you begin

Role required: flow_designer or admin
Create a subflow in Workflow Studio, test the subflow, and verify that it is working as expected.

About this task

When you make changes to a published subflow, the changes remain in the draft state until you publish the subflow again. You must publish a changed subflow to make the changes available to activated flows.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Click the subflows tab and open a tested subflow.
3. Click **Publish**.

Result
The subflow can be added to activated flows. If you update the subflow after it is published, you must click **Publish** again to see the changes when the parent flow is run. After publishing changes, all parent flows that use the subflow are automatically updated to use the current version.

Building actions

Workflow Studio is the default ServiceNow AI Platform process automation builder used to create actions. Workflow Studio replaces the Workflow Editor.

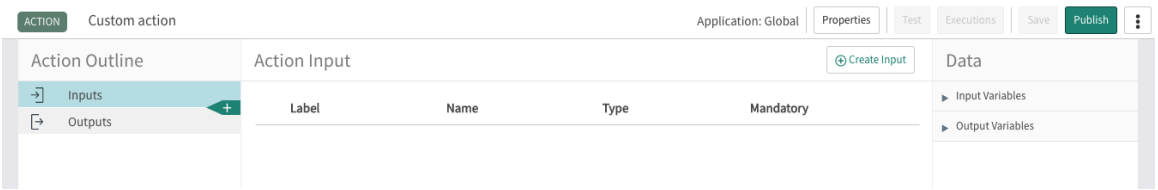
Core actions

ServiceNow Core actions that come with your instance cannot be viewed or edited from the Workflow Studio interface. Some applications include spokes which include application-specific actions. Spoke actions are typically read-only but can be copied and customized.

Custom actions

Unlike core actions where flow designers must manually configure flow logic, custom actions always use the same configuration when added to a flow. If your flow designers often use an action with the same configuration, you might create a reusable action.

Action design environment



Create and edit actions by defining inputs and adding action steps. Test actions to verify if they complete successfully and review the runtime values they generate. Copy actions to use existing actions as templates. Publish actions to activate them, which makes them available to activated flows and to preserve their current action steps, variables, and sequence as a snapshot separate from further configuration changes.

More Actions

Click the More actions icon ( ) to access additional options for the action.

Copy action

Create a copy of the open action in an application you specify.

Configurations

Enable or disable the **Show inline script toggle** option.

Code Snippet

Generate a code snippet for the action.

Manage security

Enable or disable the **Callable by Client API** option.

Manage natural language title

Create or edit an action title with styled or dynamic text. For more information, see Manage natural language titles.

Testing actions

After adding inputs and action steps, users with the action_designer or admin role can test an action. To test an action, provide the required inputs. Action designers should always test actions on non-production instances containing relevant demonstration data because testing an action can make significant changes to records on your instance.

Roles

To create custom actions, you must have the action_designer or admin role.

Action status

Every action has an Action Status data pill in the Data pane. This object data pill contains the current runtime details about the action. The Action Status object consists of a code and message.

Action Status > Code

Integer data pill containing the code returned by the first matching error condition or the last step run. You can return your own code when you create a custom error condition. See Action error evaluation.

Action Status > Message

String data pill containing the message produced by a matching error condition or the last step run. You can return your own message when you create a custom error condition. See Action error evaluation.

Getting started with actions

Transform the Ask for Approval action into a reusable action that always requires manager approval.

Demonstrates creating actions within a flow in the Flow Designer, and testing a flow.

Before you begin

Role required: admin

Note:
While Workflow Studio is designed to use the action_designer and delegated_developer roles in most scenarios, this tutorial uses the admin role to illustrate functionality without requiring additional roles to set up records and approve requests.

Complete the steps in Getting started with flows. This tutorial replaces the Ask for Approval action in the Expense Approval flow.

About this task

Actions are made up of:

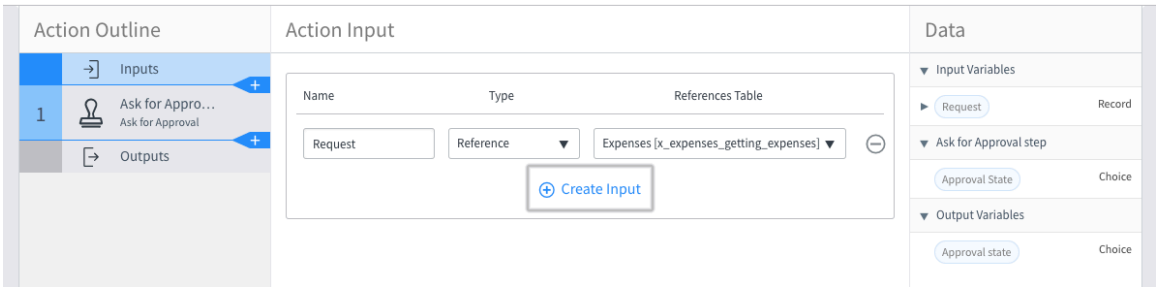
- Inputs: Data variables used in your action.
- Steps: Operations on the inputs or results from a prior step that generate data that can be used in later steps.
- Outputs: Data variables that represent the results of the action. These results are available to other actions in a flow.

Unlike the core Ask for Approval action where flow designers must manually configure the approval rules, this custom action always uses the same approval rules when added to a flow. You might create a reusable action if your flow designers often use an action with the same configuration. For example, if your flow designers always use the request manager approval

and due date options, this action automatically uses them and therefore requires less flow configuration.

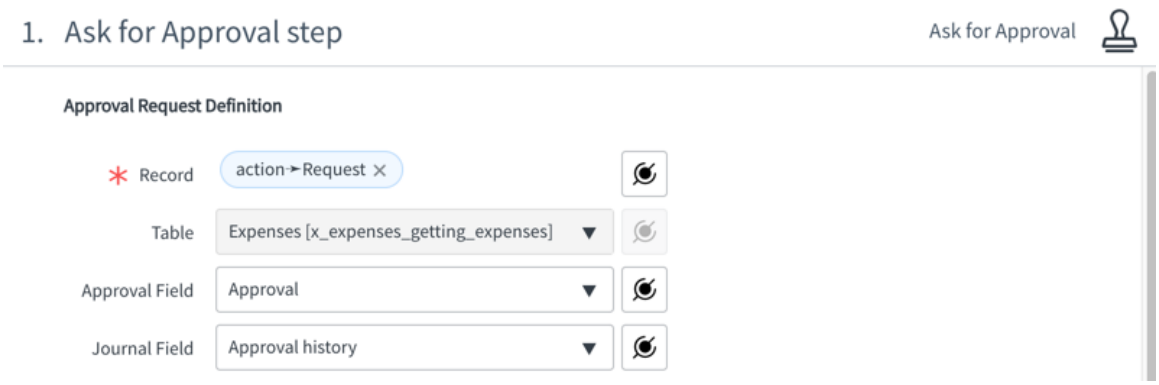
Procedure

1. Open the Expenses Getting Started application in Studio.
   Alternatively, you can navigate to **Process Automation** > **Flow Designer** and select **New Action** to access Workflow Studio in the platform. In the Action Properties, select your scoped application in the **Application** field.
2. Create an action.
   a. Select **Create Application File**.
   b. Under Workflow Studio, select Action and select **Create**.
   c. In the **Name** field, enter `Ask for Manager Approval`.
   d. In the **Description** field, enter `Approve or reject a request based on manager approval or rejection. Allow manual approvers to be added.`
   e. Select **Submit**.
   An Ask for Manager Approval action is created in the Expenses Getting Started scope.
3. Define the inputs in the Ask for Manager Approval action.
   a. Select **+ Create Input** and add the following values.
      - Name: **Request**
      - Type: **Reference**
      - Reference Table: **Expenses [x_expenses_getting_expenses]**



   This input enables you to reference any field or record from the Expenses table. Use the data pills on the right-hand side to add the record or its fields to action steps.
4. Add an Ask for Approval step.
   a. Select the **+** underneath Inputs in the Action Outline.
   b. Select **Ask for Approval**.
   c. Complete the fields in the Ask for Approval step.

- Record: Under the **Input Variables** category, drag the **[Request]** data pill from the right-hand pane.
- Table: Set to **Expenses [x_expenses_getting_expenses]**.
- Approval Field: Set to **Approval**.
- Journal Field: Set to **Approval history**.

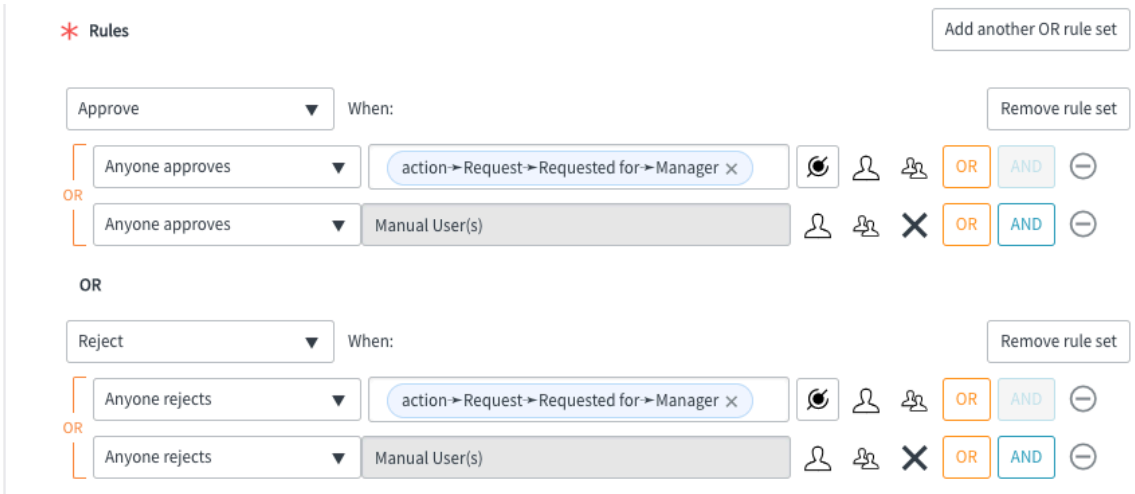


d. Define rules in the Ask for Approval step.
You can use the data pill picker, or drag the data pills from the right-hand pane to select the data you need.

- **[Approve]** when **[Anyone approves]** from the field **[action->Request->Requested for->Manager]**, **[OR]**
- **[Anyone approves]** from the field **[Manual User(s)]**.

Select **Add another OR rule set** to define rejection rules:

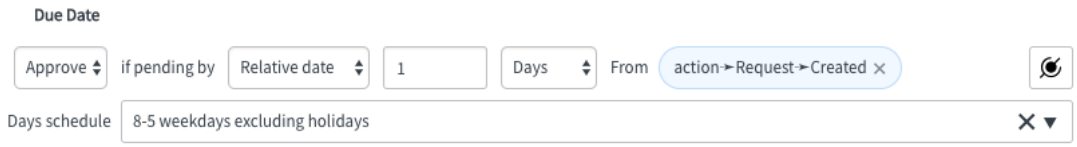
- **[Reject]** when **[Anyone rejects]** from the field **[action->Request->Requested for->Manager]**, **[OR]**
- **[Anyone rejects]** from the field **[Manual User(s)]**.



e. Define a due date in the Ask for Approval step.

- **[Approve]** if pending by **[Relative date] [1] [Days]** from **[action->Request->Created]**.
- Days schedule **[8-5 weekdays excluding holidays]**.

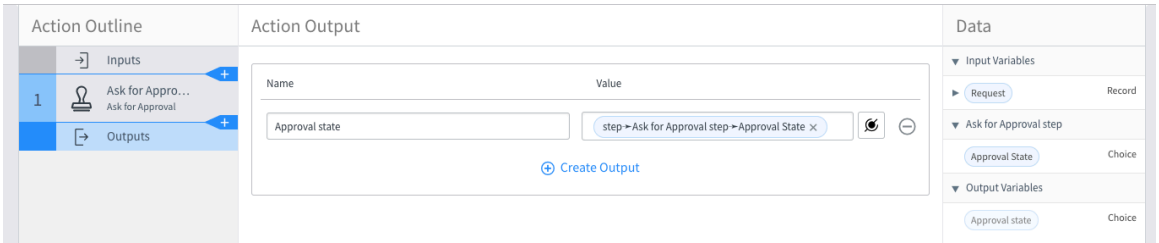
This due date automatically approves all requests that haven't been approved or denied within one day from when the request was created.



5. Define the outputs in the Ask for Manager Approval action.
   Adding an output makes data available to a flow. For example, this action outputs the approval state of the record.

   a. Select **+ Create Outputs** and add the following values.

      - Name: **Approval state**
      - Value: In the right-hand pane, expand the **Ask for Approval step** category and drag the **[Approval State]** data pill.



   b. Select **Save**.

6. Add a custom icon for your application that displays in Workflow Studio.
   All actions in the application scope use the custom icon.

   a. In Studio, navigate to **File** > **Settings**.
   The application settings open.

   b. In the **Logo** field, select **Click to add...**.

   c. Upload an icon to use with your reusable actions.

7. Test the reusable action within your flow.

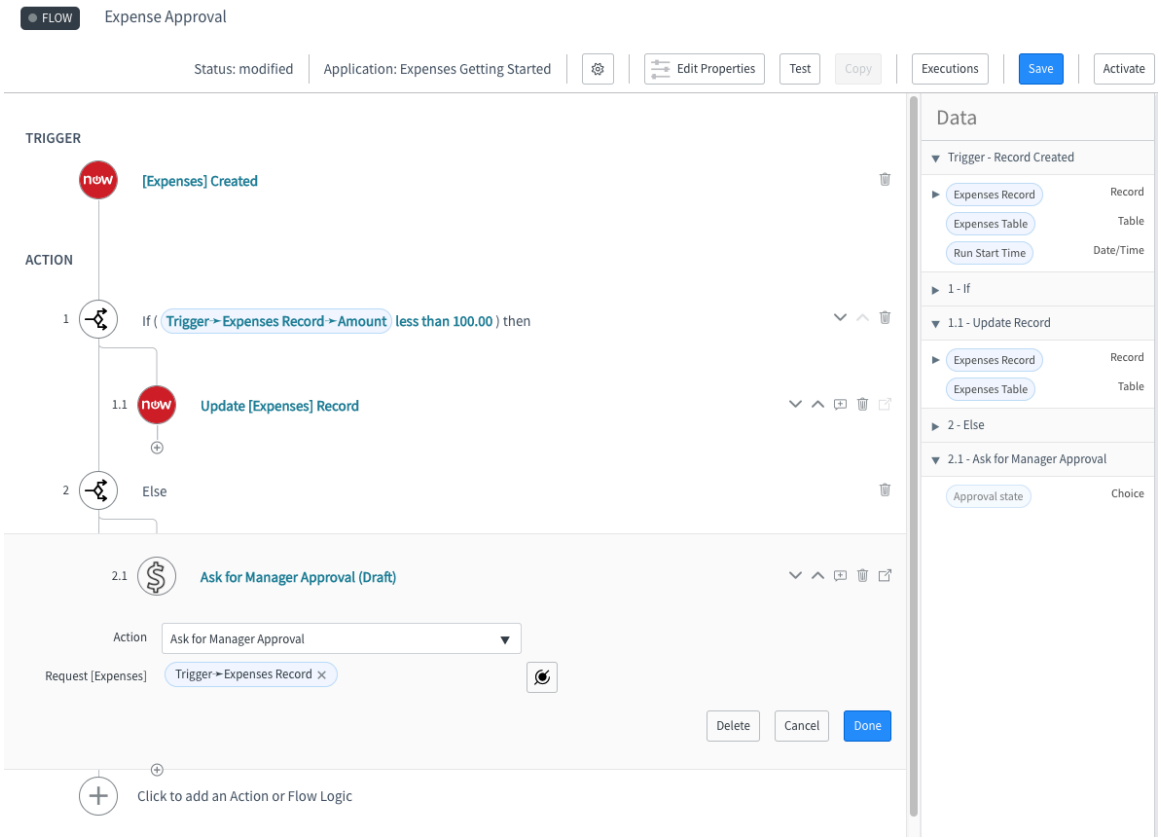
   a. Return to the Expense Approval flow.

   b. Remove the 2.1 Ask for Approval action from the flow.
   This action is replaced by the reusable Ask for Manager Approval action.

   c. Set **Show draft actions** to true.

   d. Add the Ask for Manager Approval action to your flow.

   e. In the right-hand pane, expand the **Trigger - Record Created** category and drag the **[Expenses Record]** data pill into the **Request [Expenses]** field.



8. Select **Save**.
9. Test the flow using a record with an amount below the designated limit.

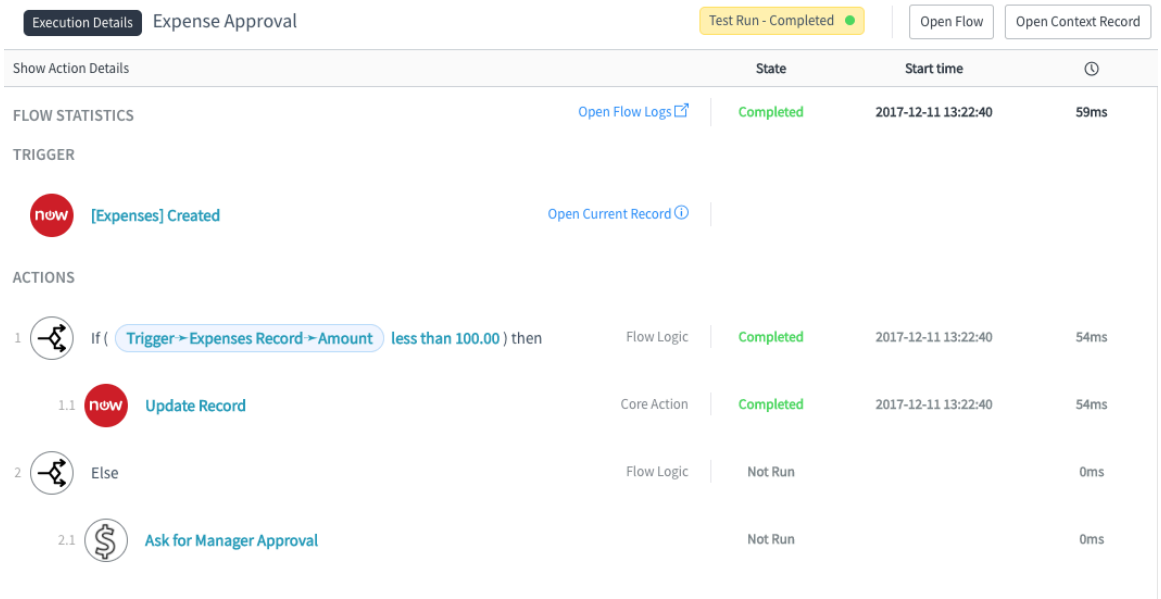
   a. From the flow, select **Test**.
   The Test flow modal appears.

   b. In the **Record** field, select a record that you created in earlier steps, has a value under the 100.00 limit in the **Amount** field, and you haven't used to run tests.

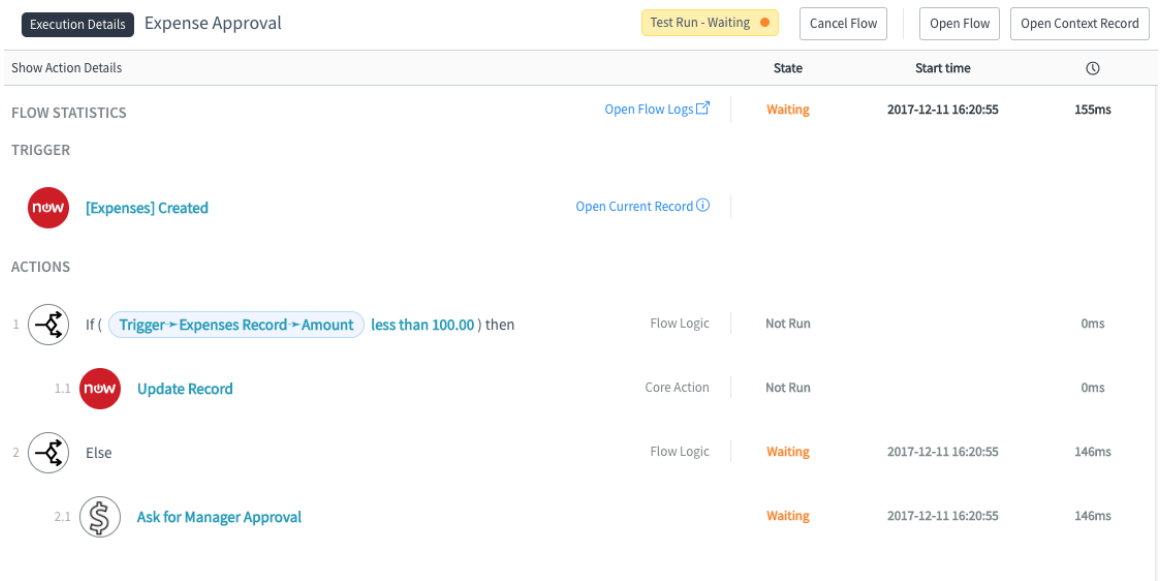
   c. Select **Run Test**.

   d. After the flow executes, select **Flow has been executed. To view the flow, click here**.
   The Execution Details open.

   Because the amount is less than 100.00, the first condition is met and the request is approved. The Else condition isn't evaluated.



10. Test a record with an amount over the designated limit and verify that you have not already run a test on the test record.
Because the amount is over the designated limit, the second condition is evaluated.

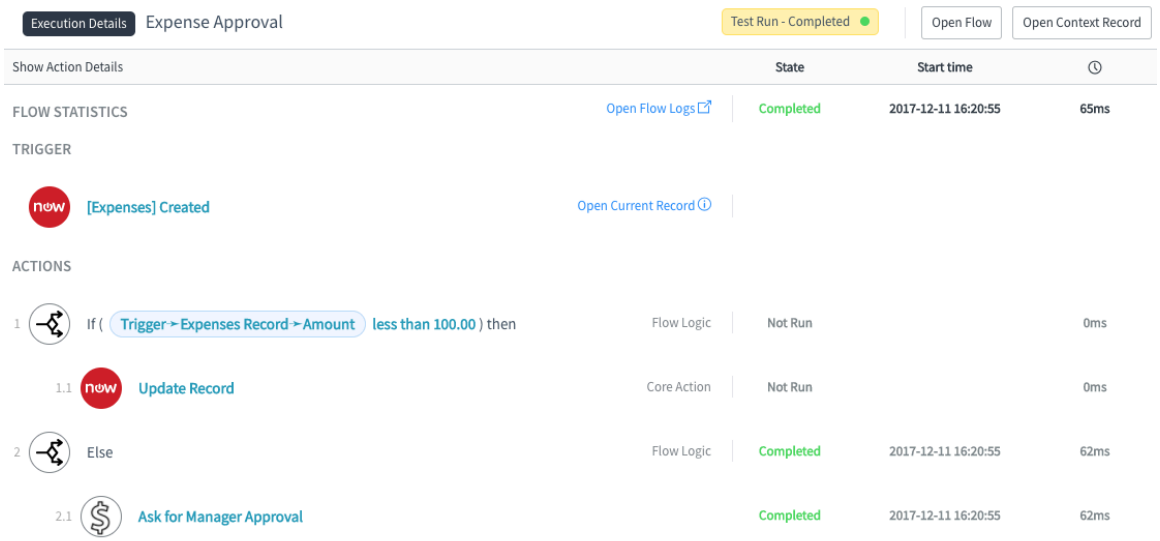


11. Approve the request.

a. Navigate to the test record and change the value of the **State** field in the Approvers related list to **Approved**.

b. Navigate back to the flow execution details and refresh the browser.

Because the request is approved, the flow completes.



12. Navigate to the Ask for Manager Approval action and select **Publish**.
    Publishing an action enables you to activate any flow that uses it.
13. Navigate to the flow and set **Show draft actions** to false.
14. Select **Activate**.
    Activating a flow sets it to run every time the trigger conditions are met.

Result

The Expense Approval flow runs every time a record is created in the Expenses table. Now that the flow is activated and working as expected, you can publish it to the application repository and deploy it to other instances.

Create an action in Workflow Studio

Create a reusable component to automate one or more steps of a process.

Before you begin

- Set up an application in Guided Application Creator to store Workflow Studio content.
- Role required: flow_designer, action_designer or admin

About this task

Action designers should know the application table structure and be aware of any existing business logic associated with the target tables of an action. Be sure to disable any conflicting business rules or workflows before creating an action.

Creating a custom application to contain your Workflow Studio content enables you to deploy it using the application repository or the ServiceNow Store.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select **New** > **Action**.
3. Fill in the Action Properties and click **Submit**.

Field	Description
Action name	Enter a unique name for your action.
Description	Description of the action.
Application	Application scope of the action.
Domain	Domain scope of the action. For more information about domain separation, see Domain separation explained.
Additional properties	
Accessible From	Accessible from all application scoped or only within the specified application scope.
Category	Defined category within the application scope for an action. Beginning with the Xanadu release, you can create a custom category to organize your actions.
Protection	Select whether the action is read-only. You can only select a value when you create the action in an application scope you own. The default value is None.
Action annotation	Help text that appears under the action title in Workflow Studio to help action authors understand what the action does when used in a flow.

An empty action opens.

4. Define action inputs to make data available to the action steps and outputs.

   a. Select **+ Create Input** and complete the fields.

Field	Description
Name	Name of the input. This value is used as the name of the data pill in the right-hand pane. **Note:** Action input names can't include any of the following reserved system names: ▪ `sys_id` ▪ `sys_created_by` ▪ `sys_created_on` ▪ `sys_updated_on` ▪ `sys_updated_by` ▪ `sys_mod_count`
Type	Data type of the input. For supported data types, see Workflow Studio input and output data variables.
Reference Table	Reference table for the data type. Only required for the following data types:

Field	Description
	▪ Records ▪ Reference
Advanced options	Inputs provide advanced options based on their data type. All inputs have advanced options to add a hint or provide a default value. Use advanced options to guide flow designers through adding and configuring an action to a flow. For example, create a choice input to provide flow designers with a pre-defined list of configuration options to choose from. For more information about the configuration options available to particular data types, see field types. Click to view the advanced options and define values.

Inputs are represented as data pills in the right-hand pane. You can add inputs to steps and outputs in the flow by dragging and dropping data pills.

5. Add an action step to perform an operation on the action inputs.
   a. Click the **+** underneath Inputs in the Action Outline.
   b. Select the step you would like to perform.
   c. Complete the fields in the step.
6. For **If this step fails**, select the action error evaluation behavior you want the step to take.

Option	Description
Stop the action and go to error evaluation	Stop running the action at the current step and go to error evaluation. The Step Status object contains the error information returned by the step.
Don't stop the action and go to the next step	Ignore the failure and continue running the action from the next step. The Step Status object contains the error information returned by the step. Action error evaluation runs regardless of whether the action continues running.

7. Add action outputs to make data available to a flow.
   a. Select **+ Create Outputs** and complete the fields.

Field	Description
Name	Name of the output. This value is the name of the data pill in the right-hand pane when the action is added to a flow.

Field	Description
	Note: Action output names can't include any of the following reserved system names: ▪ `sys_id` ▪ `sys_created_by` ▪ `sys_created_on` ▪ `sys_updated_on` ▪ `sys_updated_by` ▪ `sys_mod_count`
Value	Data used previously in the action either in a step or input. Adding a variable to the output makes the value available to the flow.

8. Click **Save**.
   Workflow Studio saves a draft of the action.

What to do next

Test the action until it is ready to be published. See related flows for action to verify that action changes will function in the flows that use the action.

Note:
By default, the system only runs published actions.

Create an action input from a step input

Create an action input based on the data type of a step input. Map the step input value to the new action input.

Before you begin

Role required: admin or action_designer

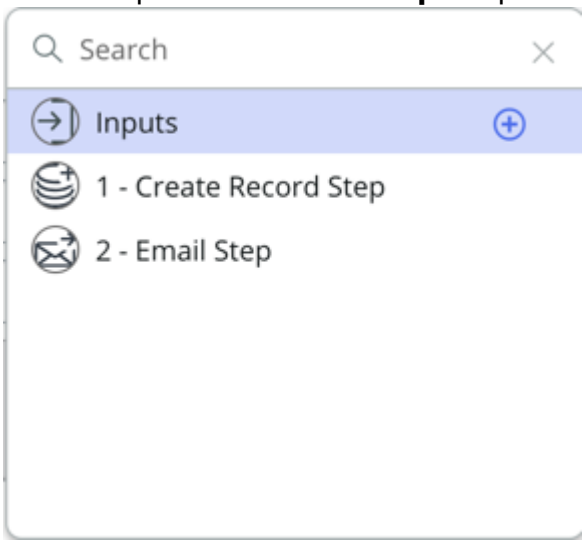
About this task

You can use a step input as a template for an action input. Workflow Studio can create an action input of the same data type as the step input.

Procedure

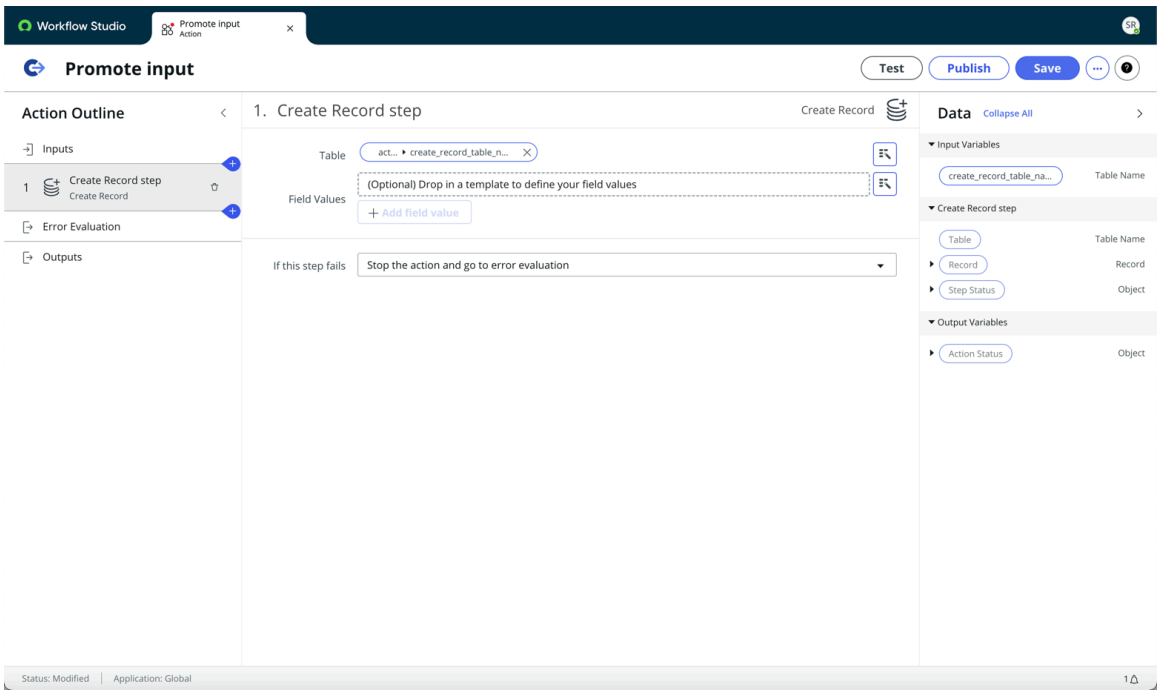
1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select **Action**.
3. Create an action or open an existing action.
4. Add a step or select an existing step.
5. Select the Data Pill Picker icon for the step input you want to use as a template for an action input.

6. Select the plus icon next to the **Inputs** option.



Result

Workflow Studio creates an input named after the step and input type. For example create_record_table_name, which is an input created for the Create Record step and the Table input. The action input is of the same data type as the step input, for example Table Name. The step input is mapped to the new action input, and the action input is available from the Data pane.



Test an action

Test an action before publishing it for other users.

Before you begin

- Create an action and save it.
- Role required: flow_designer, flow_operator, action_designer, or admin.

About this task
A user with the flow_designer role should always test actions on non-production instances containing relevant demonstration data because testing an action creates or changes records on the instance.

Procedure
1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Click the **Actions** tab and select the action that you want to test.
3. Click **Test**.
   The system displays the Test Action dialog box.
4. Fill in the fields for the action.

   Note:
   Complete all mandatory fields in the Test Action dialog box.

5. Click **Run Test**.

   Note:
   Select the **Run test in background** option to test an action asynchronously in the background.

   If you select the **Run test in background** option, the execution details are displayed only after the execution is completed asynchronously in the background.

What to do next
Click `Action has been executed. To view the action, click here` to view the action execution details. See Flow execution details for information about the executions.

Note:
Users must have the flow_operator or admin role to view the executions.

Copy an action

Copy an action to give it a new name and move it to another application scope.

Before you begin
Role required: action_designer or admin

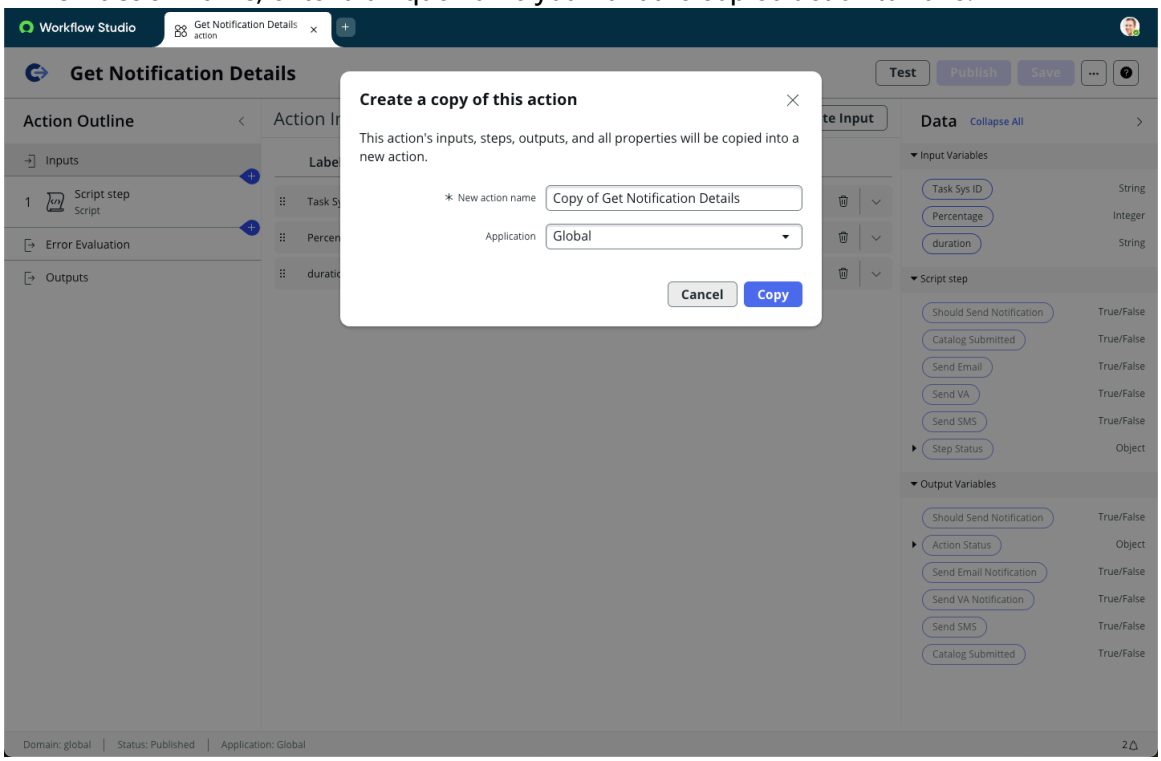
About this task
You can copy a custom action that you created to give it a new name or move it to another application scope. The new action has the same action properties, inputs, steps, and outputs as the source action.

You can't copy the default actions created by ServiceNow, Inc., nor can you copy actions that have a protection policy. You must have write access to an action to copy it.

Procedure
1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Actions**.
3. Select the action that you want to copy.
4. Click the more actions icon (⋯) and select **Copy action**.

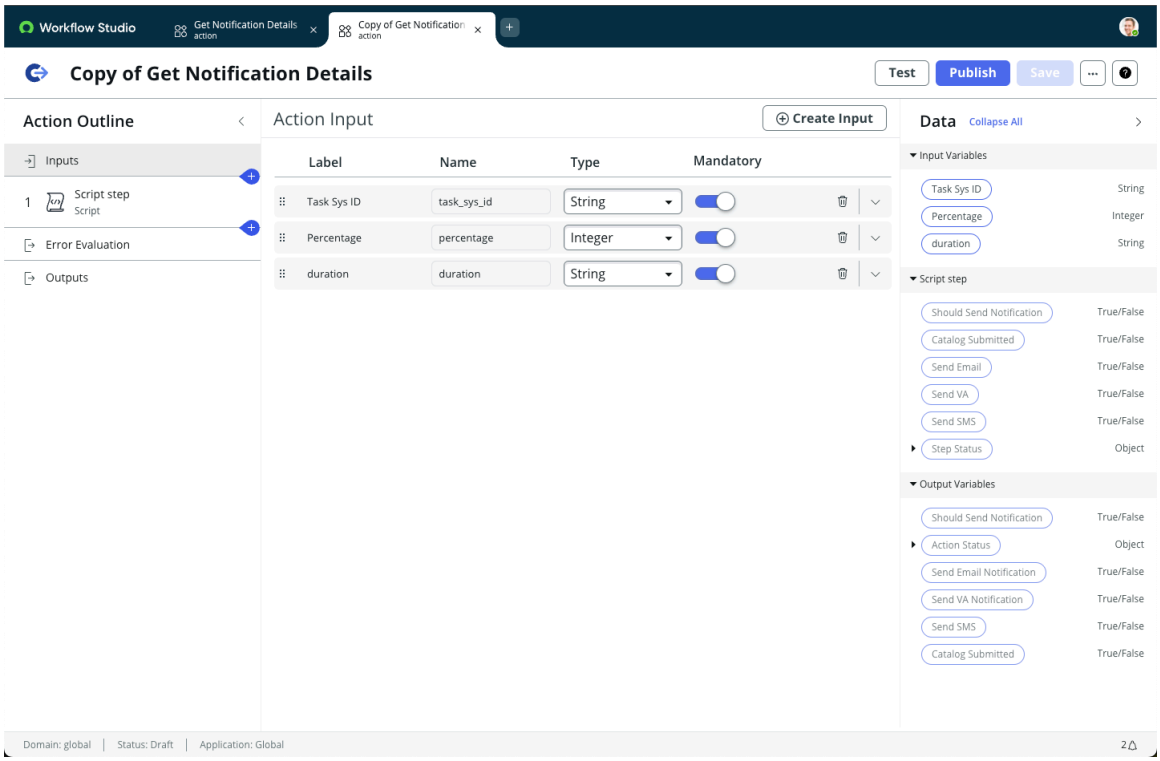
Note:
If the **Copy action** option is not visible, then you don't have permission to copy the action. This could be because the action has a protection policy or because you lack the necessary user role or developer permissions.

5. In **New action name**, enter a unique name you want the copied action to have.



6. **Optional:** From **Application**, select the application scope where you want to copy the action.
7. Select **Copy**.

Result



Workflow Studio opens the new action.

Dynamic inputs

Generate a list of action inputs or input values each time that someone configures the action in a flow. Dynamic inputs can display a list of related action inputs, a list of fields from a record, or a list of options available in a choice list.

> **Note:**
> Dynamic inputs are not available in the base system. To use dynamic inputs in Workflow Studio, you must Request an Integration Hub plugin. Dynamic inputs are part of the ServiceNow Flow Designer - Introspection [com.glide.hub.flow_designer_introspection] plugin.

During flow design, dynamic inputs retrieve values and display them as inputs within an action dynamically. These types of dynamic inputs are available to action authors.

Dynamic Choice

The Dynamic Choice type input gathers a list of choice values to display. For more information on building a dynamic choice, see Create a data gathering action for a dynamic choice.

Dynamic Inputs

The Dynamic Inputs type input gathers a list of additional action inputs to display. Use Dynamic Inputs to add arbitrary inputs to an action. For more information on building dynamic inputs, see Create a data gathering action for a dynamic inputs type input.

Dynamic Template

The Dynamic Template type input gathers a list of fields from a dynamically selected record type. Flow authors can then select which fields to include in their flow. Because the list of fields is dynamically generated, you don't have to change the action when fields are added or removed from the source table. For more

information on building a dynamic template, see Create a data gathering action for a dynamic template.

There are three general steps to using dynamic inputs.

1. An action author creates a data gathering action to generate dynamic data.
2. An action author creates an action with a dynamic input and configures the dynamic input to call the data gathering action.
3. A flow author adds the action to a flow and configures the action with dynamically gathered data.

Data gathering actions

A data gathering action collects data to be used by other actions. Data gathering actions are intended to be called from dynamic inputs rather than be added directly to a flow. Data gathering actions typically collect data from third-party systems using a REST call. All data gathering actions must meet these requirements and constraints.

- The action has a script step that contains an output variable of type JSON.
- The action has an output named *output* of type JSON whose value is derived from the script step's JSON output variable.

> **Note:**
> The action can have multiple outputs but can only have one of type JSON.

- The script step formats the JSON output to have a property named `data`.
- The JSON output should not return more than 5000 choice options, field template values, or array element items when the data is intended for a dynamic choice or a dynamic template input.

> **Note:**
> Dynamic choice and dynamic template inputs can only display up to 5000 choice options or 5000 template values from the JSON output.

- The action waits for up to 300 seconds (5 minutes) to gather data before it times out.

> **Note:**
> To change the timeout period for all actions, modify the value of the *sn_flow_designer.sync_action_execution_timeout_in_seconds* system property.

Supported dynamic input data types

Dynamic inputs support a limited number of ServiceNow AI Platform data types. You can use the example JSON to build your own dynamic inputs. You can change the values of the label and name properties to met your needs. The type property must specify a ServiceNow AI Platform data type name. For more information about ServiceNow AI Platform field data types and how to configure them, see Field types.

Dynamic input data types supported

Input data type	Example JSON
Choice	`{ data: [{`

Dynamic input data types supported (continued)

Input data type	Example JSON
	label: 'Choice type input', name: 'choicetype', defaultValue: 'choice_1', type: 'choice', choices: [{ label: 'Choice 1', value: 'choice_1' }, { label: 'Choice 2', value: 'choice_2' }] }] }
Datetime	{ data: [{ label: 'Datetime type input', name: 'datetimetype', type: 'datetime', }] }
Decimal	{ data: [{ label: 'Decimal type input', name: 'decimaltype', type: 'decimal', }] }
Email	{ data: [{ label: 'Email type input', name: 'emailtype', type: 'email', }] }
HTML	{ data: [{ label: 'HTML type input', name: 'htmltype', type: 'html',

Dynamic input data types supported (continued)

Input data type	Example JSON
	}] }
Integer	{ data: [{ label: 'Integer type input', name: 'integertype', type: 'integer', }] }
Password 2	{ data: [{ label: 'Password2 type input', name: 'password2type', type: 'password2', }] }
Reference	{ data: [{ label: 'Reference type input', name: 'referencetype', reference: 'sys_user', type: 'reference', }] }
String	{ data: [{ label: 'String type input', name: 'stringtype', defaultValue: 'abcdef', type: 'string', mandatory: true }] }

General guidelines

Consider dynamic inputs for third-party integrations

Dynamic inputs let you create flows that fetch data dynamically from external sources. In third-party integrations, dynamic inputs can provide data values that pertain to a particular endpoint. For more information on setting up third-party integrations with Workflow Studio, see IntegrationHub.

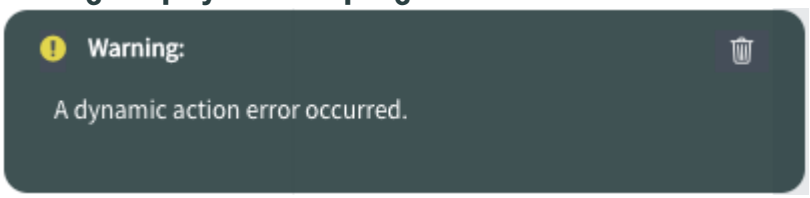
Be aware of the time required to retrieve large amounts of data

By default, dynamic inputs have up to 300 seconds to gather data before they time out. If your data gathering action needs more time to gather data, set the *sn_flow_designer.sync_action_execution_timeout_in_seconds* system property to a higher value. However, don't use long timeout values for interactive flows where an end user must enter or select a value.

Be aware of scripting errors

Because all data gathering actions use a script step, potential errors could occur from scripting. When using scripts to output JSON variables for your dynamic inputs, you may encounter errors that prevent inputs from receiving the JSON values they need. When a dynamic input scripting error occurs, the following warning message may appear.

Message displayed for scripting error



Limit dynamic inputs type inputs to 40 input values

A dynamic inputs type input can only render a certain number of inputs before the JSON object becomes to too big to store in memory. Limiting your dynamic inputs to 40 input values minimizes the chances that you will run out of memory and experience unexpected behaviors such as rendering errors or data truncation.

Limit JSON output to 5000 array items for dynamic templates and dynamic choices

Dynamic choice and dynamic template inputs can only display up to 5000 array items. A dynamic choice can only display up to 5000 choice list options, and a dynamic template can only display up to 5000 field template values. If your data gathering action collects data for a dynamic template or a dynamic choice, restrict the maximum number of array items it returns to 5000. The 5000 array items limit prevents the instance from having performance issues when rendering the choices or field values.

Get started with dynamic inputs

Create a sample action that illustrates using all available types of dynamic inputs in a flow.

Before you begin

Role required: action_designer or admin

Procedure

1. Create credential and connection records for your instance.
   This connection & credential alias will provide the base URL and user account needed to configure the REST steps of your data gathering actions.
2. Create a data gathering action to get table names.
   This data gathering action will provide JSON data for dynamic choice inputs.
3. Create a data gathering action to get field names.
   This data gathering action provides JSON data for dynamic template inputs.
4. Create a data gathering action to add dynamic inputs.
   This data gathering action provides JSON data to create arbitrary dynamic inputs.

5. Create a custom action to test dynamic inputs.
   This custom action illustrates different types of dynamic inputs.

Create credential and connection records for your instance

Create the aliases, connections, and credentials needed to connect to your local instance.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **Connections & Credentials** > **Credentials**.
2. Select **New**, select **Basic Auth Credentials**, and enter these field values.
   a. For **Name**, enter `Local Admin`.
   b. For **User name**, enter a user account with access to Flow Designer and the REST API.

      Example
      For example, enter `admin`.

   c. For **Password**, enter the account password.
3. Select **Submit** to create the credential record.
4. Navigate to **All** > **Connections & Credentials** > **Connection & Credential Aliases**.
5. Select **New** and enter these fields values.
   a. For **Name**, enter Local Instance.
   b. Accept the default value of HTTP for the **Connection type**.
   c. Select **Submit** to create the Connection & Credential Alias record.
6. Select the alias you created.

   Example
   For example, select **Local Instance**.
7. From the Connections related list, select **New**, and enter these field values.
   a. For **Name**, enter `My Instance`.
   b. For Credential, select the basic authentication credential record you created.

      Example
      For example, select the **Local Admin** credential.

   c. For Connection URL, enter the base URL for your instance including the forward slash at the end.
      Include the URL prefix https:// and add a slash character at the end of the URL.

      Example
      For example, `https://example.service-now.com/`.

   d. Select **Submit** to create the HTTP(s) Connection record.

Result
You can use the Local Instance alias to connect to your local instance when configuring REST steps.

Create a data gathering action to get table names

Create a custom data gathering action to fetch a list of table names from the Table API.

Before you begin
Role required: action_designer or admin

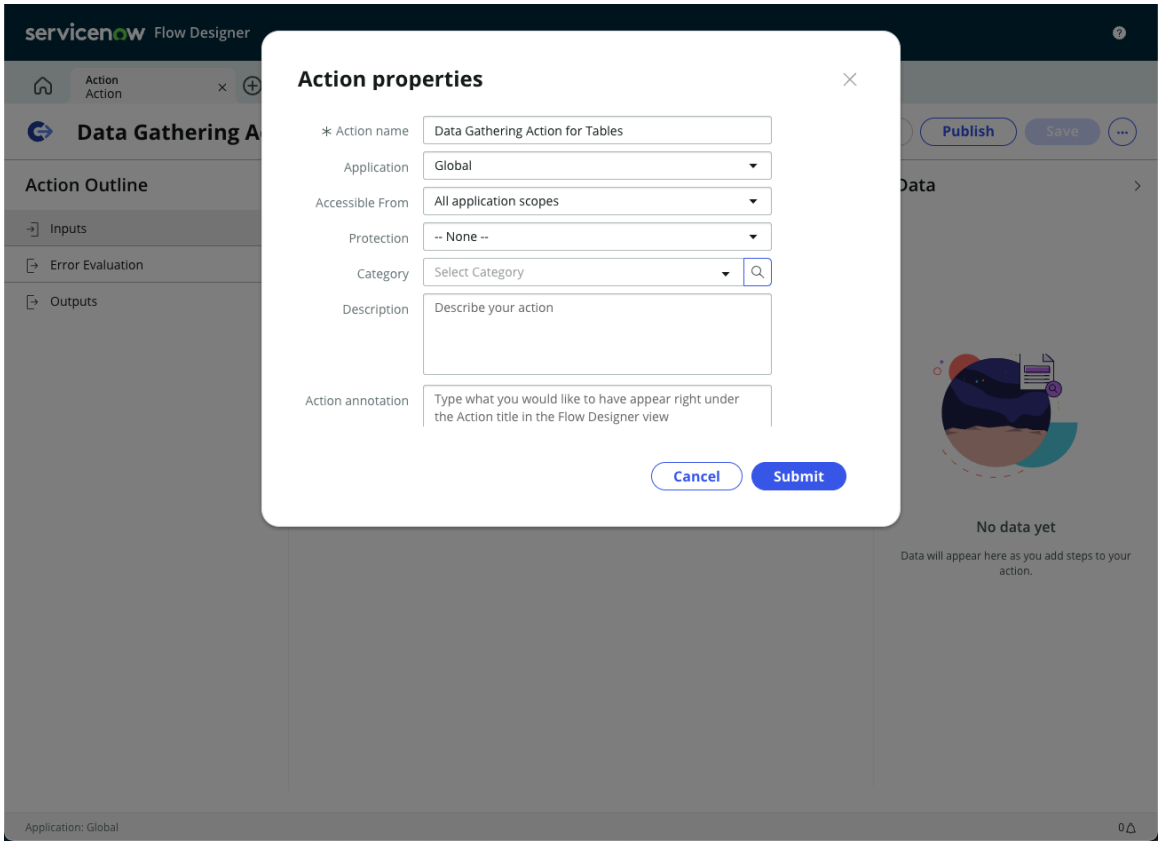
About this task
This data gathering action consists of these elements.

- A REST step to gather table names from the REST Table API
- A script step to construct a JSON payload from the REST step's Response Body
- An action output variable named `output` to store table name and value pairs as a JSON document

This custom action uses a REST API call to duplicate the functionality of the ServiceNow core Look Up Record action and Look Up Records action. Typically, you do not need to create custom actions to perform basic record operations. This action illustrates working with the REST and Script steps to gather table data for dynamic inputs.

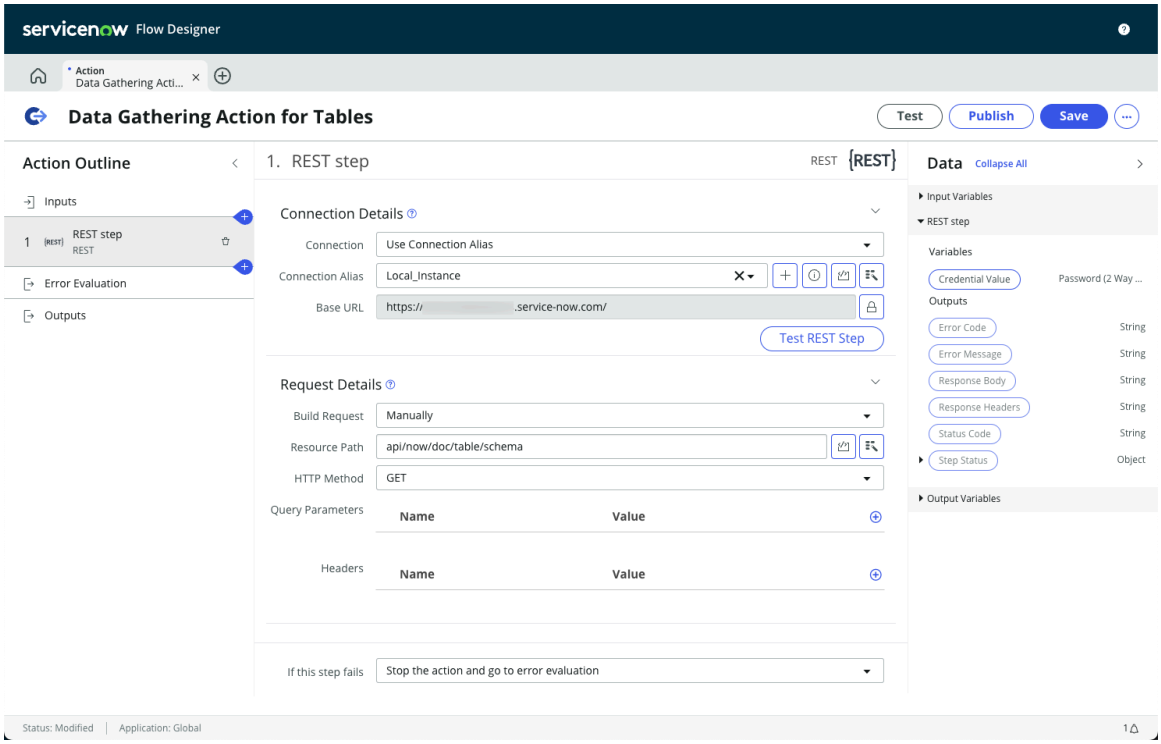
Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Actions**.
3. Select **Create new** > **Action**.
   a. On the Action properties screen, in the **Action name** field, enter `Data Gathering Action for Tables`.
   b. For **Application**, select Global.
   c. Select **Submit**.



4. In the Action Outline under the Inputs section, select the add a new step icon ( ).
5. From the **Integrations** section, select the **REST** step, and enter these field values.

Input	Value
Connection	Leave **Use Connection Alias** selected.
Connection Alias	Select the alias you created earlier, or select the Create new record icon (+) to Create an HTTP(s) connection. **Note:** The **Credential** for the HTTP(s) connection must use Basic authentication credentials. Additionally, the **Connection URL** must be the base URL for your instance, including the forward slash at the end. For more information on connections and credentials, see Getting started with connections and Getting started with credentials.
Build Request	Leave the **Manually** option selected
Resource Path	Enter `api/now/doc/table/schema`
HTTP Method	Enter `GET`



6. In the Action Outline under your REST step, select the Add a new step icon ( ), and select the **Script** step.

   a. In the Input Variables section, select **Create Variable**.

   b. In the **Name** field, enter `payload`.

   c. Next to the **Value** field, select the data pill picker ( ) and select **REST Step** > **Response Body**.

   d. In the **Script** field, enter the following code.

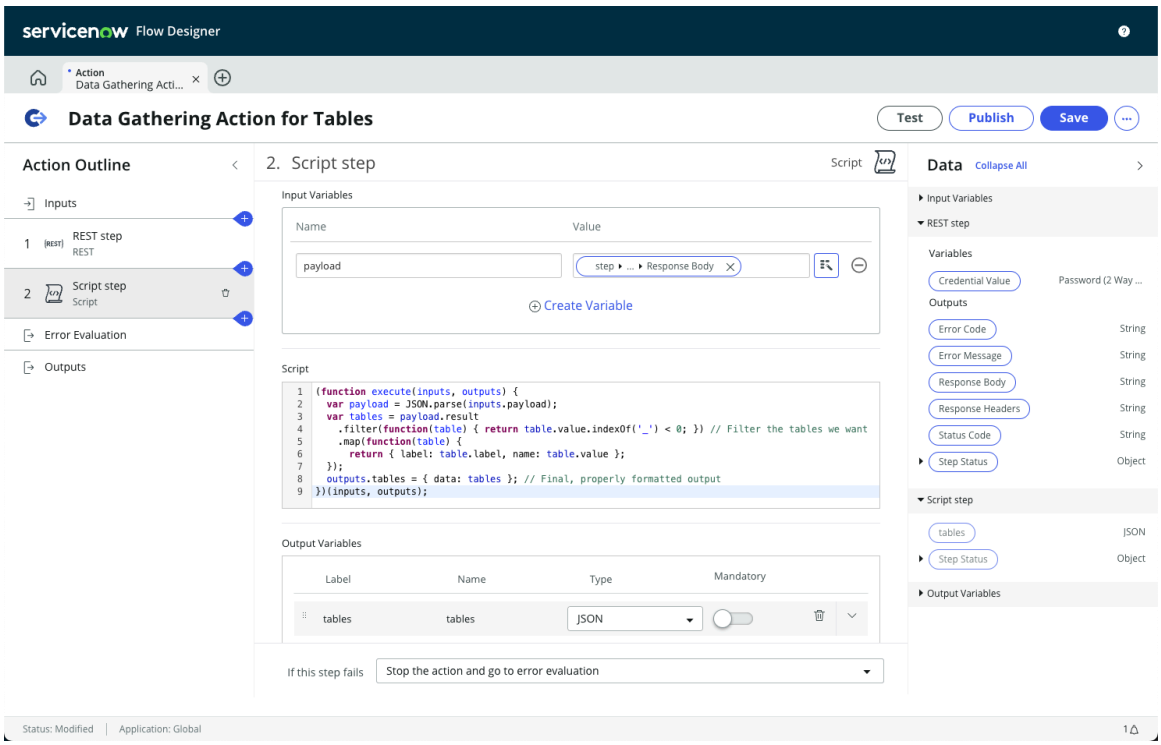
```
(function execute(inputs, outputs) {
  var payload = JSON.parse(inputs.payload);
  var tables = payload.result // Get the value of the result
 array
    .filter(function(table) { return table.value.indexOf('_')
 < 0; }) // Filter the tables we want
    .map(function(table) {
      return { label: table.label, name: table.value }; //
 Set values for label and name
  });
  outputs.tables = { data: tables }; // Final, properly
 formatted output
})(inputs, outputs);
```

   The REST API returns a JSON-formatted string that describes each table with the image, reference, rawLabel, selected, label, missing, used, and value properties. The label and value properties provide the values needed by for a dynamic choice. A dynamic choice requires that each choice option have a label and a name. This script maps the value property to a name property.

e. In the Output Variables section, select **Create Variable**.

f. In the **Label** and **Name** fields, enter `tables`.

g. In the **Type** field, select `JSON`.



7. In the Action Outline, select **Outputs**.

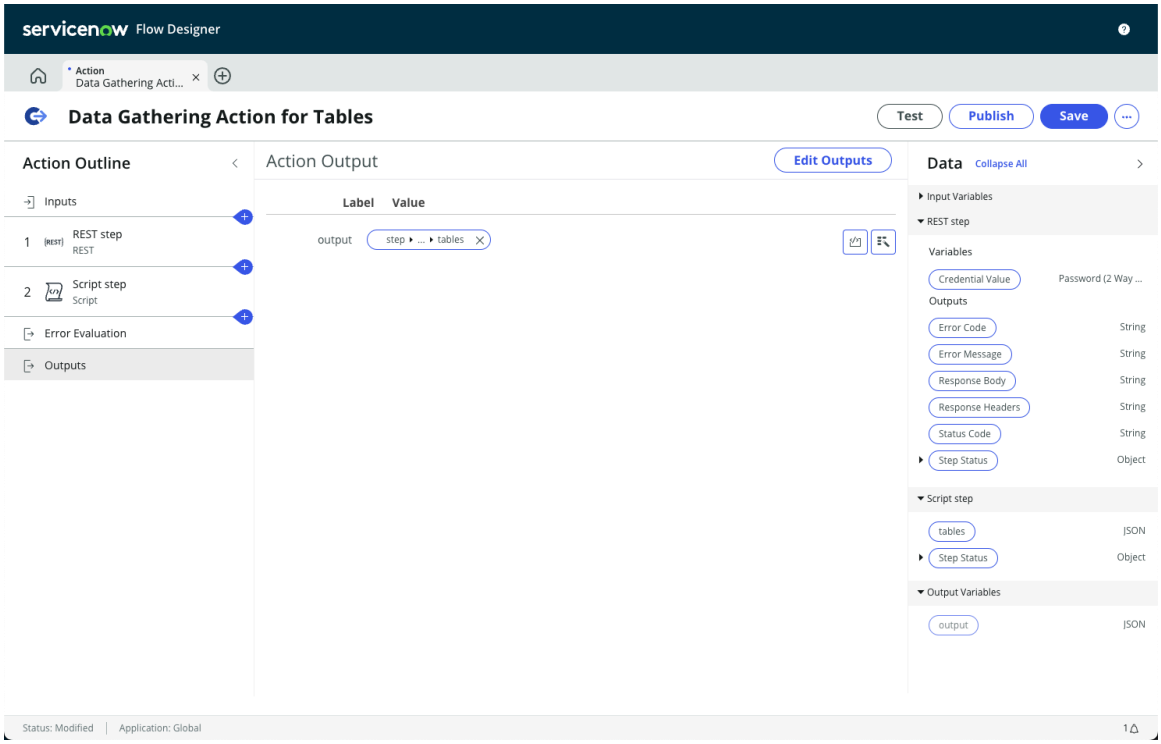
   a. Select **Create Output**.

   b. In the **Label** and **Name** fields, enter `output`.

   c. In the **Type** field, select `JSON`.

   d. Select **Exit Edit Mode**.

   e. Next to the **Value**, select the data pill picker ( ) and select **Script step** > **tables**.



8. In the Action header, select **Save** and then select **Test** to test the action.

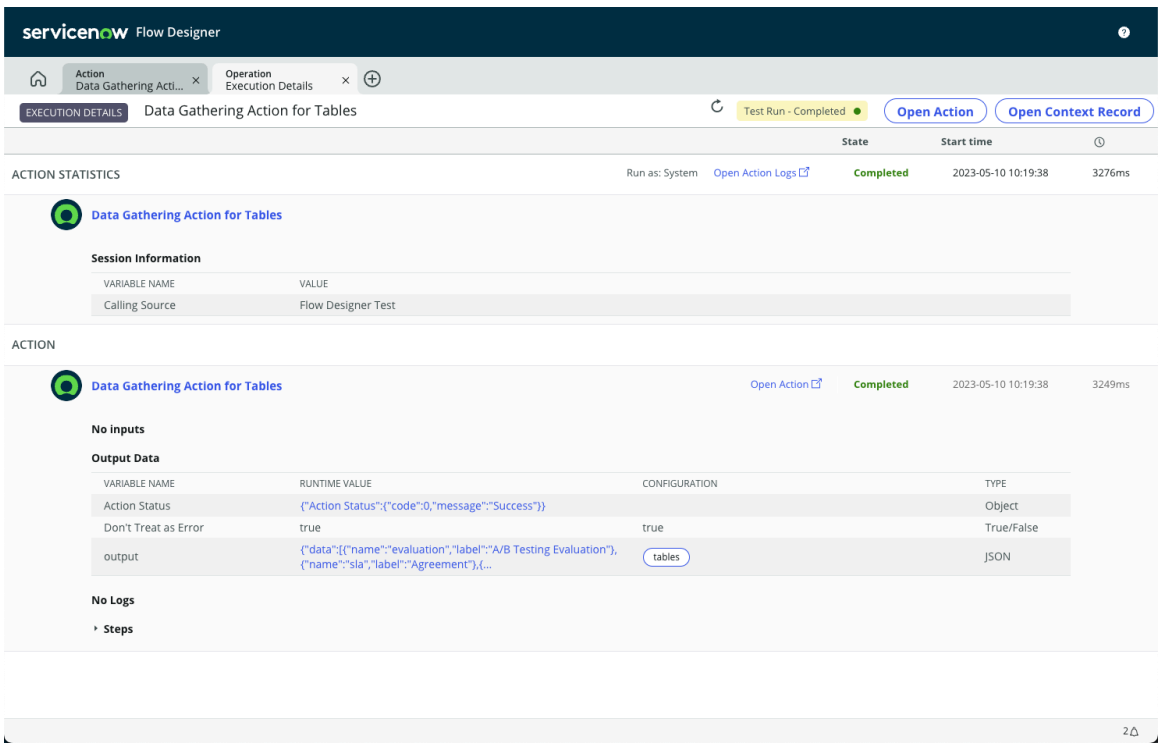
   a. Select **Run Test**.

   b. View the action's execution details.
   Your data gathering action runs successfully if the runtime value for `tables` is a complex object containing an array of key-value pairs for `label` and `name` as shown in the following abbreviated example.

```
{
  "data": [
    {
      "name": "evaluation",
      "label": "A/B Testing Evaluation"
    },
    {
      "name": "sla",
      "label": "Agreement"
    },
    {
      "name": "announcement",
      "label": "Announcement"
    },
    {
      "name": "cmdb",
      "label": "Base Configuration Item"
    },
    {
      "name": "checklist",
      "label": "Checklist"
    },
    {
      "name": "collaborator",
      "label": "Collaborator"
```

```
    },
    {
      "name": "conflict",
      "label": "Conflict"
    },
    {
      "name": "clone",
      "label": "Database Clone"
    },
    {
      "name": "dsl",
      "label": "DML"
    },
    {
      "name": "global",
      "label": "Global"
    },
    {
      "name": "goal",
      "label": "Goal"
    },
    {
      "name": "incident",
      "label": "Incident"
    },
    {
      "name": "interaction",
      "label": "Interaction"
    },
    {
      "name": "map",
      "label": "Map"
    },
    {
      "name": "ola",
      "label": "OLA"
    },
    {
      "name": "problem",
      "label": "Problem"
    },
    {
      "name": "question",
      "label": "Question"
    },
    {
      "name": "reminder",
      "label": "Reminder"
    },
    {
      "name": "instance",
      "label": "ServiceNow Instance"
    },
    {
      "name": "label",
      "label": "Tag"
    },
```

```
    {
      "name": "task",
      "label": "Task"
    },
    {
      "name": "taxonomy",
      "label": "Taxonomy"
    },
    {
      "name": "ticket",
      "label": "Ticket"
    },
    {
      "name": "topic",
      "label": "Topic"
    },
    {
      "name": "expert",
      "label": "Wizard"
    }
  ]
}
```

9. In the Action header, select **Publish** to make this action available to other flows and actions within the Global scope.

Create a data gathering action to get field names

Create a data gathering action to get a list of fields from a table.

Before you begin

Role required: action_designer or admin

About this task

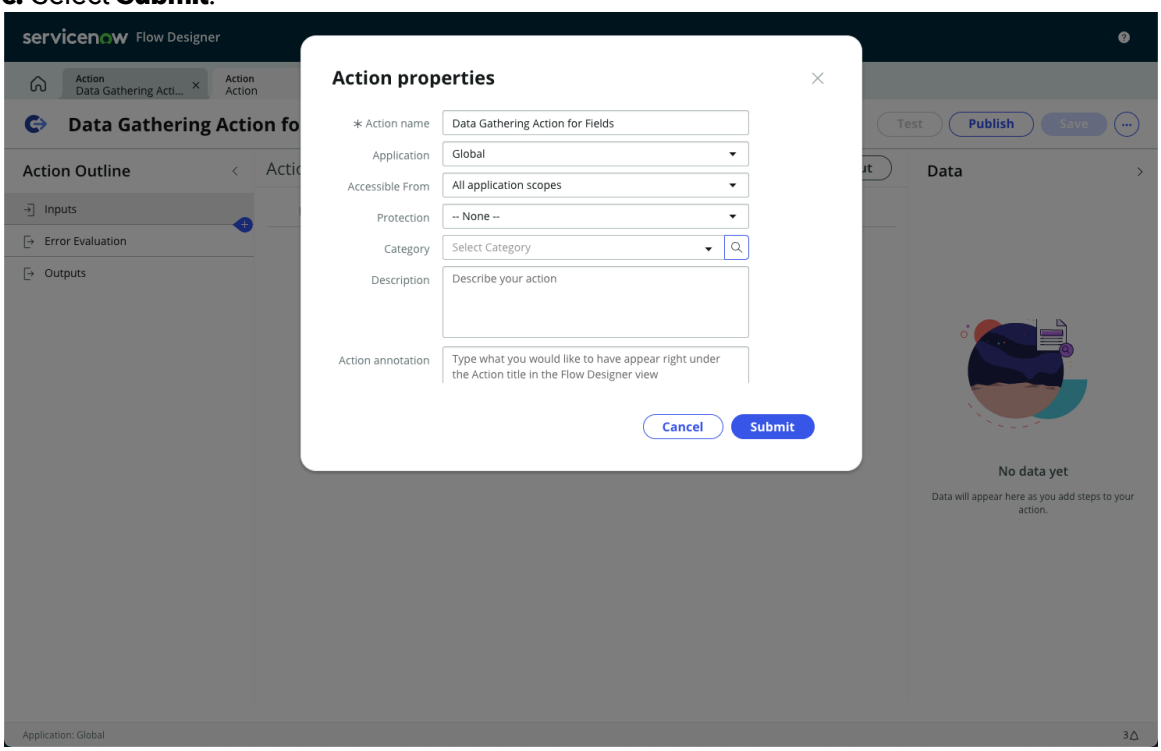
This data gathering action consists of these elements.

- A REST step to gather fields from the REST Table API
- A script step to construct a JSON payload from the REST step's Response Body
- An action output variable named `output` to store field name and value pairs as a JSON document

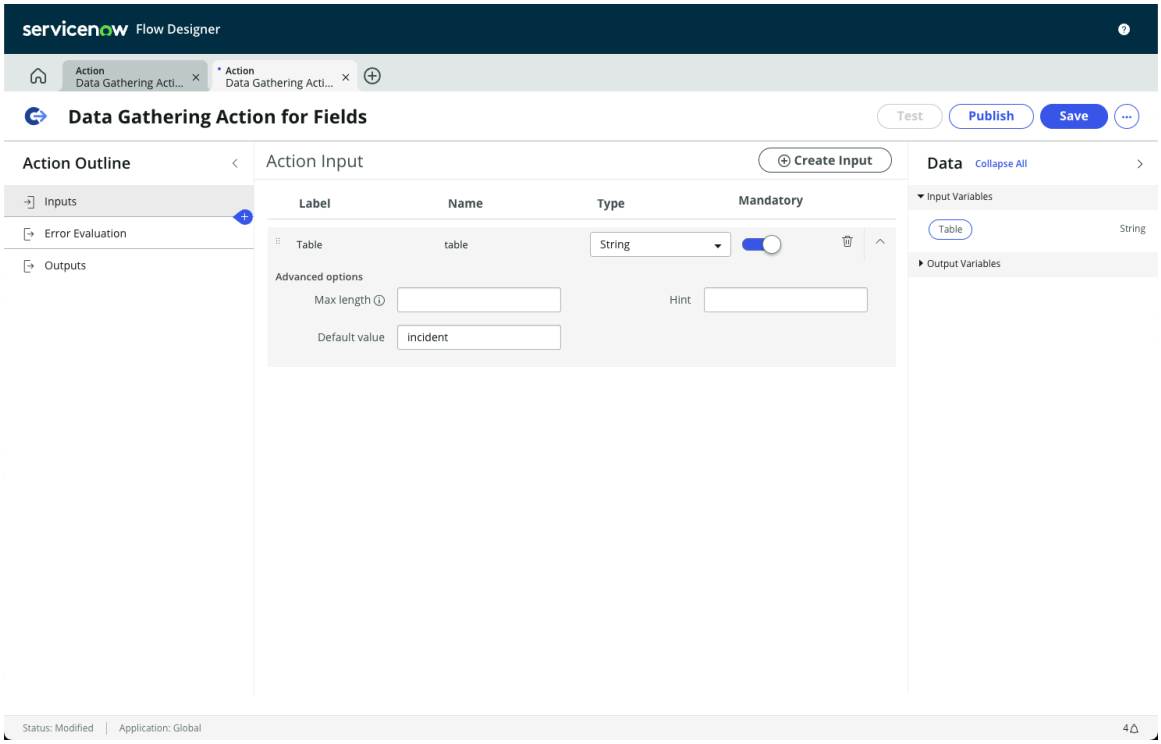
This custom action uses a REST API call to duplicate the functionality of the ServiceNow core Look Up Record action and Look Up Records action. Typically, you do not need to create custom actions to perform basic record operations. This action illustrates working with the REST and Script steps to gather field data for dynamic inputs.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Actions**.
3. Select **Create new** > **Action**.
   a. On the Action properties screen, in the **Action name** field, enter `Data Gathering Action for Fields`.
   b. For **Application**, select Global.
   c. Select **Submit**.



4. From the Action Outline, select **Inputs**.
   a. In the Action Input header, select **Create Input**.
   b. In the **Label** and **Name** fields, enter `Table`.
   c. In the **Type** field, select **String**.
   d. Toggle the **Mandatory** slider so that it is active.
   e. Select the Toggle Advanced Inputs icon to show the advanced inputs.
   f. For **Default value**, enter `incident`.



5. In the Action Outline under the REST step, select the Add a new step icon ( ).
6. From the **Integrations** section, select the **Script** step, and enter these field values.

   a. In the Input Variables section, select **Create Variable**.

   b. In the **Name** field, enter `table`.

   c. Next to the **Value**, select the data pill picker ( ) and select **Action** > **Table**.

   d. In the **Script** field, enter the following code.

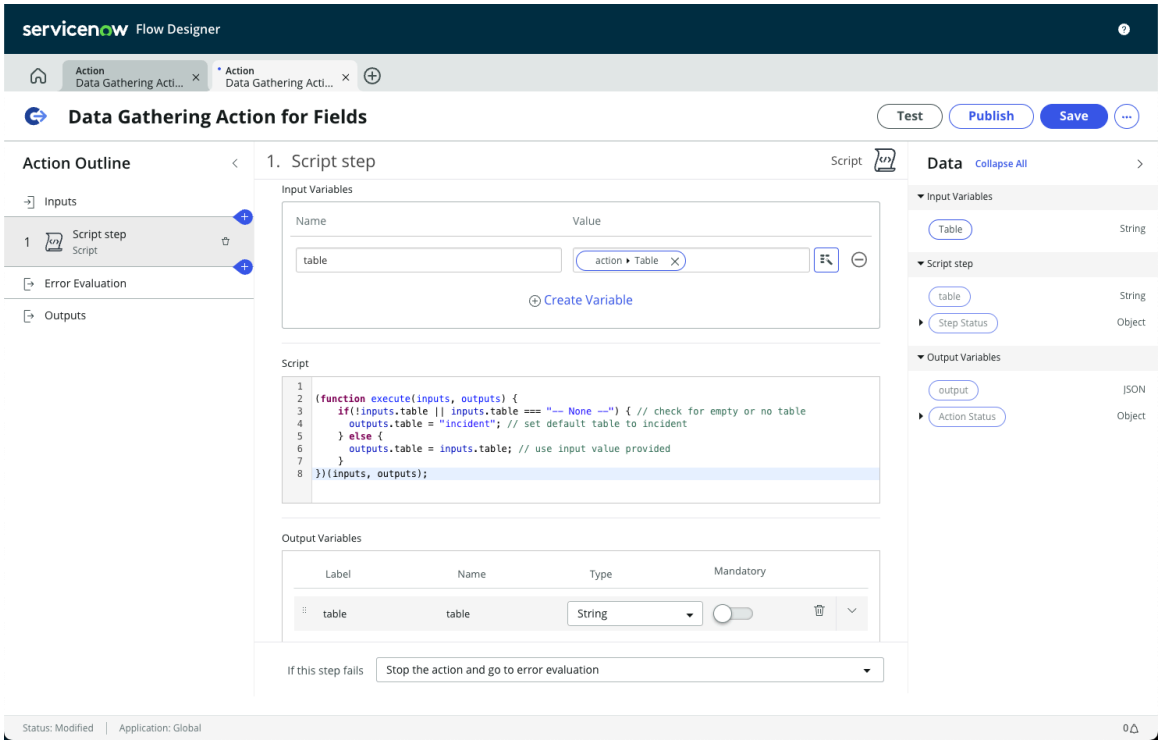
```
(function execute(inputs, outputs) {
    if(!inputs.table || inputs.table === "-- None --") { //
 check for empty or no table
      outputs.table = "incident"; // set default table to
 incident
    } else {
      outputs.table = inputs.table; // use input value
 provided
    }
})(inputs, outputs);
```

   This script ensures that the REST step always has a table name.

   e. In the Output Variables section, select **Create Variable**.

   f. In the **Label** and **Name** fields, enter `table`.

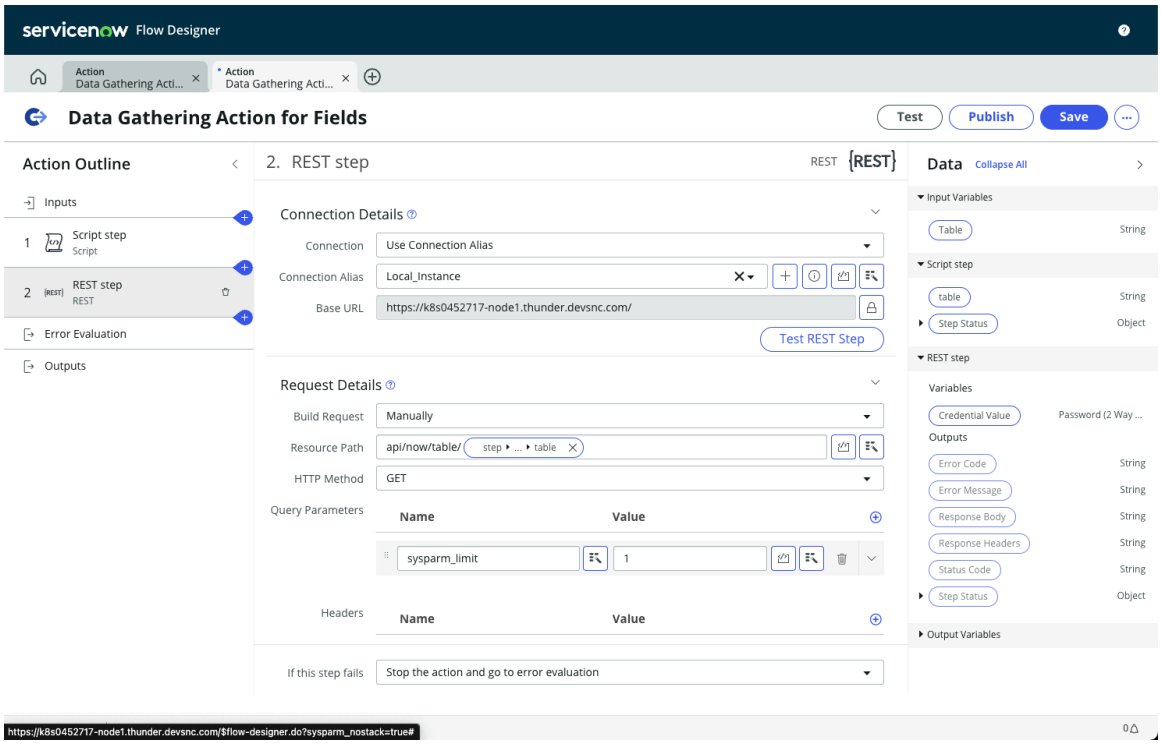
   g. In the **Type** field, select **String**.



7. In the Action Outline under the Inputs section, select the Add a new step icon ( ).
8. From the **Integrations** section, select the **REST** step, and enter these field values.

Input	Value
Connection	Leave **Use Connection Alias** selected.
Connection Alias	Select the alias you created earlier, or select the Create new record icon ( ) to Create an HTTP(s) connection. **Note:** The **Credential** for the HTTP(s) connection must use Basic authentication credentials. Additionally, the **Connection URL** must be the base URL for your instance, including the forward slash at the end. For more information on connections and credentials, see Getting started with connections and Getting started with credentials.
Build Request	Leave **Manually** selected.
Resource Path	Enter `api/now/table/` and then select the data pill picker ( ). Select **Script step** > **Table**.
HTTP Method	Enter `GET`

Input	Value
Query Parameters	Add this query parameter to make the REST API only return one record. ○ **Name**: sysparm_limit ○ **Value**: 1



9. In the Action Outline under the REST step, select the Add a new step icon ( ).
10. From the **Integrations** section, select the **Script** step, and enter these field values.

    a. In the Input Variables section, select **Create Variable**.

    b. In the **Name** field, enter `payload`.

    c. Next to the **Value**, select the data pill picker ( ) and select **REST Step** > **Response Body**.

    d. In the **Script** field, enter the following code.

```
(function execute(inputs, outputs) {
  var payload = JSON.parse(inputs.payload);
  var fields = Object.keys(payload.result[0]) // Get first
 record of array
    .map(function(property) {
      return {
        label: property.charAt(0).toUpperCase() +
 property.slice(1).replace(/_/g, ' '), // Create label from
 field name
        name: property, // Set name to field name
        value: '' // Set value to empty so that dynamic
 template can set value
```

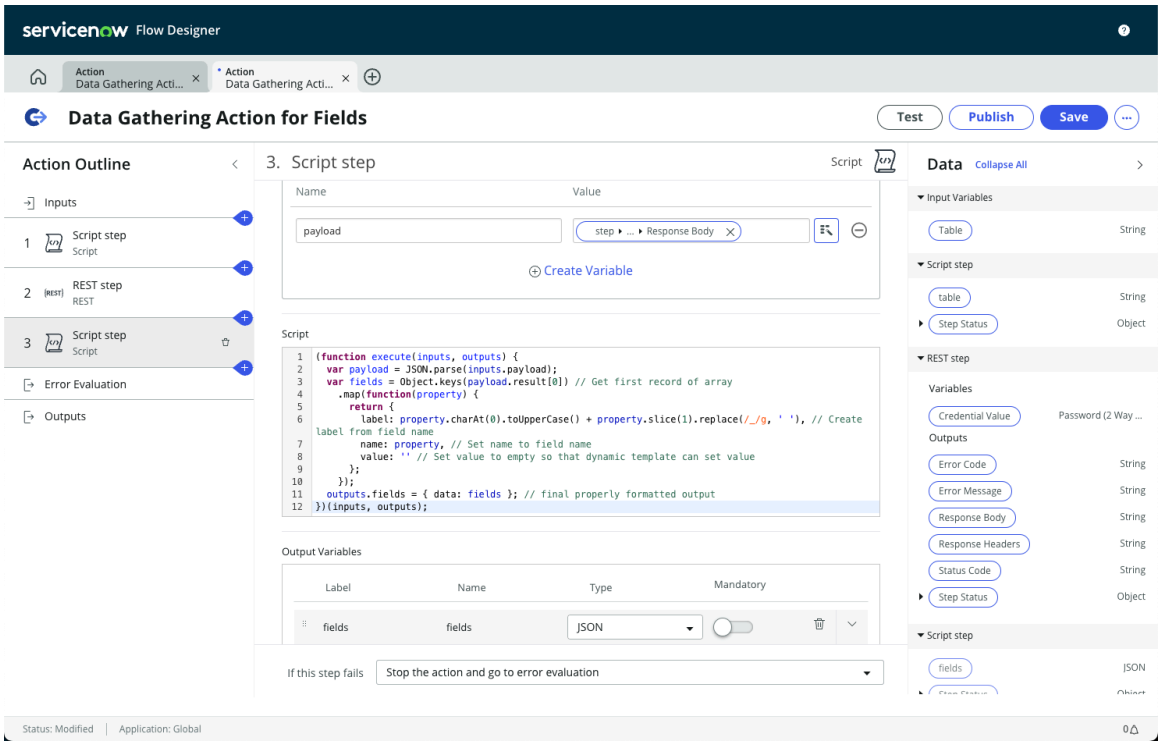
```
      };
    });
  outputs.fields = { data: fields }; // final properly
 formatted output
})(inputs, outputs);
```

The REST API returns a JSON-formatted string that describes each field as a set of name value pairs where the name is the field name and value is the field value. A dynamic template only needs the field name and label. The field value must be blank so that it can be set by the template value. The label property is calculated from the field name by capitalizing the first letter and replacing underscore characters with spaces.

e. In the Output Variables section, select **Create Variable**.

f. In the **Label** and **Name** fields, enter `fields`.

g. In the **Type** field, select **JSON**.



11. In the Action Outline, select **Outputs**.

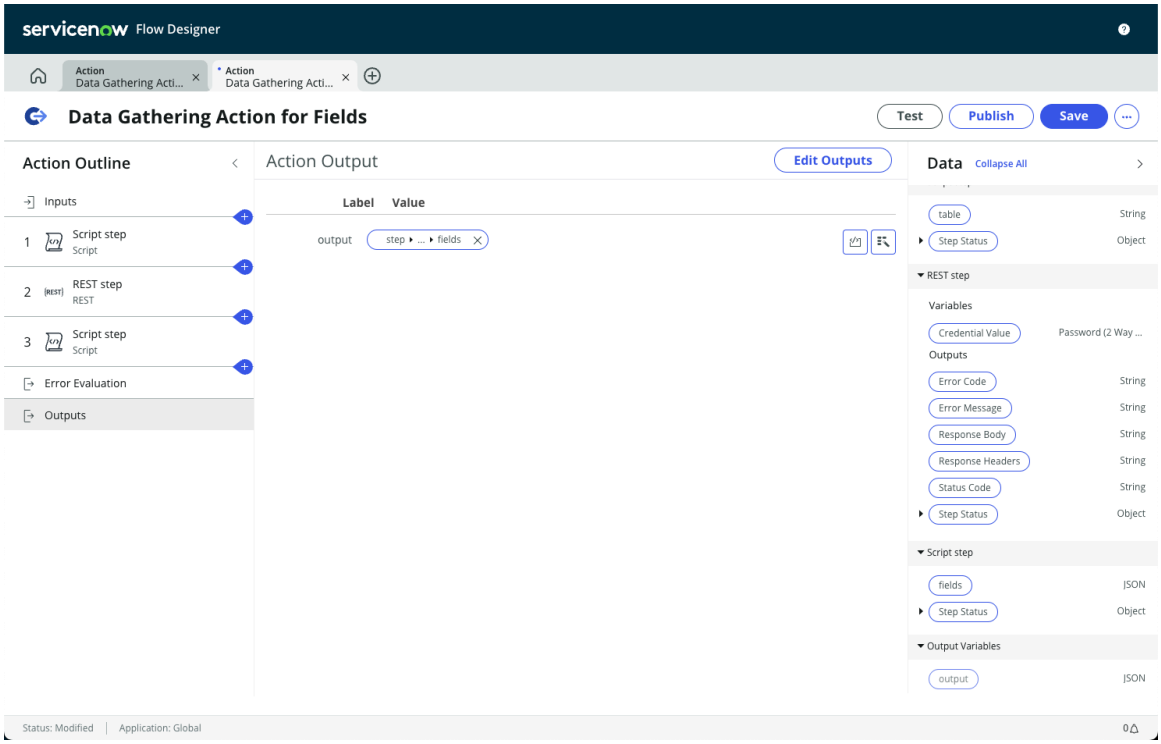
a. On the Action Output header, select **Create Output**.

b. In the **Label** and **Name** fields, enter `output`.

c. In the **Type** field, select **JSON**.

d. Select **Exit Edit Mode**.

e. Next to the **Value** field, select the data pill picker ( ) and then select **Script step** > **fields**.



12. In the Action header, select **Save** and then select **Test** to test the action.

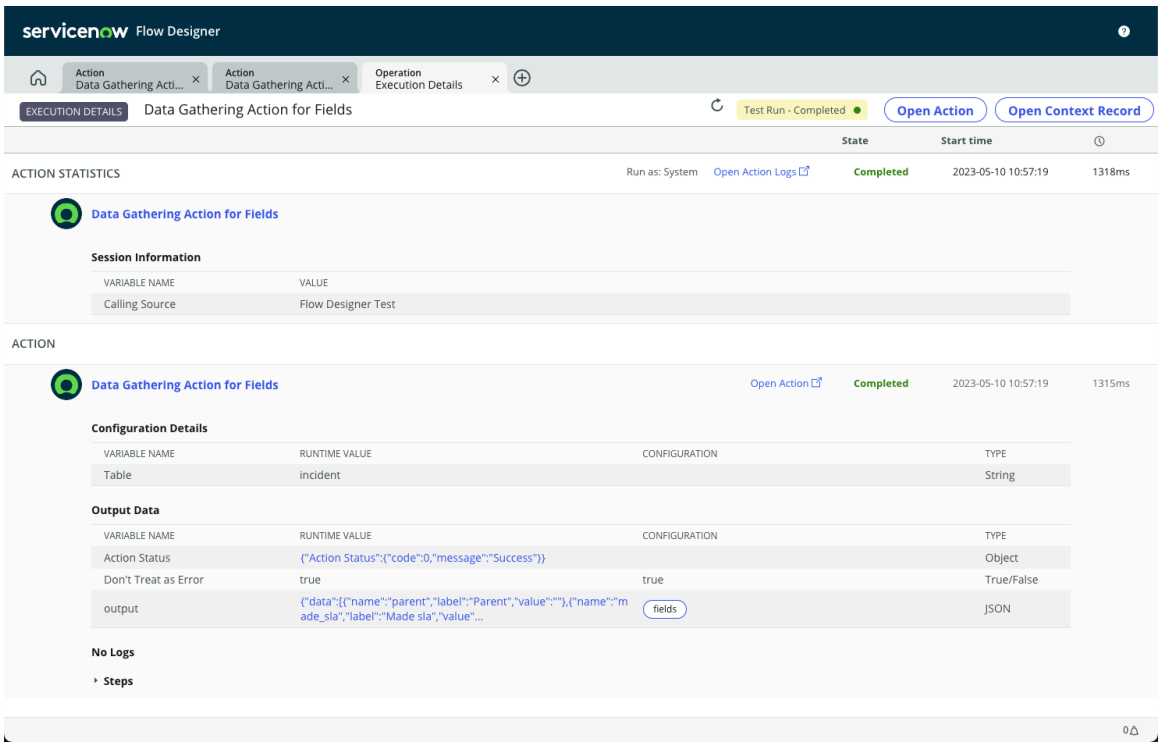
a. On the Test Action screen, in the **Table** field, enter `incident`.

b. Select **Run Test**.

c. View the action's execution details.
Your data gathering action runs successfully if the runtime value for `output` is a complex object containing an array of key-value pairs for `label`, `name`, and `value` as shown in the following abbreviated example.

```
{
    "data": [
      {
        "name": "parent",
        "label": "Parent",
        "value": ""
      },
      {
        "name": "made_sla",
        "label": "Made sla",
        "value": ""
      },
      {
        "name": "caused_by",
        "label": "Caused by",
        "value": ""
      },
      {
        "name": "watch_list",
        "label": "Watch list",
        "value": ""
      },
      {
```

```
        "name": "upon_reject",
        "label": "Upon reject",
        "value": ""
      },
      {
        "name": "sys_updated_on",
        "label": "Sys updated on",
        "value": ""
      },
      {
        "name": "...",
        "label": "...",
        "value": ""
      },
      {
        "name": "category",
        "label": "Category",
        "value": ""
      }
    ]
}
```

13. In the Action header, click **Publish** to make this action available to other actions within the Global scope.

Create a data gathering action to add dynamic inputs

Create a data gathering action to add a reference type dynamic input.

Before you begin

Role required: action_designer or admin

About this task

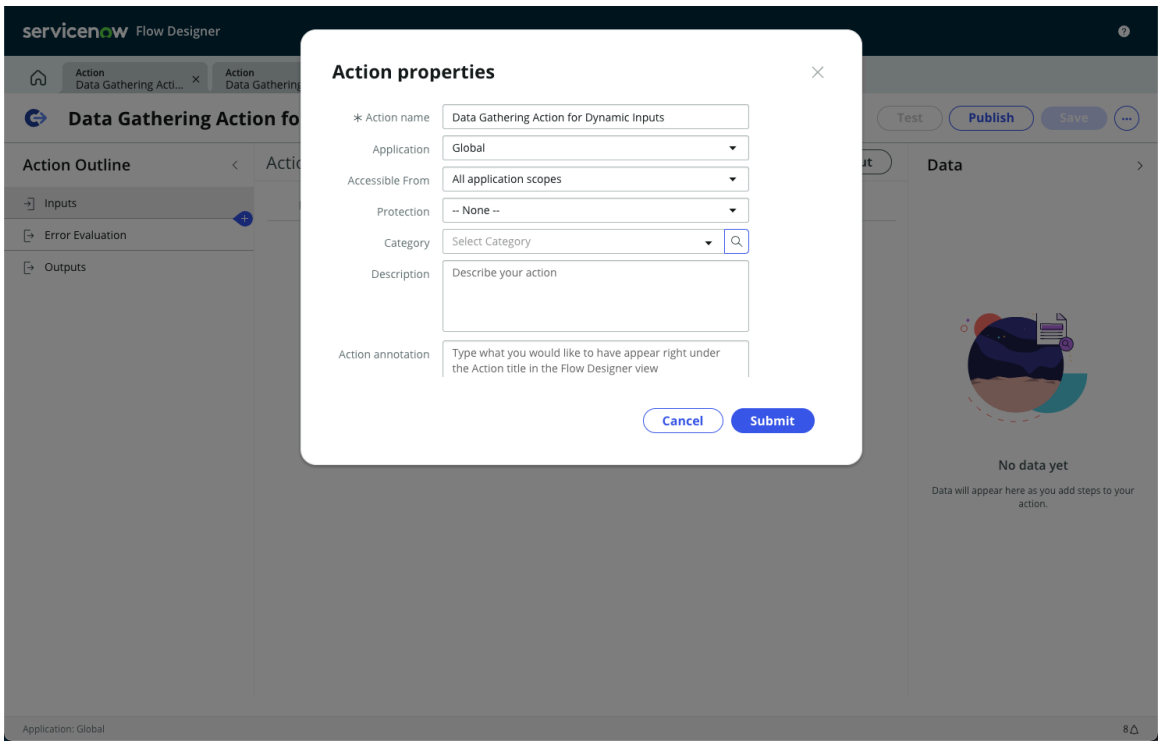
This data gathering action includes these elements.

- An action input variable to store a table name
- A script step to construct two dynamic inputs as a JSON object
- An action output variable named `output` to store the dynamic inputs

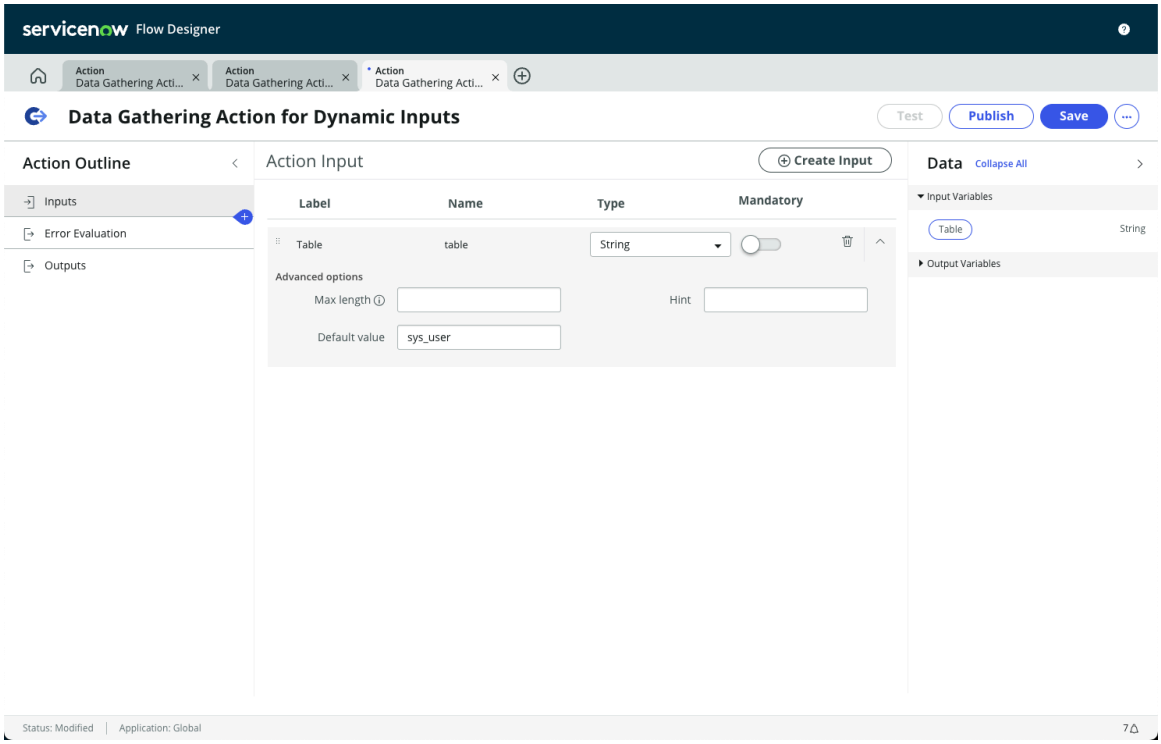
This custom action uses a reference type input to duplicate the functionality of the ServiceNow core Look Up Record action and Look Up Records action. Generally, you don't need to create custom actions to perform basic record operations. This action illustrates working with the Script steps to gather field data for dynamic inputs.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Actions**.
3. Select **Create new** > **Action**.
   a. On the Action properties screen, in the **Action name** field, enter `Data Gathering Action for Dynamic Inputs`.
   b. For **Application**, select Global.
   c. Select **Submit**.



4. From the Action Outline, select **Inputs**.
   a. In the Action Input header, select **Create Input**.
   b. In the **Label** and **Name** fields, enter `Table`.
   c. In the **Type** field, select **String**.
   d. Select the Toggle Advanced Inputs icon ( ).
   e. In the **Default value** field, enter `sys_user`.



5. In the Action Outline under Inputs, select the Add a new step icon ( ).
6. From the **Integrations** section, select the **Script** step, and enter these field values.

   a. In the Input Variables section, select **Create Variable**.

   b. In the **Name** field, enter `table`.

   c. Next to the **Value**, select the data pill picker ( ) and select **Inputs** > **Table**.

   d. In the **Script** field, enter the following code.

```
(function execute(inputs, outputs) {
    // Define JSON for desired dynamic input type
    outputs.data = {
        data: [{
          label: 'Reference type input',
          name: 'referencetype',
          reference: inputs.table,
          type: 'reference',
        }]
      }
  })(inputs, outputs);
```

   Each type of dynamic input has its own JSON structure. A reference field dynamic input needs these properties.

   label

   The text to display next to the input. For example, Reference field input.

   name

   The internal name of the dynamic input used to identify it and to store values. For example, referencetype.

reference

The target table name of the reference field. For example, sys_user. In this script, the table name is a variable.

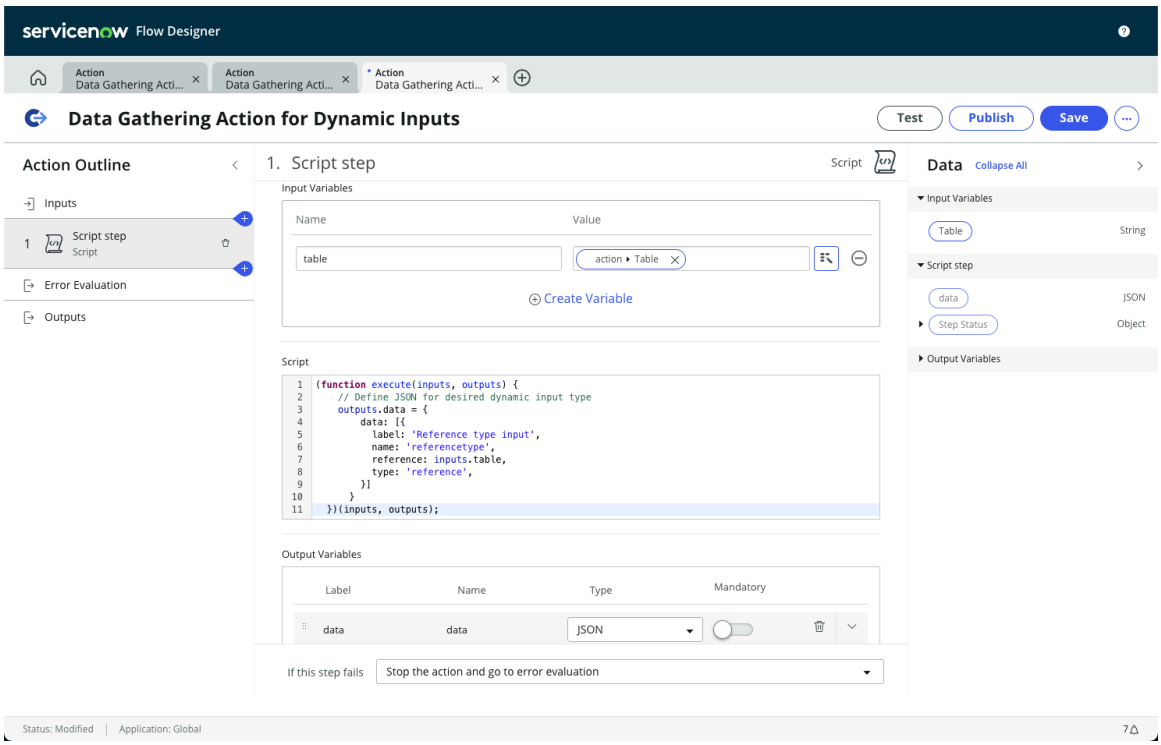
type

The data type of the dynamic input. For example, reference. The type property determines how Workflow Studio displays the input and what other properties are required to configure it.

e. In the Output Variables section, select **Create Variable**.

f. In the **Label** and **Name** fields, enter `data`.

g. In the **Type** field, select **JSON**.



7. In the Action Outline, select **Outputs**.

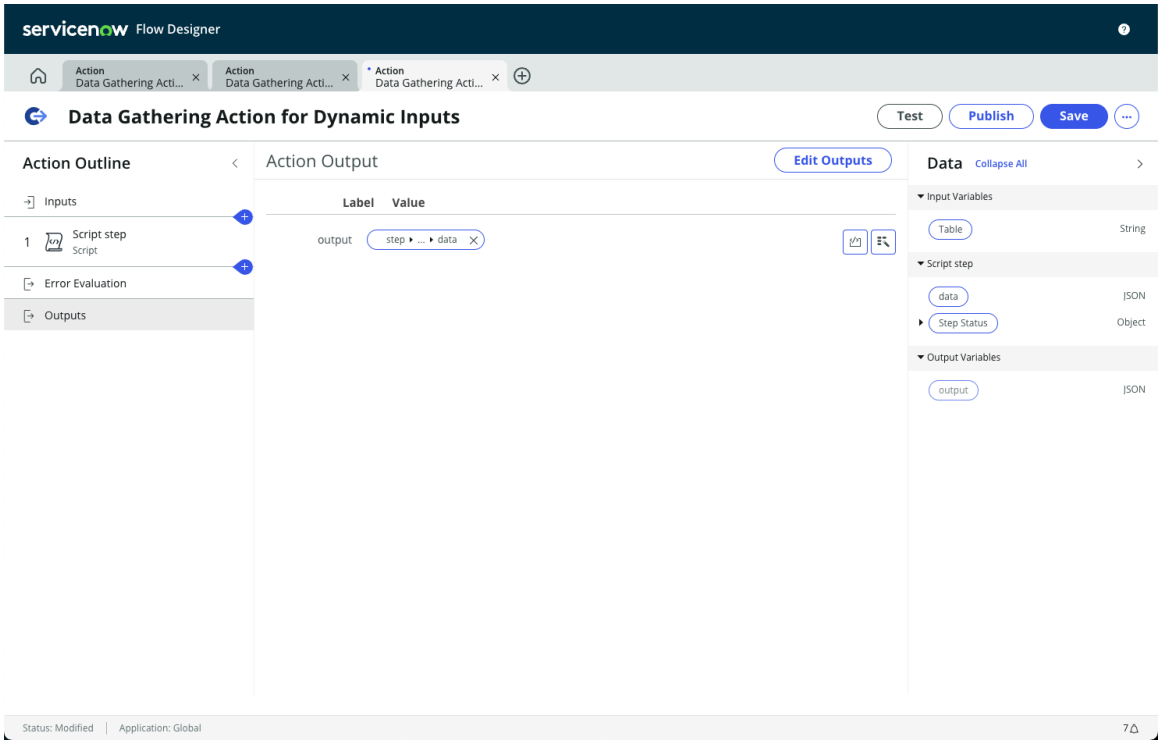
a. On the Action Output header, select **Create Output**.

b. In the **Label** and **Name** fields, enter `output`.

c. In the **Type** field, select **JSON**.

d. Select **Exit Edit Mode**.

e. Next to the **Value** field, select the data pill picker ( ) and then select **Script step** > **data**.

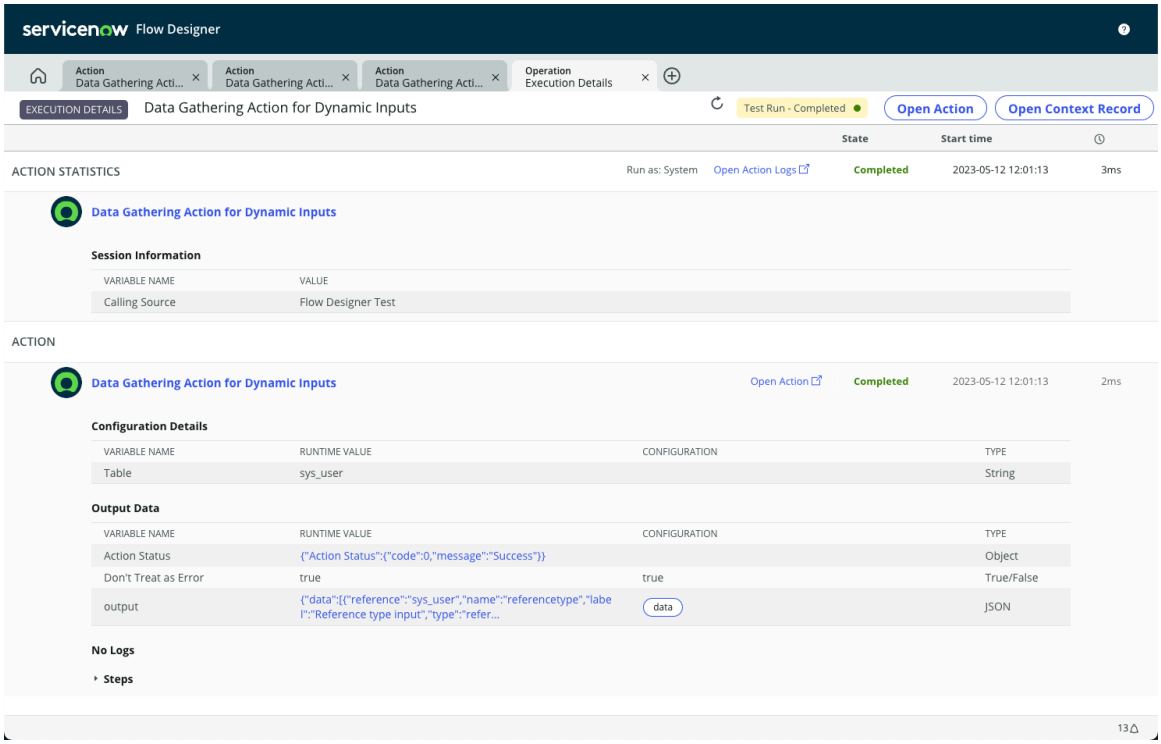


8. In the Action header, select **Save** and then select **Test** to test the action.

a. Select **Run Test**.

b. View the action's execution details.
If your data gathering action runs successfully, then the runtime value for `output` is a JSON string as shown in the following example.

```
{
  "data": [
    {
      "reference": "sys_user",
      "name": "referencetype",
      "label": "Reference type input",
      "type": "reference"
    }
  ]
}
```



9. In the Action header, select **Publish** to make this action available to other actions within the Global scope.

Create a custom action to test dynamic inputs

Create a custom action to test dynamic inputs.

Before you begin

Role required: action_designer or admin

About this task

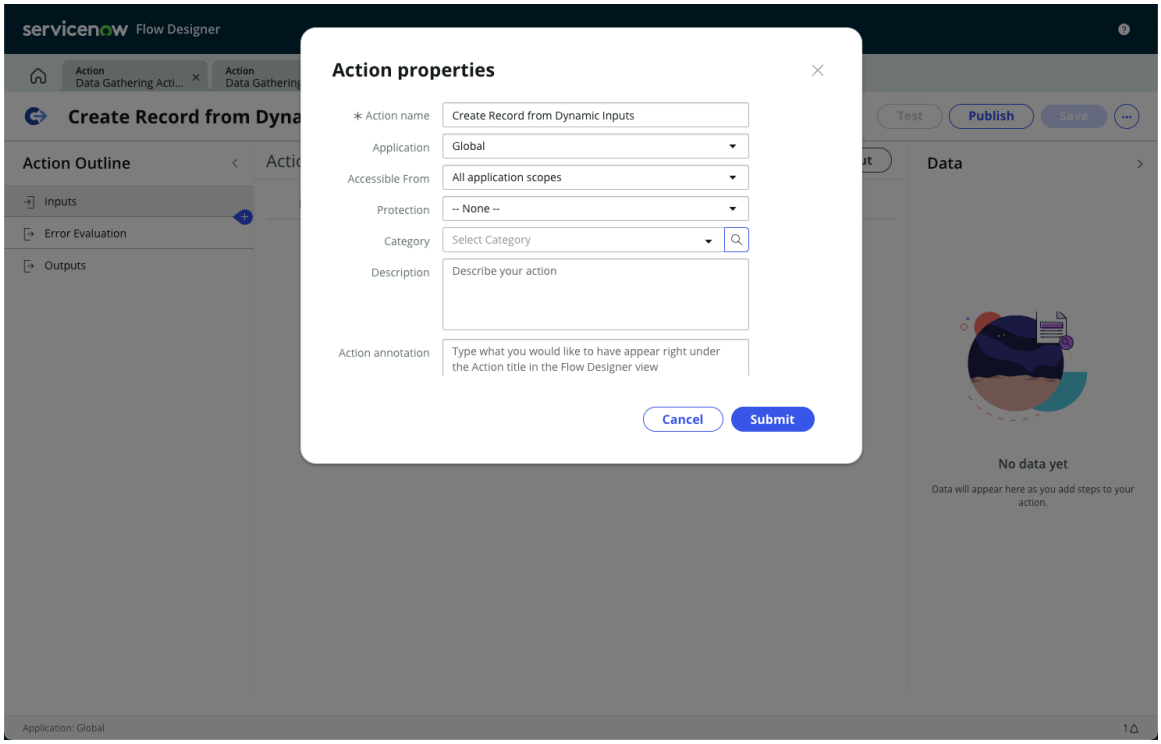
This helper action duplicates the functionality of the Create Record step to illustrate gathering data from a REST step. Generally, you use dynamic inputs for integrations to third-party systems and data.

This custom action includes these elements.

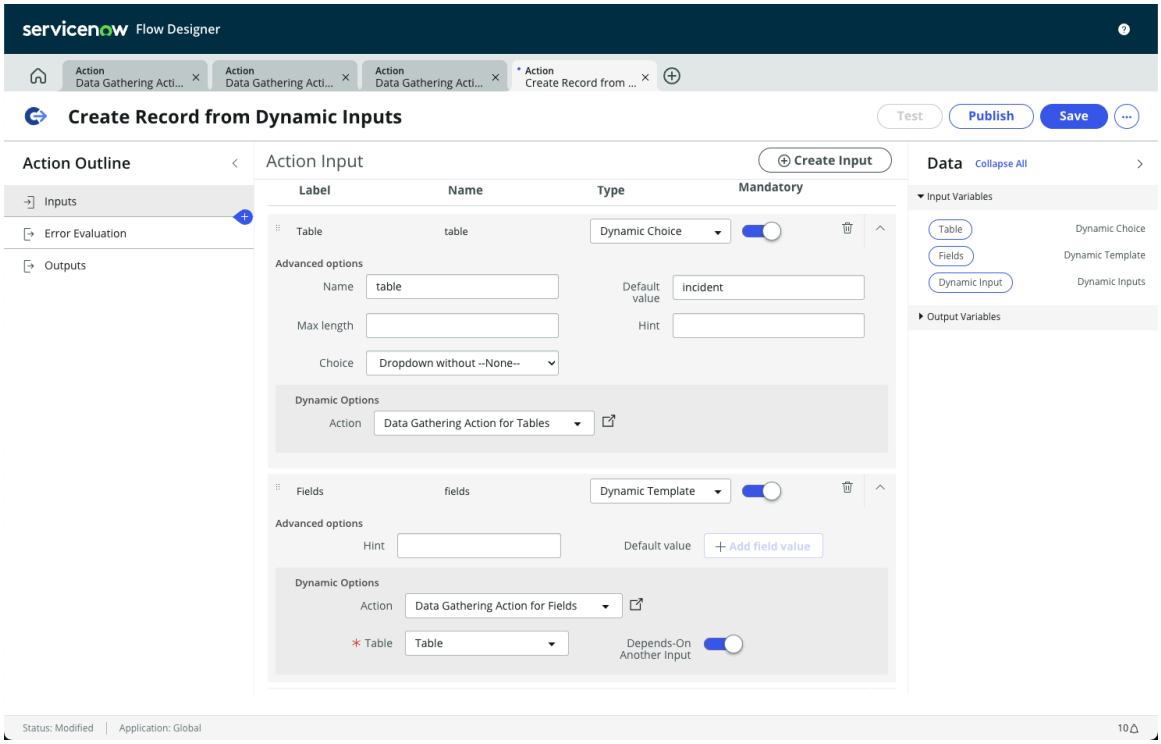
- An input of type Dynamic Choice to select a table
- An input of type Dynamic Template to select and set field values
- A script step to create a record from the GlideRecord - Global
- An action output variable named `output` to store the sys_id of the record created

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Actions**.
3. Select **Create new** > **Action**.
   a. On the Action properties screen, in the **Action name** field, enter `Create Record from Dynamic Inputs`.
   b. For **Application**, select Global.
   c. Select **Submit**.



4. In the Action Outline, select **Inputs**.
   a. Select **Create Input**.
   b. In the **Label** and **Name** fields, enter `Table`.
   c. In the **Type** field, select **Dynamic Choice**.
   d. Turn on the **Mandatory** toggle switch.
   e. In **Default value**, enter `incident`.
   f. Accept the default value for **Choice**.
   g. Under Dynamic Options for the **Action** field, select **Data Gathering Action for Tables**.
   h. Select **Create Input** to create another action input.
   i. In the **Label** and **Name** fields, enter `Fields`.
   j. In the **Type** field, select **Dynamic Template**.
   k. Turn on the **Mandatory** toggle switch.
   l. Under Dynamic Options in the **Action** field, select **Data Gathering Action for Fields**.
   m. Next to the **Table** field, turn on the **Depends-On Another Input** toggle switch.
   n. In the **Table** field, select the **Table** input.
   o. Select **Create Input** to create another action input.
   p. In the **Label** and **Name** fields, enter `Dynamic Input`.
   q. In the **Type** field, select **Dynamic Inputs**.
   r. Under Dynamic Options in the **Action** field, select **Data Gathering Action for Dynamic Inputs**.
   s. Leave the **Table** field blank to use the default value of sys_user provided by the data gathering action.



5. In the Action Outline under the Inputs section, select the Add a new step icon ( ).
6. From the **Integrations** section, select the **Script** step, and enter these field values.

   a. In the Input Variables section, select **Create Variable**.

   b. In the **Name** field, enter `table`.

   c. Next to the **Value** field, select the data pill picker ( ) and select **Inputs** > **Table**.

   d. Select **Create Variable** to create another input variable.

   e. In the **Name** field, enter `fields`.

   f. Next to the **Value** field, select the data pill picker ( ) and select **Inputs** > **Fields**.

   g. Select **Create Variable** to create another input variable.

   h. In the **Name** field, enter `payload`.

   i. Next to the **Value** field, select the data pill picker ( ) and select **Inputs** > **Dynamic Input**.

   j. In the **Script** field, enter the following code.

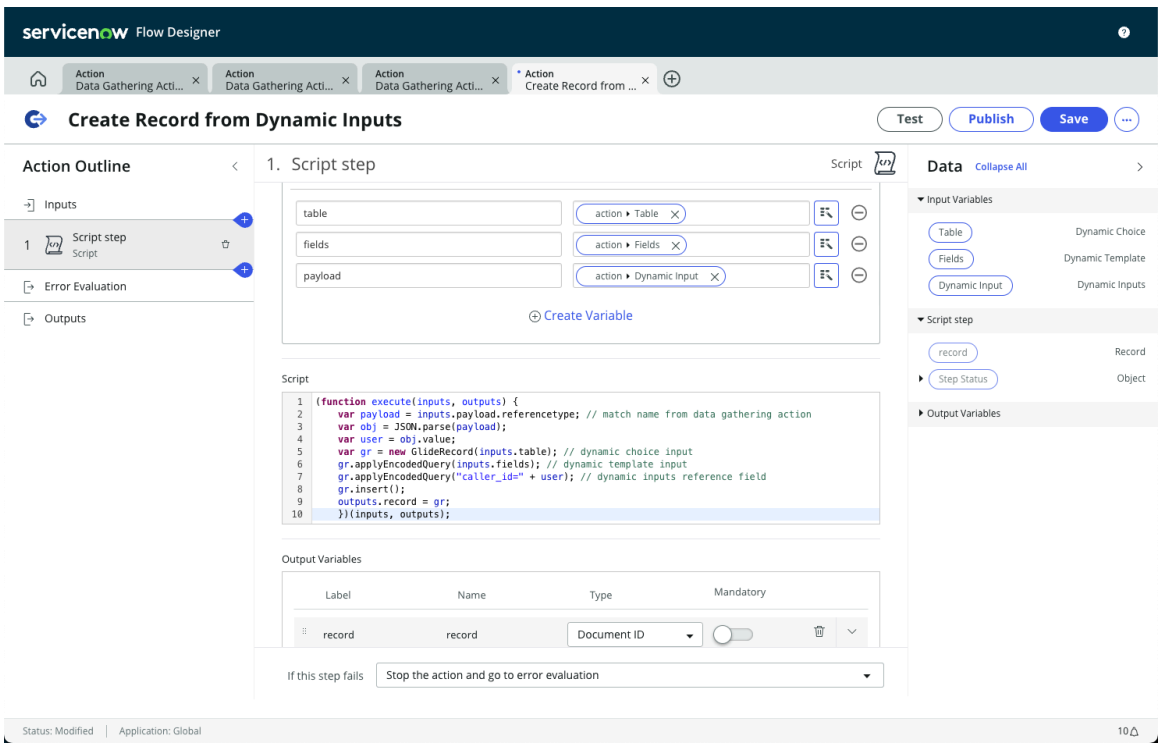
```
(function execute(inputs, outputs) {
    var payload = inputs.payload.referencetype; // match name
 from data gathering action
    var obj = JSON.parse(payload);
    var user = obj.value;
```

```
   var gr = new GlideRecord(inputs.table); // dynamic choice
input
   gr.applyEncodedQuery(inputs.fields); // dynamic template
input
   gr.applyEncodedQuery("caller_id=" + user); // dynamic
inputs reference field
   gr.insert();
   outputs.record = gr;
   })(inputs, outputs);
```

k. In the Output Variables section, select **Create Variable**.

l. In the **Label** and **Name** fields, enter `record`.

m. In the **Type** field, select **Document ID**.



7. In the Action Outline, select **Outputs**.

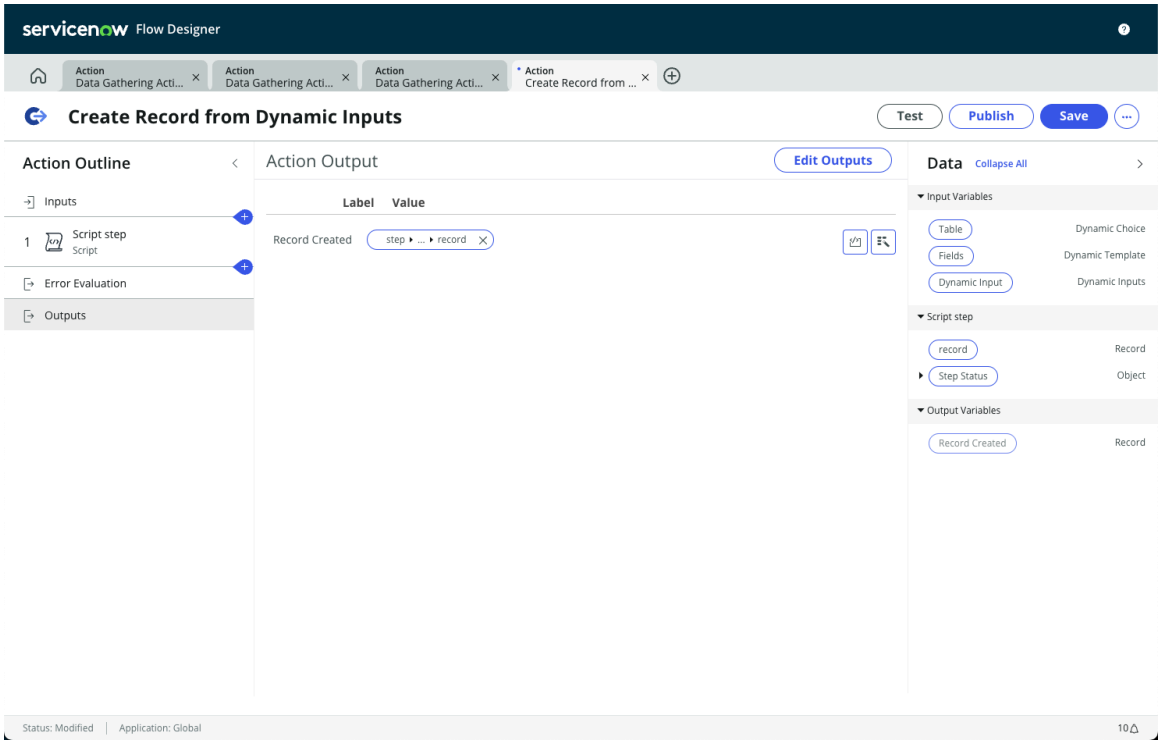
a. On the Action Output header, select **Create Output**.

b. In the **Label** and **Name** fields, enter `Record Created`.

c. In the **Type** field, select **Document ID**.

d. Select **Exit Edit Mode**.

e. Next to the **Value** field, select the data pill picker ( ) and then select **Script step** > **record**.



8. In the Action header, select **Save** and then select **Test** to test the action.

   a. On the Test Action screen, in the **Table** input, select any dynamically generated table choice option.

      Example
      For example, select the Incident table.

   b. Select **Add Field Value**, select any field, and enter any value.

      Example
      For example, select the Short description field and enter the value test.

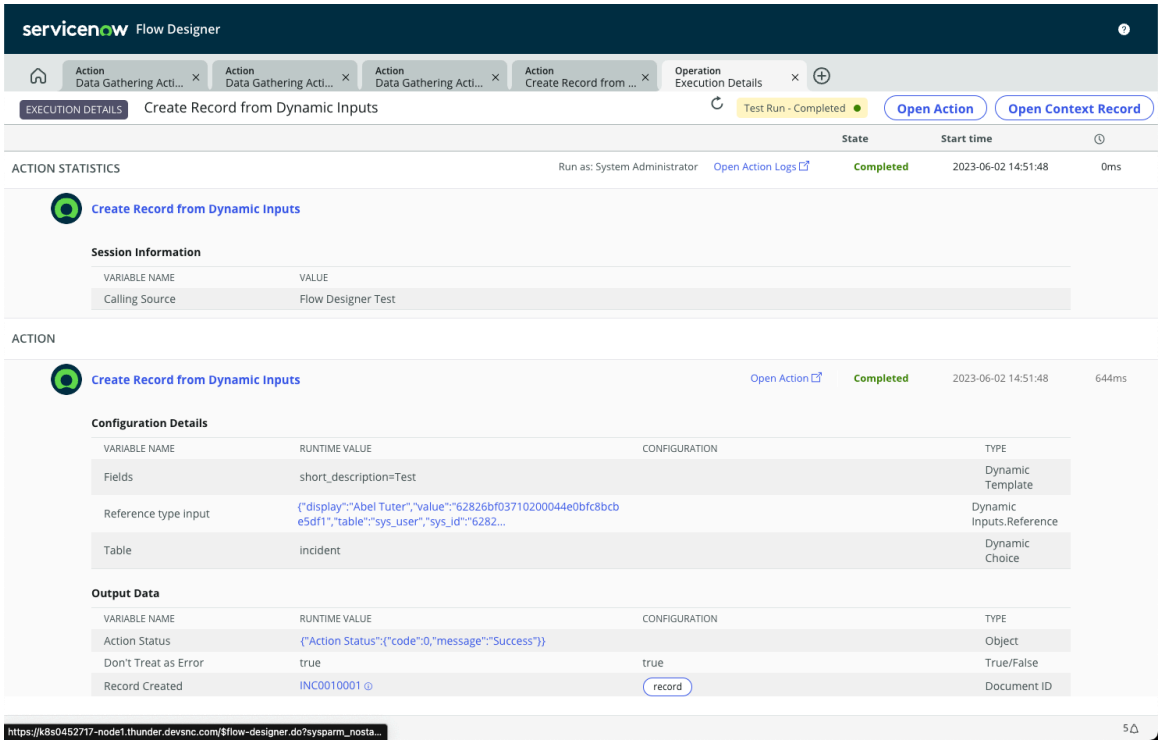
   c. In the Dynamic inputs section and the **Reference type input** field, select a reference field value.

      Example
      For example, select the user Abel Tuter.

   d. Select **Run Test**.

      Note:
      Records you create from the Table API bypass data policies and required fields normally related with record creation. This action is for illustration purposes only. Use the Create Record step instead to create records with standard protections and validations.

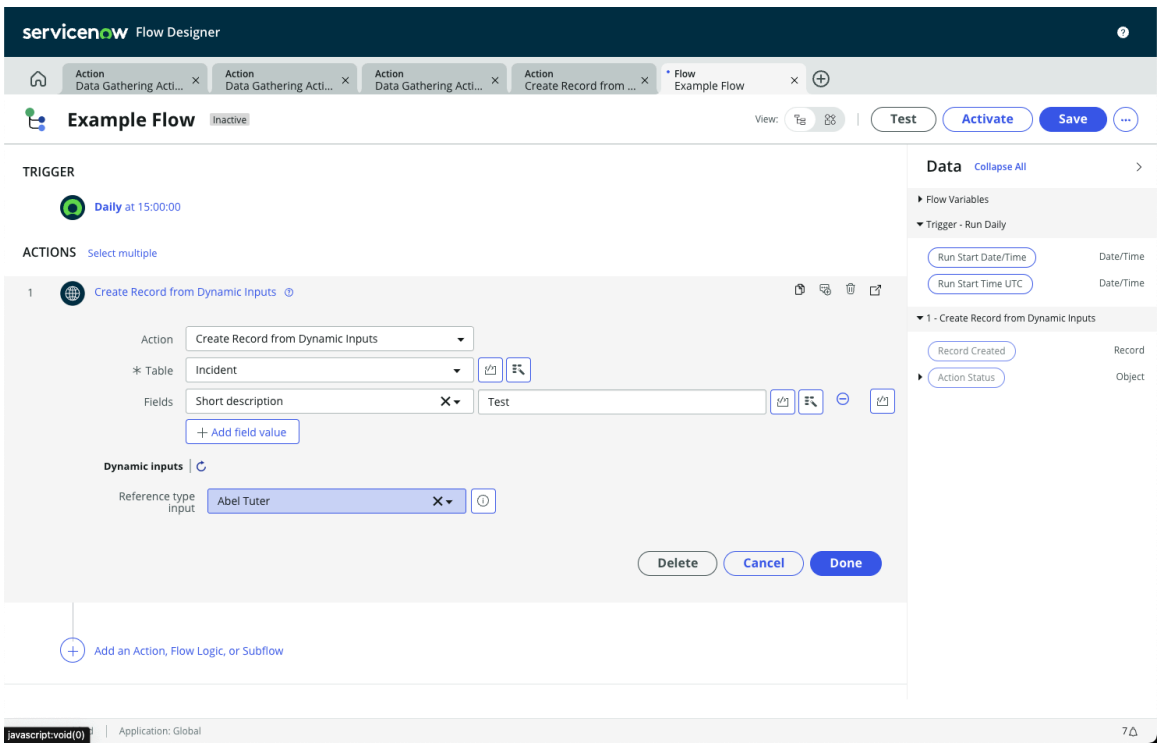
   e. View the action's execution details.
      You can use the Record Created output to verify that the runtime values for `Table` and `Field` match the test values you entered.



9. In the Action header, select **Publish** to make this action available to flows within the Global scope.

Result

You can add this action to a flow in Workflow Studio. This sample action dynamically generates a list of tables and related fields in your instance whose values that you can assign during flow design.



Create a data gathering action for a dynamic choice

Create a data gathering action to generate a list of choice options for a dynamic choice input.

Before you begin

- Role required: action_designer or admin
- Create credentials and connections for your REST endpoint

Note:
Dynamic inputs are not available in the base system. To use dynamic inputs in Workflow Studio, you must Request an Integration Hub plugin.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select **Create New** > **Action**.
3. In the **Name** field, enter a name for your action, choose the proper **Application** scope, and then select **Submit**.
4. In the Action Outline, select the add a new step icon ( ) under **Inputs** and select the **REST** step.
   Configure your REST step to get data from the proper **Base URL** and **Resource Path** with any applicable **Query Parameters** for the **HTTP Method** GET. For more information on using the REST step in Integration Hub, see REST step and REST in IntegrationHub.
5. In the Action Outline under the REST step, select the add a new step icon ( ) and select the **Script** step.

   a. From Input Variables, select **Create Variable**.
   This input variable stores the REST response body.

   Example
   For example for the input variable Name, enter `payload`.

   b. For the input variable Value, select the data pill for the REST step's **Response Body** output.

   c. In Script, enter a script to create a JSON output from the input variable.
   Your script needs to do these tasks.

   Parse the input variable as JSON

   You can use the JSON - Global API to transform the input variable from a string to a JSON object.

   Create a new JSON object formatted for a dynamic choice

   You must know the data structure of the REST Response to map response values to choice list options. Your script must create a dynamic choice JSON object that has a property named `data`. The value of the data property must be an array of choice list options. Each choice list option must have a `label` and a `name` property set to string values. The label property determines how the option appears in the choice list. The name property determines how the option is stored and referred to in the system.

   For example, this JSON object defines an array with three choice list options.

```
{
    data: [
        {
            label: "Choice Option 1",
            name: "choice_option_1"
```

```
        },
        {
            label: "Choice Option 2",
            name: "choice_option_2"
        },
        {
            label: "Choice Option 3",
            name: "choice_option_3"
        }
    ]
}
```

Note:
A dynamic choice input can only display up to 5000 choice list options. A JSON object that returns more than 5000 choice list options will be truncated when it is rendered.

Set the outputs object

Set the outputs object to return your dynamic choice JSON object.

d. From Output Variables, select **Create Variable**.
This output variable stores the choice list options your script creates.

e. For the output variable type, select **JSON**.
The script output variable type must be JSON.

6. In the Action Outline, select **Outputs**

a. Select Create Output.

b. Set the output label and name `output`.

c. Set the output type to **JSON**.

Note:
The action can have multiple outputs, but there can only be one of type JSON.

d. Select **Exit Edit Mode**.

e. Next to the **Value** field, select the data pill picker ( ) and then select Script step output variable you created to store choice list options.

7. Select **Save** and test the action.
The runtime value for `output` must be a JSON object that has a `data` property and an array of choice list options.

Example

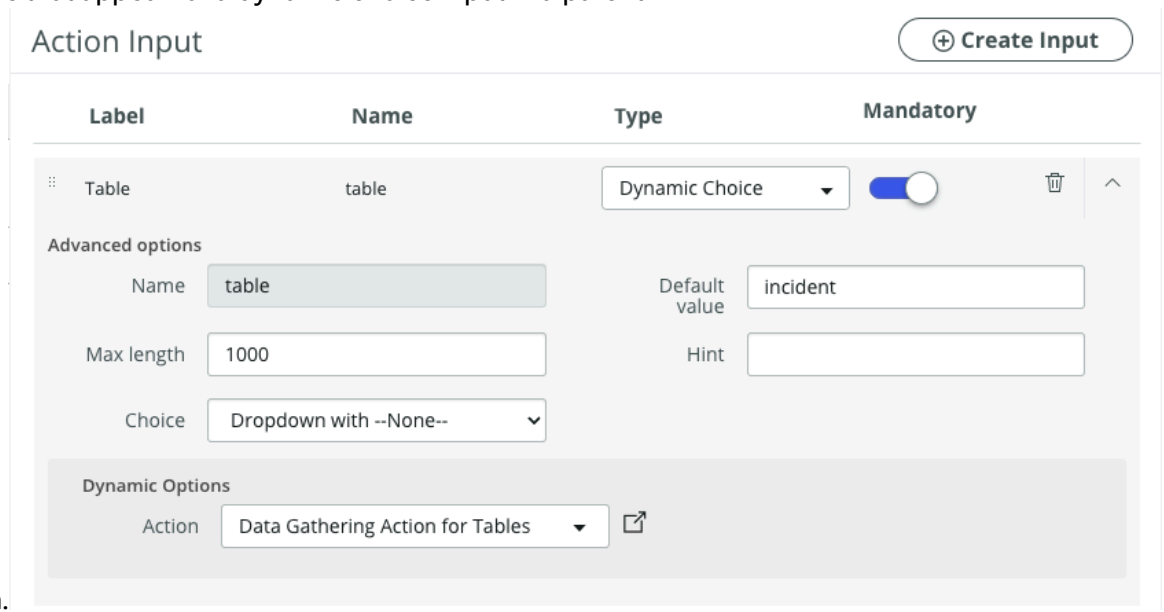
Output Data

VARIABLE NAME	RUNTIME VALUE	CONFIGURATION	TYPE
Action Status	{"Action Status":{"code":0,"message":"Success"}}		Object
Don't Treat as Error	true	true	True/False
output	{"data":[{"name":"evaluation","label":"A/B Testing Evaluation"}, {"name":"sla","label":"Agreement"},{...	tables	JSON

8. Select **Publish** to make the action available to other flows and actions within the same application scope.

Result

You can now use your data gathering action to populate the options that appear for a dynamic choice input in a parent action.



Create a data gathering action for a dynamic template

Create a data gathering action to collect record field values for a dynamic template input.

Before you begin

- Role required: action_designer or admin
- Create credentials and connections for your REST endpoint

> **Note:**
> Dynamic inputs are not available in the base system. To use dynamic inputs in Workflow Studio, you must Request an Integration Hub plugin.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select **Create new** > **Action**.
3. In the **Action name** field, enter a name for your action, choose the proper **Application** scope, and then select **Submit**.
4. **Optional:** From the Action Outline, select **Inputs**.
   You may need to create an input to provide a table name or a dynamic URL path to your REST step.

Example
For example, create a String input to store a table name. See Create a data gathering action to get field names for instructions.

5. In the Action Outline under Inputs, select the add a new step icon ( ) and select the **REST** step.

   a. From the Connection Details, select a connection alias or define a connection inline.
   A connection alias allows you to update connection details without having to edit the action.

   Example
   For example, select an connection alias to your local instance.

   b. For the Request Details, select a method to build a request, provide a resource path, select an HTTP method, and provide any query parameters.
   Data gathering actions generally use a GET HTTP method to request data from a REST endpoint. For more information on using the REST step in Integration Hub, see REST step and REST in IntegrationHub .

6. In the Action Outline under the REST step, select the add a new step icon ( ) and select the **Script** step.

   a. From Input Variables, select **Create Variable**, and create an input variable to store the REST response body.

   b. For the input variable Value, select the data pill for the REST step's **Response Body** output.

   c. In Script, enter a script to create a JSON output from the input variable.
   Your script needs to do these tasks.

   Parse the input variable as JSON

   You can use the JSON - Global API to transform the input variable from a string to a JSON object.

   Create a new JSON object formatted for a dynamic template

   You must know the data structure of the REST Response to map response values to template values. Your script must create a dynamic template JSON object that has a property named `data`. The value of the data property must be an array of template values. Each template value must have a `label` property, a `name` property, and an empty `value` property. The label property determines how the template value option appears in the action. The name property determines how the template value option is stored and referred to in the system. The value property is empty so that the value can be set dynamically when the action is configured.

   For example, this JSON object defines an array of template values from the incident table.

```
{
    "data": [
      {
        "name": "parent",
        "label": "Parent",
        "value": ""
      },
```

```
      {
        "name": "number",
        "label": "Number",
        "value": ""
      },
      {
        "name": "state",
        "label": "State",
        "value": ""
      },
      {
        "name": "active",
        "label": "Active",
        "value": ""
      },
      {
        "name": "priority",
        "label": "Priority",
        "value": ""
      },
      {
        "name": "caller_id",
        "label": "Caller id",
        "value": ""
      },
      {
        "name": "short_description",
        "label": "Short description",
        "value": ""
      },
      {
        "name": "description",
        "label": "Description",
        "value": ""
      },
      {
        "name": "sys_id",
        "label": "Sys id",
        "value": ""
      },
      {
        "name": "urgency",
        "label": "Urgency",
        "value": ""
      },
      {
        "name": "assigned_to",
        "label": "Assigned to",
        "value": ""
      },
      {
        "name": "severity",
        "label": "Severity",
        "value": ""
      },
      {
        "name": "category",
```

```
        "label": "Category",
        "value": ""
      }
    ]
}
```

Note:
A dynamic template input can only display up to 5000 field template values. A JSON object that returns more than 5000 field template values will be truncated when it is rendered.

Set the outputs object

Set the outputs object to return your dynamic template JSON object.

d. From Output Variables, select **Create Variable**.
This output variable stores the template values your script creates.

e. For the output variable type, select **JSON**.
The script output variable type must be JSON.

7. In the Action Outline, select **Outputs**

a. Select Create Output.

b. Set the output label and name `output`.

c. Set the output type to **JSON**.

Note:
The action can have multiple outputs, but there can only be one of type JSON.

d. Select **Exit Edit Mode**.

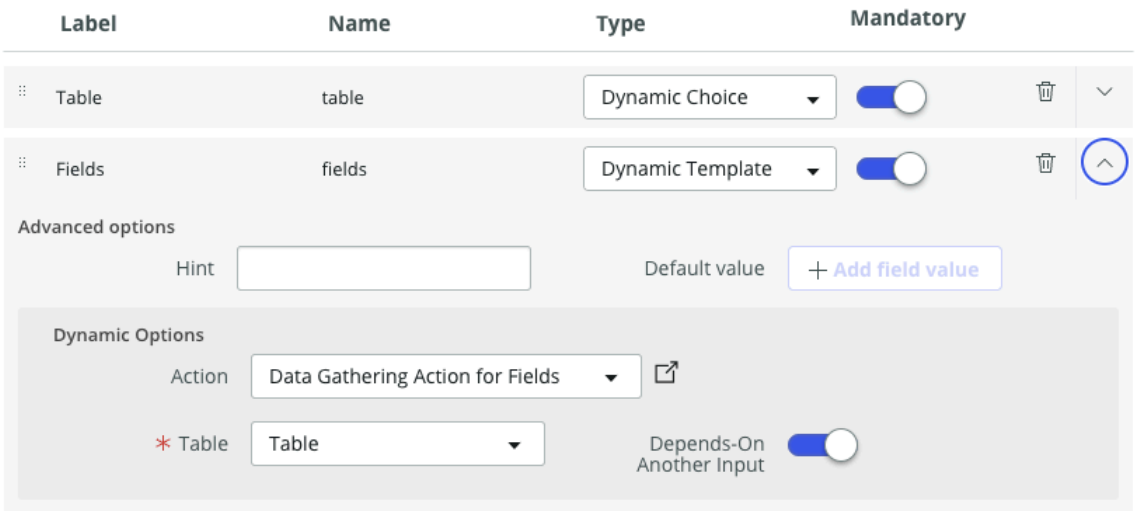
e. Next to the **Value** field, select the data pill picker ( ) and then select Script step output variable you created to store template values.

8. Click **Save** and test the action.
In the execution details, your data gathering action runs successfully if the runtime value for `output` contains the `data` property in the proper format.

9. Click **Publish** to make the action available to other flows or actions within the same application scope.

Result

You can now use your data gathering action to populate the options that appear for a dynamic template input in a parent action.



Create a data gathering action for a dynamic inputs type input

Create a data gathering action to create arbitrary action inputs using a dynamic inputs type input.

Before you begin

Role required: admin

Note:
Dynamic inputs are not available in the base system. To use dynamic inputs in Workflow Studio, you must Request an Integration Hub plugin.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select **Create new** > **Action**.
3. In the **Action name** field, enter a name for your action, choose the proper **Application** scope, and then select **Submit**.
4. **Optional:** From the Action Outline, select **Inputs**.
   You may need to create one or more inputs to provide data for your dynamic inputs.

   Example
   For example, create a String input to store a table name. See Create a data gathering action to add dynamic inputs for instructions.
5. In the Action Outline under the Inputs section, select the add a new step icon ( ) and select the **Script** step.

   a. **Optional:** From Input Variables, select **Create Variable**, and create an input variable to store any action input values.

      Example
      For example create an input variable to store a table name, and map it to the matching action input.

   b. In Script, enter a script to create a JSON output from the input variable.
      Your script needs to do these tasks.

Create a new JSON object formatted for an action input

You must know the data structure of an action input to create a dynamic input. Your script must create a JSON object that has a property named `data`. The value of the data property must be an array of dynamic inputs. Each dynamic input must has its own required and optional properties.

This example script sets an outputs variable named data. There is a property for each type of dynamic input available.

```
(function execute(inputs, outputs) {
  outputs.data = {
    data: [{
      label: 'Choice type input',
      name: 'choicetype',
      defaultValue: 'choice_1',
      type: "choice",
      choices: [
          { label: "Choice 1", value: "choice_1" },
          { label: "Choice 2", value: "choice_2" }
      ]
    }, {
      label: 'Datetime type input',
      name: 'datetimetype',
      type: 'datetime',
    }, {
      label: 'Decimal type input',
      name: 'decimaltype',
      type: 'decimal',
    }, {
      label: 'Email type input',
      name: 'emailtype',
      type: 'email',
    }, {
      label: 'HTML type input',
      name: 'htmltype',
      type: 'html',
    }, {
      label: 'Integer type input',
      name: 'integertype',
      type: 'integer',
    }, {
      label: 'Password2 type input',
      name: 'password2type',
      type: 'password2',
    }, {
      label: 'Reference type input',
      name: 'referencetype',
      reference: 'sys_user',
      type: 'reference',
    }, {
      label: 'String type input',
      name: 'stringtype',
      defaultValue: 'abcdef',
      type: 'string',
      mandatory: true
    }]
```

```
  }
})(inputs, outputs);
```

Note:
A dynamic inputs type input can only support 40 input values before it risks running out of memory and producing unexpected behavior such as rendering errors and data truncation.

Set the outputs object

Set the outputs object to return the dynamic inputs as a JSON object. See Create a data gathering action to add dynamic inputs for an example of creating an output for a Reference field type action input.

c. From Output Variables, select **Create Variable**.
This output variable stores the dynamic inputs your script creates.

d. For the output variable type, select **JSON**.
The script output variable type must be JSON.

6. In the Action Outline, select **Outputs**

a. Select Create Output.

b. Set the output label and name `output`.

c. Set the output type to **JSON**.

Note:
The action can have multiple outputs, but there can only be one of type JSON.

d. Select **Exit Edit Mode**.

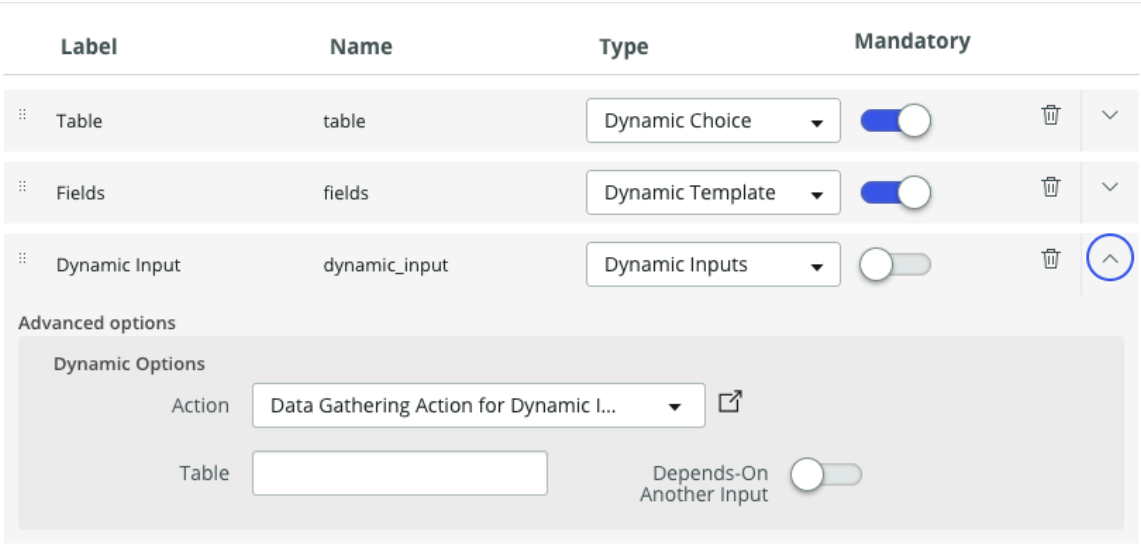
e. Next to the **Value** field, select the data pill picker ( ) and then select Script step output variable you created to store dynamic inputs.

7. Click **Save** and test the action.
In the execution details, your data gathering action runs successfully if the runtime value for `output` contains the `data` property in the proper format.

8. Click **Publish** to make the action available to other flows or actions within the same application scope.

Result

You can now use your data gathering action to add arbitrary action inputs to a parent action.



Dynamic input configuration options

Use these options to configure dynamic inputs.

Common input options

Option	Description
Label	Enter a label that appears for the action input when the action is added to a flow.
Name	Enter a name used to refer to the input in system records and in scripts.
Type	Select one of the dynamic data types. • Dynamic Choice • Dynamic Inputs • Dynamic Template

Dynamic Choice advanced options

Option	Description
Name	Enter a name used to refer to the input in system records and in scripts.
Default Value	Enter a string value to use as a default for choice list.
Max length	Enter the maximum character length for one choice or template field value.
Hint	Enter a hint to display for the choice list.
Choice	Select one of the following choice list options: • **Dropdown with --None--** • **Dropdown without --None--**

Option	Description
Action	Select the data gathering action to generate choice list values. For example, an action that converts records into a choice list options.

Dynamic Inputs advanced options

Option	Description
Action	Select the data gathering action to generate additional action inputs. For example, an action that converts Service Catalog variables into additional action inputs.

Dynamic Template advanced options

Option	Description
Hint	Enter a hint to display for the record whose fields you want to dynamically display.
Default Value	Select a field and field value to use as a default for the action.
Action	Select the data gathering action to generate record template values. For example, an action that displays the fields of a third-party record or service.

Dynamic outputs

Access action and subflow outputs as dynamically generated data pills during flow design. You can also build data gathering actions to generate complex objects from ServiceNow AI Platform and Integration Hub outputs.

Note:
Dynamic outputs are not available in the base system. To use dynamic outputs in Workflow Studio, you must Request an Integration Hub plugin. Dynamic outputs are part of the ServiceNow Flow Designer - Introspection [com.glide.hub.flow_designer_introspection] plugin.

During the flow design, a dynamic output retrieves the complex object's schema values and displays them as data pills in the data panel.

A dynamic output must point to a data gathering action that collects the displayed data. For example, a data gathering action can retrieve values from a third-party system as part of an Integration Hub spoke. To use a dynamic output in Workflow Studio:

1. An action designer creates a data gathering action.
2. An action designer creates a parent action with a dynamic output that points to the child data gathering action.
3. A flow designer adds the parent action to a flow.

Data gathering actions

A data gathering action collects data to be used by other actions. Data gathering actions are intended to be called from dynamic inputs rather than be added directly to a flow. Data gathering actions typically collect data from third-party systems using a REST call. All data gathering actions must meet these requirements and constraints.

- The action has a script step that contains an output variable of type JSON.
- The action has an output named *output* of type JSON whose value is derived from the script step's JSON output variable.

> **Note:**
> The action can have multiple outputs but can only have one of type JSON.

- The script step formats the JSON output to have a property named `data`.
- The JSON output should not return more than 5000 choice options, field template values, or array element items when the data is intended for a dynamic choice or a dynamic template input.

> **Note:**
> Dynamic choice and dynamic template inputs can only display up to 5000 choice options or 5000 template values from the JSON output.

- The action waits for up to 300 seconds (5 minutes) to gather data before it times out.

> **Note:**
> To change the timeout period for all actions, modify the value of the *sn_flow_designer.sync_action_execution_timeout_in_seconds* system property.

Dynamic object

A dynamic object is either a complex object or an array of complex objects. Action designers build the object structure dynamically with a data gathering action rather than hard-coding it. For more information on building a dynamic object, see Create a data gathering action for a dynamic object.

General guidelines

Use dynamic outputs for third-party integrations

Use dynamic outputs to introspect and fetch data from external systems during the flow design. For example, you can specify service endpoints or call actions that interact with specific endpoint APIs. For more information on setting up third-party integrations with Workflow Studio, see IntegrationHub.

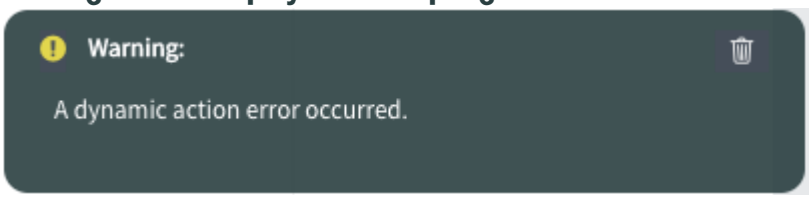
Note the time that is required to retrieve large amounts of data

By default, dynamic outputs have up to 300 seconds to gather data before the system stops them. If your data gathering action needs more time to gather data, set the *sn_flow_designer.sync_action_execution_timeout_in_seconds* system property to a greater value. Avoid long timeout values for interactive flows where an end user is expecting to enter or select a value.

Be aware of scripting errors

Because all data gathering actions use a script step, potential errors could occur from scripting. Review any scripts that are used to output JSON variables because script errors may prevent the outputs from receiving the JSON values that they need. When a dynamic output scripting error occurs, the following warning message may appear.

Message that is displayed for scripting error



Get started with dynamic outputs

Create a sample action that builds dynamic outputs for use in a flow.

Before you begin

Role required: action_designer or admin

Procedure

1. Create connection and credential records for dynamic outputs
   This connection & credential alias will provide the base URL and user account needed to configure the REST steps of your data gathering actions.
2. Create a data gathering action to get a record schema
   This data gathering action will convert a single record into a JSON object for a record dynamic output.
3. Create a data gathering action to get an array of records schema
   This data gathering action will convert a list of records record into JSON array of objects for a records dynamic output.
4. Create a custom action to test dynamic outputs
   This custom action illustrates two types of dynamic output data. One dynamic output generates an object for a single record. Another dynamic output generates an array of objects for a list of records.

Create connection and credential records for dynamic outputs

Create the aliases, connections, and credentials that REST steps will use to connect to your local instance.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **Connections & Credentials** > **Credentials**.
2. Select **New**, select **Basic Auth Credentials**, and enter these field values.
   a. For **Name**, enter `Local Admin`.
   b. For **User name**, enter a user account with access to Flow Designer and the REST API.
      Example
      For example, enter `admin`.
   c. For **Password**, enter the account password.
3. Select **Submit** to create the credential record.
4. Navigate to **All** > **Connections & Credentials** > **Connection & Credential Aliases**.

5. Select **New** and enter these fields values.
   a. For **Name**, enter Local Instance.
   b. Accept the default value of HTTP for the **Connection type**.
   c. Select **Submit** to create the Connection & Credential Alias record.
6. Select the alias you created.

   Example
   For example, select **Local Instance**.
7. From the Connections related list, select **New**, and enter these field values.

   a. For **Name**, enter `My Instance`.

   b. For Credential, select the basic authentication credential record you created.

      Example
      For example, select the **Local Admin** credential.

   c. For Connection URL, enter the base URL for your instance including the forward slash at the end.
      Include the URL prefix https:// and add a slash character at the end of the URL.

      Example
      For example, `https://example.service-now.com/`.

   d. Select **Submit** to create the HTTP(s) Connection record.

Create a data gathering action to get a record schema

Create a data gathering action to look up a table schema and convert into a JSON object.

Before you begin
Role required: action_designer or admin

About this task
In this task, you create a data gathering action that collects the schema for a record on your instance. The goal is to create a complex object for use as a dynamic output. This data gathering action consists of the following:

- A REST step to gather table schema data for a selected table. The REST step Response Body is in JSON format.
- A script step to transform the REST step's JSON Response Body into a dynamic object. The dynamic object consists of JSON name-value pairs, where there is an entry for each field in the table.
- An output variable named `output` of type JSON to store the dynamic object.

> **Note:**
> This task re-creates the demo actions that are installed when you Request an Integration Hub plugin for your instance.

Procedure
1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Actions**.

3. Select **New** and select **New Action**.
   a. On the Action Properties screen, in the **Name** field, enter `Get ServiceNow Object Schema (Dynamic)`.
   b. Select **Submit**.
4. In the Action Outline, select **Inputs**.
   a. In the Action Input header, select **Create Input**.
   b. In the **Label** and **Name** fields, enter `Table`.
   c. In the **Type** field, select `String`.
   d. To make this input required, toggle the **Mandatory** slider so that it is active.
5. In the Action Outline, select the add a new step icon ( ) under Inputs and select **REST Step**.
6. Under the REST step header, fill in the following fields.

Field	Value
Connection	Leave the **Use Connection Alias** option selected.
Connection Alias	select the create new record icon ( ) to create a new Create an HTTP(s) connection, or use an existing connection for your instance. The **Credential** for the HTTP(s) connection must use Basic authentication credentials. Additionally, the **Connection URL** must be the base URL for your instance, including the forward slash at the end. For more information on connections and credentials, see Getting started with connections and Getting started with credentials.
Build Request	Leave the **Manually** option selected.
Resource Path	Enter `api/now/processflow/table/` and then select the data pill picker ( ). Select **Inputs** > **Table**. Next, enter `/schema`.
HTTP Method	Enter `GET`
Query Parameters	select the plus icon ( ) to add a new query parameter. Then, in the **Name** field, enter `get_choices` and `true` in the **Value** field.

7. In the Action Outline, select the add a new step icon ( ) under your REST step and select the **Script** step.
   a. In the Input Variables section, select **Create Variable**.
   b. In the **Name** field, enter `payload`.

c. Next to the **Value** field, select the data pill picker ( ) and select **REST Step** > **Response Body**.

d. In the **Script** field, enter the following code.

```
(function execute(inputs, outputs) {
  var payload = JSON.parse(inputs.payload);

  var columns = payload.result.data.columns;
  var schema = columns.map(function(column) {
    var value = {
      label: column.label,
      name: column.name,
      type: getCOType(column.definition.base_type),
    };
    if (column.definition.type === 'choice') {
      value.type = 'choice';
      value.choices = column.definition.choices;
    }
    if (column.definition.base_type === 'GUID') {
      value.children = [
        { label: 'Link', name: 'link', type: 'string' },
        { label: 'Value', name: 'value', type: 'string' },
      ];
    }
    return value;
  });
  outputs.schema = {
    data: {
      type: 'object',
      children: schema,
    },
  };

  function getCOType(type) {
    if (type === 'GUID') return 'reference';
    return type;
  }
})(inputs, outputs);
```

e. In the Output Variables section, select **Create Variable**.

f. In the **Label** and **Name** fields, enter `schema`.

g. In the **Type** field, select `JSON`.

8. In the Action Outline, select **Outputs**.

a. In the Action Output header, select **Create Output**.

b. In the **Label** and **Name** fields, enter `output`.

c. In the **Type** field, select `JSON`.

d. In the Action Output header, select **Exit Edit Mode**.

e. Next to the **Value** field, select the data pill picker ( ) and select **Script Step** > **schema**.

9. In the Action header, select **Save** and then select **Test** to test the action.

a. On the Test Action screen, enter `incident` for the **Table** input.

b. Select **Run Test**.

c. Check the action's execution details.
Your data gathering action runs successfully if the runtime value for `fields` is a complex object in a format that is similar to the following abbreviated example.

```
{
       "data": {
           "type": "object",
           "children": [
               {
                   "name": "active",
                   "label": "Active",
                   "type": "boolean"
               },
               {
                   "name": "activity_due",
                   "label": "Activity due",
                   "type": "datetime"
               }, ...
```

10. In the Action header, select **Publish** to make the *`Get ServiceNow Object Schema (Dynamic)`* action available to other flows and actions within the Global scope.

Create a data gathering action to get an array of records schema

Create a data gathering action to generate an array of objects from a list of records.

Before you begin
Role required: action_designer or admin

About this task
In this task, you create a data gathering action that collects the schema for a record on your instance. The goal is to create a complex object for use as a dynamic output. This data gathering action consists of the following:

- A REST step to gather table schema data for a selected table. The REST step Response Body is in JSON format.
- A script step to transform the REST step's JSON Response Body into a dynamic object. The dynamic object consists of a JSON array of objects, where each source record is converted into one object of the array.
- An output variable named `output` of type JSON to store the dynamic object.

Note:
This task re-creates the demo actions that are installed when you Request an Integration Hub plugin for your instance.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.

2. On the homepage, select **Actions**.

3. Select **New** and select **New Action**.
   a. On the Action Properties screen, in the **Name** field, enter `Get ServiceNow Array.Object Schema (Dynamic)`.
   b. Select **Submit**.
4. In the Action Outline, select **Inputs**.
   a. In the Action Input header, select **Create Input**.
   b. In the **Label** and **Name** fields, enter `Table`.
   c. In the **Type** field, select `String`.
   d. To make the input required, toggle the **Mandatory** slider so that it is active.
5. In the Action Outline, select the add a new step icon ( ) under Inputs and select the **REST** step.
6. Under the REST step header, fill in the following fields.

Field	Value
Connection	Leave **Use Connection Alias** selected.
Connection Alias	select the create new record icon ( ) to create a new Create an HTTP(s) connection, or use an existing connection for your instance. The **Credential** for the HTTP(s) connection must use Basic authentication credentials. Additionally, the **Connection URL** must be the base URL for your instance, including the forward slash at the end.
Build Request	Leave **Manually** selected.
Resource Path	Enter `api/now/processflow/table/` and then select the data pill picker ( ). Select **Inputs** > **Table**. Finally, enter `/schema`
HTTP Method	Enter `GET`
Query Parameters	select the plus icon ( ) to add a new query parameter. Then, in the **Name** field, enter `get_choices` and `true` in the **Value** field.

7. In the Action Outline, select the **Add a new step**( ) icon under your REST step and select the **Script** step.
   a. In the Input Variables section, select **Create Variable**.
   b. In the **Name** field, enter `payload`.
   c. Next to the **Value** field, select the data pill picker ( ) and select **REST Step** > **Response Body**.
   d. In the **Script** field, enter the following code.

```
(function execute(inputs, outputs) {
  var payload = JSON.parse(inputs.payload);

  var columns = payload.result.data.columns;
  var schema = columns.map(function(column) {
    var value = {
      label: column.label,
      name: column.name,
      type: getCOType(column.definition.base_type),
    };
    if (column.definition.type === 'choice') {
      value.type = 'choice';
      value.choices = column.definition.choices;
    }
    return value;
  });
  outputs.schema = {
    data: {
      type: 'array.object',
      attributes: {
        child_type: 'object',
      },
      children: schema,
    },
  };

  function getCOType(type) {
    if (type === 'GUID') return 'string';
    return type;
  }
})(inputs, outputs);
```

e. In the Output Variables section, select **Create Variable**.

f. In the **Label** and **Name** fields, enter `schema`.

g. In the **Type** field, select `JSON`.

8. In the Action Outline, select **Outputs**.
   a. On the Action Output header, select **Create Output**.
   b. Enter `output` in the **Label** field and **Name** field.
   c. Select `JSON` for the **Type** field.
   d. Select **Exit Edit Mode**.
   e. Next to the **Value** field, select the data pill picker ( ) and select **Script Step** > **schema**.
9. In the Action header, select **Save** and then select **Test** to test the action.
   a. On the Test Action screen, in the **Table** field, enter `incident`.
   b. Select **Run Test**.

c. Check the action's execution details.
Your data gathering action runs successfully if the runtime value for `fields` output is a complex object that contains an array of key-value pairs for `label`, `name`, and `value` as shown in the following abbreviated example.

```
{
       "data": {
           "type": "array.object",
           "children": [
               {
                   "name": "active",
                   "label": "Active",
                   "type": "boolean"
               },
               {
                   "name": "activity_due",
                   "label": "Activity due",
                   "type": "datetime"
               }, ...
```

10. In the Action header, select **Publish** to make the *Get ServiceNow Array.Object Schema (Dynamic)* action available to other actions within the Global scope.

Create a custom action to test dynamic outputs

Create a sample action to dynamically generate two action outputs, `Record` and `Records` which refresh dynamically when the value for the **Table** input changes.

Before you begin
Role required: action_designer or admin

About this task
This custom action uses two data gathering actions to populate dynamic outputs.

Procedure

1. In the main header, select the create flow, subflow, or action icon ( ) and select **Action**.
   a. In the Action Properties modal, in the **Name** field, enter `Get ServiceNow Records (Dynamic)`.
   b. Select **Submit**.
2. In the Action Outline, select **Inputs**.
   a. In the Action Input header, select **Create Input**.
   b. In the **Label** and **Name** fields, enter `Table`.
   c. In the **Type** field, select `Dynamic Choice`.
   d. To make the input required, toggle the **Mandatory** slider so that it is active.
   e. Select the Toggle advanced inputs icon ( ) to display the advanced options for the `Table` input.
   f. In the **Default value** field, enter `incident`.
   g. Under Dynamic Options, in the **Action** field, select **Get ServiceNow Tables - Dynamic**.
   h. Select **Create Input** to create another action input.

i. In the **Label** and **Name** fields, enter `NumberOfRecords`.

j. In the **Type** field, select **Integer**.

k. To make the input required, toggle the **Mandatory** slider so that it is active.

l. Select the Toggle advanced inputs icon ( ) to display the advanced options for the `Table` input.

m. In the **Default value** field, enter `3`.

3. In the Action Outline, select the add a new step icon ( ) under Inputs and select the **REST** step.
4. Under the REST step header, fill in the following fields.

Field	Value
Connection	Leave **Use Connection Alias** selected.
Connection Alias	Select the create new record icon ( ) to create a new Create an HTTP(s) connection, or use an existing connection for your instance. The **Credential** for the HTTP(s) connection must use Basic authentication credentials. Additionally, the **Connection URL** must be the base URL for your instance, including the forward slash at the end.
Build Request	Leave **Manually** selected
Resource Path	Enter `api/now/table/` and then select the data pill picker ( ). Select **Inputs** > **Table**.
HTTP Method	Enter `GET`
Query Parameters	Select the plus icon ( ) to add a new query parameter. Then, enter `sysparm_limit` in the **Name** field. Next to the **Value** field, select the data pill picker ( ) and then select **Inputs** > **NumberOfRecords**.

5. In the Action Outline, select the add a new step icon ( ) under **Inputs** and select the **Script** step.

a. In the Input Variables section, select **Create Variable**.

b. In the **Name** field, enter `payload`.

c. Next to the **Value** field, select the data pill picker ( ) and select **REST Step** > **Response Body**.

d. In the **Script** field, enter the following code.

```
(function execute(inputs, outputs) {
```

```
  var response = JSON.parse(inputs.payload);
  var records = response.result;
  outputs.record = records[0];
  outputs.records = JSON.stringify(records);
})(inputs, outputs);
```

e. In the Output Variables section, select **Create Variable**.

f. In the **Label** and **Name** fields, enter `record`.

g. Select **JSON** for the **Type** field.

h. Toggle the **Mandatory** slider so that it is active.

i. Select **Create Variable** to create another output variable for the script step.

j. In the **Label** and **Name** fields, enter `records`.

k. In the **Type** field, select `JSON`.

l. To make the input required, toggle the **Mandatory** slider so that it is active.

6. In the Action Outline, select **Outputs**.
   a. In the Action Output header, select **Create Output**.
   b. In the **Label** and **Name** fields, enter `Records`.
   c. In the **Type** field, select **Dynamic Object**.
   d. Select the Toggle advanced inputs icon ( ) to display the advanced options for the `Records` output.
   e. Under Dynamic Options, select **Get ServiceNow Array.Object Schema (Dynamic)** as the **Action**.
   f. To make the Table input dependent on another input, toggle the **Depends-On Another Input** slider to make it active.
   g. In the **Table** field, select **Table**.
   h. In the Action Output header, select **Exit Edit Mode**.
   i. Next to the **Value** field, select the data pill picker ( ) and select **Script Step** > **records**.
   j. In the Action Output header, select **Edit Outputs** > **Create Output** to create another action output.
   k. In the **Label** and **Name** fields, enter `Record`.
   l. In the **Type** field, select `Dynamic Object`.
   m. Select the Toggle advanced inputs icon ( ) to display the advanced options for the `Record` output.
   n. Under Dynamic Options, in the **Action** field, select **Get ServiceNow Object Schema (Dynamic)**.
   o. To make the Table input dependent on another input, toggle the **Depends-On Another Input** slider to make it active.

p. In the **Table** field, select **Table**.

q. In the Action Output header, select **Exit Edit Mode**.

r. For the **Value**, select the data pill picker ( ) and select **Script Step** > **record**.

7. In the Action header, select **Save** and then select **Test** to test the action.

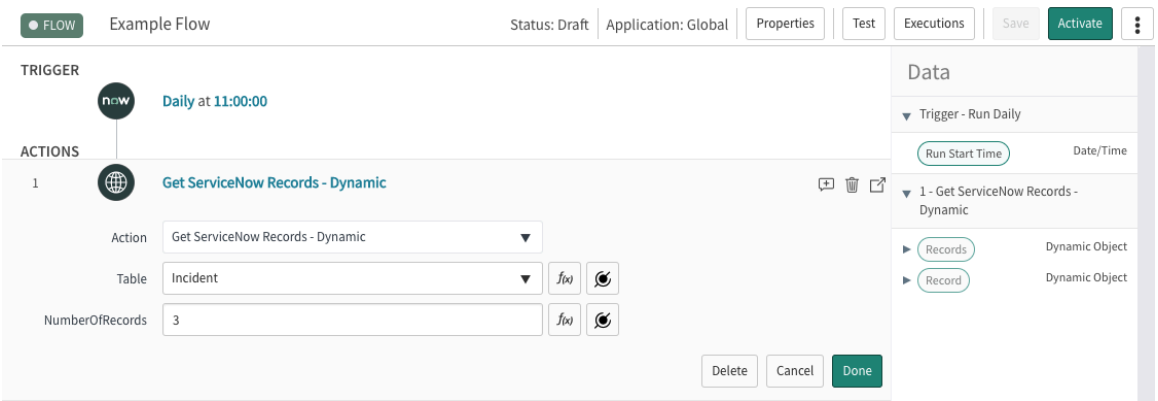
   a. On the Test Action screen, select any dynamically generated choice value for the **Table** input.

   b. Select **Run Test**.

   c. Check the action's execution details.
   Your action runs successfully if the runtime value for `Record` is a properly formatted complex object and the runtime value for `Records` is a properly formatted complex object array.

8. In the Action header, select **Publish** to make the *`Get ServiceNow Records (Dynamic)`* action available to flows within the Global scope.

Result



You can now add the *`Get ServiceNow Records (Dynamic)`* action to a flow. This sample action dynamically generates two action outputs, `Record` and `Records`, which can be accessed as data pills in the data panel. The data pills refresh dynamically when the value for the **Table** input changes.

Create a data gathering action for a dynamic object

Create an action to collect output values. Then, pass the values to a parent action as a dynamic object.

Before you begin

Role required: action_designer or admin

Note:
Dynamic outputs are not available in the base system. To use dynamic outputs in Workflow Studio, you must Request an Integration Hub plugin.

About this task

These steps allow you to create a generic data gathering action for a dynamic output. To see working examples of data gathering actions for dynamic outputs, see Get started with dynamic outputs.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Actions**.
3. Select **New** > **New Action**.
4. In the **Name** field, enter a name for your action, choose the proper **Application** scope, and then select **Submit**.
5. In the Action Outline, select the add a new step icon ( ) under Inputs and select the **REST** step.
   Configure your REST step to get data from the proper Base URL and Resource Path with any applicable Query Parameters for the HTTP Method GET. For more information on using the REST step in Integration Hub, see REST step and REST in IntegrationHub .
6. In the Action Outline, select the add a new step icon ( ) under your REST step and select the **Script** step.
   This script step must transform the response from the REST step's Response Body into a format that defines the schema of a dynamic object output for a parent action. Your script step must:
   - Have a single JSON output variable. The script step can have other output variables, but only one must be of type JSON.
   - Format the JSON output variable so that it contains a property named `data`. For a complex object output, your `data` property must have a format similar to the following example:

```
{
       data: {
           type: "object",
           //Required

           children: [
           //Required - This is a collection of field
 definitions

               {
                   name: "Name 1",
                   //Required - Unique name

                   label: "Label 1",
                   //Required - Display name

                   type: "string"
                   //Required - Supported field type (See the
 Note below)

               },

               {
                   name: "Name 2",
                   label: "Label 2",
                   type: "string"
               }
           ]
       }
   }
```

For a complex object array output, your `data` property must have a format similar to the following example:

```
{
       data: {
           type: "object",
           //Required

           attributes: { child_type: "object" }
           //Required - Indicates that the array's children
 are of type object

           children: [
           //Required - This is a collection of field
 definitions

               {
                   name: "Name 1",
                   //Required - Unique name

                   label: "Label 1",
                   //Required - Display name

                   type: "string"
                   //Required - Supported field type (See the
 Note below)
               },

               {
                   name: "Name 2",
                   label: "Label 2",
                   type: "string"
               }
           ]
       }
   }
```

Note:
Supported action data types for the `type` property include:

- `string`
- `integer`
- `datetime`
- `choice`
- `boolean`
- `object`
- `array.string`
- `array.integer`
- `array.datetime`
- `array.choice`
- `array.boolean`
- `array.object`

7. In the Action Outline, select **Outputs**.

   a. Create an output of type **Dynamic Output**.

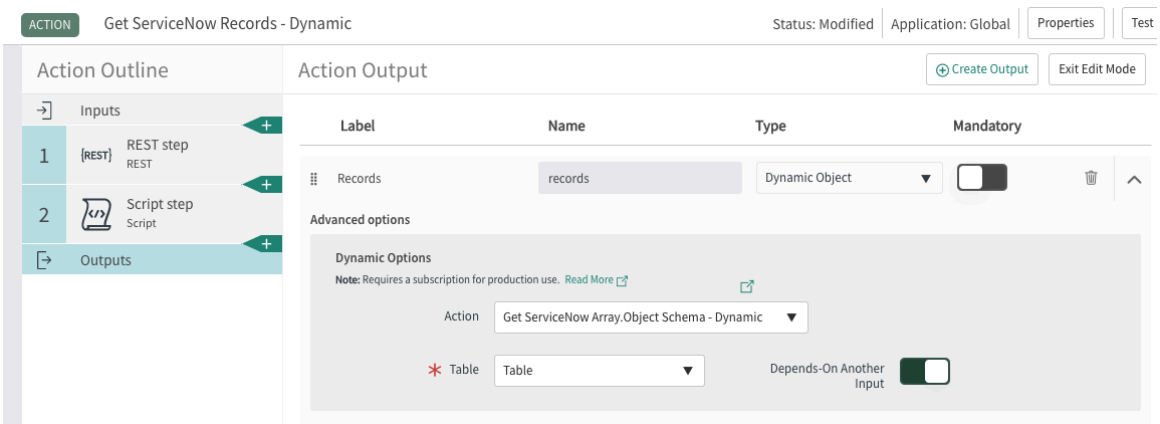
   > **Note:**
   > If you don't see Dynamic Outputs as a data type option, then you're missing an Integration Hub subscription.

   b. In the Dynamic Options, select a data gathering action for the **Action** option.
   For examples of dynamic output data gathering actions, see Get started with dynamic outputs.

   c. **Optional:** If needed set the **Depends On Another input** option, and select the input.

   Create an outputs of type Dynamic Object.

8. Select **Save** and test the action.
   In the execution details, your data gathering action runs successfully if the runtime value for `output` contains the `data` property in the proper format.
9. Select **Publish** to make the action available to other flows or actions within the same application scope.



You can now use your data gathering action to populate the schema values for a dynamic object in a parent action.

Dynamic output configuration options

Use these options to configure your dynamic outputs for a parent action.

Dynamic output configuration

Dynamic output options

Option	Description
Label	Enter a label that appears within the output data pill for the action.
Name	Enter a descriptive name for the dynamic object.
Action	Select a data gathering action that generates values as JSON output.
Depends-on Another Input	Enable this option to require an input value from the parent action to be passed as an input to the data gathering action. If enabled, select a dependent input from the parent action.

Action error evaluation

Enable actions to catch step failures and continue running. Identify when specific error conditions occur and return your own action status code, status message, and error state.

Benefits

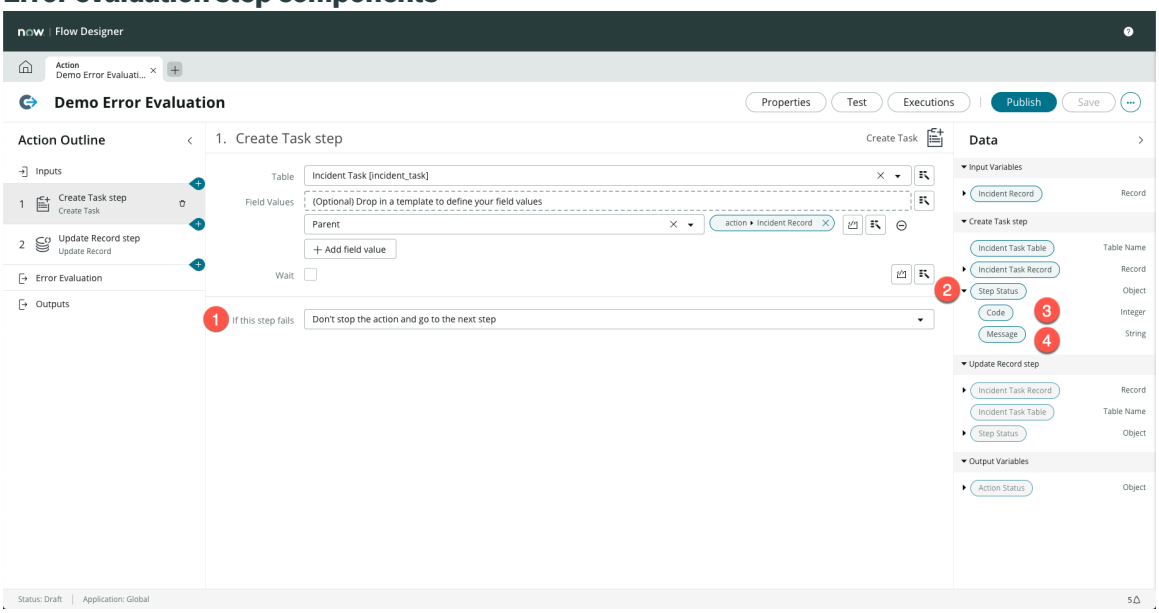
Enable action error evaluation to gain these benefits.

- Catch step failures and allow an action to continue running. Specify the failure behavior of each step you add to an action.
- Create your own error conditions. Specify when an action returns an error state as well as the status codes and messages they return.
- Provide more error handling information and options to flow designers. Use your own action status codes and messages to identify issues and provide details for corrective actions.

Action error evaluation step components

Each step offers these error evaluation components.

Error evaluation step components



1. If this step fails option

Option to continue running the next step or go to error evaluation. This option has no effect on the Step Status.

2. Step Status

Object data pill containing runtime details about the step. Each step in an action returns a Step Status.

3. Step Status > Code

Integer data pill indicating whether the step produced an error. A step returns a value of 1 when it produces an error for any reason. For example, a step can produce an error if it is missing mandatory input data or returns output in the wrong data type. A step returns a value of 0 when it runs successfully. You cannot customize these codes.

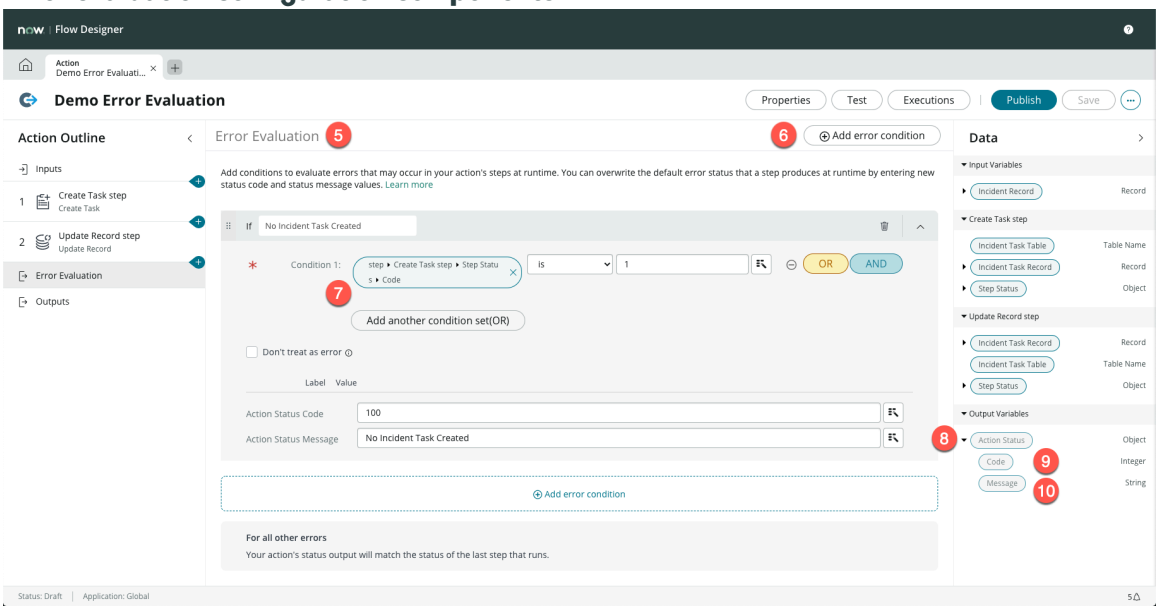
4. Step Status > Message

String data pill containing the error message produced by the step or system operation. You cannot customize the step status message.

Action error evaluation configuration components

Action error evaluation consists of these configuration components.

Error evaluation configuration components



5. Error Evaluation section

Section containing possible error conditions. When an action runs, it evaluates the available error conditions from top to bottom looking for a possible match. An action returns the Action Status specified by the first matching error condition. If there is no matching error condition, the Action status is set to the Step Status of the last step run.

> **Note:**
> Data stream actions do not have an error evaluation section.

6. Add error condition option

Option to add an error condition. Each error condition is equivalent to an else if flow logic block. Only one error condition can be true at a time.

7. Error condition configuration

Options available to configure an error condition.

- Label you want to use to identify this error condition
- Conditions that must be met to match this error condition
- Error state you want the action to return to flow
- Action Status Code you want the action to return to flow
- Action Status Message you want the action to return to flow

8. Action Status

Object data pill containing runtime details about the action. An action always returns an Action Status.

9. Action Status > Code

Integer data pill containing the code returned by the first matching error condition or the last step run. You can return your own code when you create a custom error condition.

10. Action Status > Message

String data pill containing the message produced by a matching error condition or the last step run. You can return your own message when you create a custom error condition.

Flow and action error handling resources

For more information about using error handling in actions and flows, see the ServiceNow® Community post Flow and Action Error Handling Overview: Why and how to test for errors - Worfklow Automation CoE.

- Flow and Action Error Handling Level 1: Retry and Action Error Evaluation - Workflow Automation CoE
- Flow and Action Error Handling Level 2: Flow Logic - Workflow Automation CoE
- Flow and Action Error Handling Level 3: Flow Error Handling - Workflow Automation CoE
- Flow and Action Error Handling Level 4: Good Practices and Summary - Workflow Automation CoE

General guidelines

Follow these general guidelines to achieve the benefits offered by action error evaluation.

Allow only independent steps to continue running

Allow a step to continue running if it does not return data required by a later step. If a step provides data necessary for later steps, then you know that the later steps cannot run successfully.

Avoid more than 10 error conditions

While there is no limit to the number of error conditions you can create, each error condition requires evaluation. The more error conditions your action has to evaluate, the potentially slower your action can run.

Identify specific step failures

You can use the Step Status to identify when a specific step fails. Identifying a specific step can be useful when your action contains multiple instances of the same type of step. You may also want to identify a specific step so that a flow error handler can take specific corrective actions for the failure.

Put specific error conditions before general error conditions

Error evaluation stops when the action finds a matching error condition. Putting general error conditions first may prevent the action from ever matching specific error conditions.

Use descriptive error condition labels

Identify an error condition without having to edit it. By default, you can only see error conditions when you edit them.

Add error condition

Enable an action to return custom status information when specific conditions are met. Specify whether a flow considers your custom action status as an error or a successful run.

Before you begin

- Role required: flow_designer, action_designer, or admin
- Create an action in Workflow Studio

About this task

An action always returns an Action Status object. When an action runs, it evaluates the available error conditions from top to bottom looking for a possible match. An action returns the Action Status specified by the first matching error condition. If there is no matching error condition, the Action status is set to the Step Status of the last step run.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select the **Actions** tab and select the custom action you want to edit.
3. From the Action Outline, select **Error Evaluation**.
4. Select **Add error condition** for each error condition you want to define.
   Workflow Studio adds an If block for configuration.
5. Configure each error condition.

Field	Description
If	Description of your error condition. You can use this label to identify the error condition when the condition builder is not visible. Each error condition is equivalent to an else if flow logic block. Only one error condition can be true at a time.
Condition N	Condition builder to specify the matching criteria for the error condition. You can add conditions or condition sets to define the matching criteria. For each condition, select an action data pill to evaluate. For each data pill to evaluate, select an operator and matching value. **Note:** Workflow Studio only displays the operators and values available for the type of data pill you selected.
Don't treat as error	Option to report the Action Status as an error or a success when returned to a flow.
Action Status Code	Integer value you want the action to return as part of the Action Status object. You can use this integer value as part of a Flow error handler.
Action Status Message	String value you want the action to return as part of the Action Status object. You can use this string value as part of a Flow error handler.

6. Order error conditions from top to bottom in the order you want the action to evaluate them.
7. Select **Save**.

Result

Your custom action evaluates each error condition for matching conditions. The action returns the Action Status Code and Action Status Message for the first matching error condition. If there is no matching error condition, the Action status is set to the Step Status of the last step run.

Retry policy

Automatically retry failed requests when a step encounters an intermittent issue such as a network failure or request rate limit. Set a retry policy to prevent having to manually trigger the step again.

Features

Retry policies can be:

- Created to support connection timeouts or failed requests based on header, status, response body, error, and HTTP method.
- Applied to all actions that use a given connection alias.
- Applied directly to an action step.

Use retry policies to define:

- The conditions that must be met to retry a step.
- The time interval to wait before retrying a step.
- The maximum number of retry attempts the step makes before stopping.

Associate a default retry policy to a Connection & Credentials alias and apply the retry policy to all HTTP connections.

Note:
You can only create retry policies for JDBC, REST, and SOAP steps.

Create a retry policy

Automatically retry failed requests when a step encounters an intermittent issue such as a network failure or request rate limit. Set a retry policy to prevent having to manually trigger the step again.

Before you begin

- Role required: connection_admin or credential_admin

Procedure

1. Navigate to **All** > **IntegrationHub** > **Retry Policy** > **Create New**.
2. On the form, fill in the fields.

Retry Policy form

Field	Description
Name	Name to uniquely identify the retry policy.
Connection Type	**HTTP**
Condition	Conditions that must be met to trigger the retry policy. Conditions that trigger a retry policy include the **is**, **is not**, **contains**, and **contains not** operators.

Field	Description
Retry Strategy	○ **Exponential Backoff**: Option to exponentially increase the time interval for the subsequent retry attempts. The multiplier is 2. ○ **Fixed Interval**: Option to specify a fixed time interval after which a retry attempt should be made. ○ **Honor "Retry-After" Header**: Option to specify a retry attempt based on the date and time value returned in the Retry After header value of the HTTP request. For more information about the header, see RFC 7231, section 7.1.3: Retry-After. **Note:** **Honor "Retry-After" Header** supports only REST and SOAP steps.
Interval (seconds)	Time interval in seconds after which a retry attempt should be made. This field applies only to **Exponential Backoff** and **Fixed Interval** retry strategies. **Note:** If **Retry Strategy** is **Exponential Backoff**, the time interval exponentially increases after every retry attempt till the maximum numbers of attempts is reached.
Count	Maximum number of retry attempts. This field applies only to **Exponential Backoff** and **Fixed Interval** retry strategies. If no value is specified, the maximum number of retry attempts is based on the value provided in the *glide.fdih.retry.max_count* system property. Default value of the *glide.fdih.retry.max_count* system property is 0. For more information about system properties, see Available system properties.
Max Elapsed Time (seconds)	Maximum cumulative time in seconds after which the retry attempts are stopped. This field appears only when **Honor "Retry-After" Header** is selected from **Retry Strategy**. **Note:** If the maximum retry time is specified in the *glide.fdih.retry.max_time_in_seconds* property, the system property value takes precedence over this field value. Also, make sure that the max elapsed time is equal to or greater than the system property value.

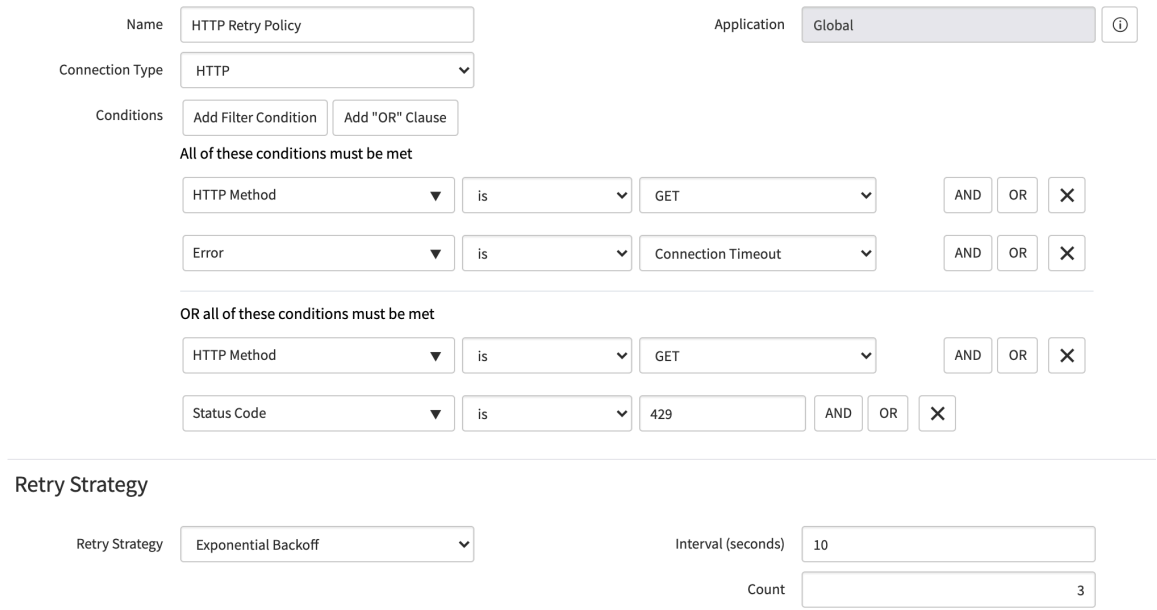
3. **Optional:** Create a global system property with the following attributes.
   For more information on how to create a property, see Add a system property. You can use this system property to specify the maximum time in seconds for a retry policy.

Field	Value
Name	`glide.fdih.retry.max_time_in_seconds`
Type	`integer`
Value	Default value: `86400` (seconds) Maximum supported time: `604800` (seconds)

4. Click **Submit**.

Example: Retry policy with Retry Strategy as Exponential Backoff

Sample retry policy



In this example, the policy is defined to attempt retry when one of these conditions is met:

- **HTTP Method is GET and Error is Connection Timeout**
- **HTTP Method is GET and Status Code is 429**

When the condition is met, retry attempts are made for a maximum number of three times. The time interval between the retry attempts is exponentially increased. The time intervals in this example are 10 seconds, 20 seconds, and 40 seconds.

What to do next

- Create a Connection & Credential alias, if you do not have the required alias.
- Assign the retry policy as **Default Retry Policy** to the required Connection & Credential alias.

> **Note:**
> A default retry policy is provided and is selected as **Default Retry Policy**. If you have created retry policies, you can select the required policy as **Default Retry Policy**.

- Create an HTTP(s) Connection in the Connections related list for the Connection & Credential alias. For more information, see Connections and Credentials.
- Verify and view the details of the retry attempts by navigating to **System Logs** > **Outbound HTTP Requests**.

Complex data

Use a graphical interface to work with collections of complex structured data. Help design users understand the organization of structured data, and add, remove, or configure its individual elements.

Complex data allows you to encode and store structured data in a machine-readable format such as JavaScript Object Notation (JSON) or eXtensible Markup Language (XML). You can use the Workflow Studio interface to view and understand the organization of structured data as well as create data structures. For example, you can create a contact data structure consisting of information you look up from a user record such as first name, last name, and email address.

You can create complex data from these Workflow Studio interfaces.

Complex data usage examples

Workflow Studio interface	Example usage
Action inputs and outputs	Create an action that generates an object from record data. Populate the object using record data the action looks up. See Create a custom action to generate an object from a record
Script step input and output variables	Write a script to create an array of objects from a list of records. See Create a custom action to generate an array of objects from a list of records. Write a script to parse a JSON document into an output of type Object.
Subflow inputs and outputs	Create a subflow that accepts an object as an input and uses it to call an external service.
XML parser step Target field configurations	Parse an XML payload into a complex data object using the XML parser step.
REST API Trigger Body field	Parse an inbound REST API request into a complex data object and use the values in a flow. See REST API trigger.

Benefits

Complex data offers these benefits.

- Parse and format data without having to write code. For example, create data variables to parse a response message or format a request message.
- Create arbitrary data structures. For example, create an issue data structure that combines information from existing interaction and incident records, or create a data structure to support a custom integration.
- View the organization of data structures. For example, an issue data structure might consist of a user object to describe who to contact about the issue and a history object to describe the work done to resolve it. You could configure a notification action with the path to the email address listed in the user object, and call an escalate issue subflow with the path to the status or reassign count from the history object.
- Allow access to data structure from API calls. For example, call an action or subflow from a script and use the predefined data structure as input values.
- Save and reuse data structures as templates. For example, save the user object as a template data structure for reuse in other actions and flows.

Data structure

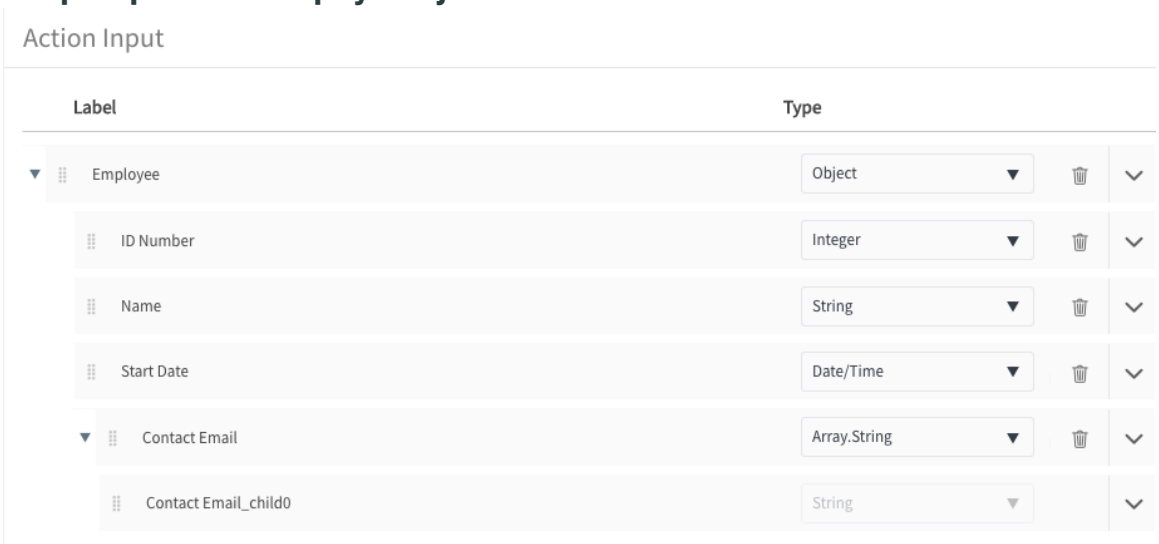
A data structure is a collection of related data elements organized into a hierarchy. Each element in a data structure has its own data type and its own unique position in the hierarchy.

The Editor pane displays data structure hierarchy with indentation. The indentation level identifies whether an element is a parent, child, or sibling in the hierarchy. Parent elements have children indented underneath them, and siblings have the same indentation level.

Note:
The Workflow Studio interface allows you to create hierarchies with an unlimited number of child levels, but you may have to scroll horizontally to see them.

For example, this data structure consists of an Employee parent element with four child elements for ID Number, Name, Start Date, and Contact Email. The Contact Email element is also a parent element with one child.

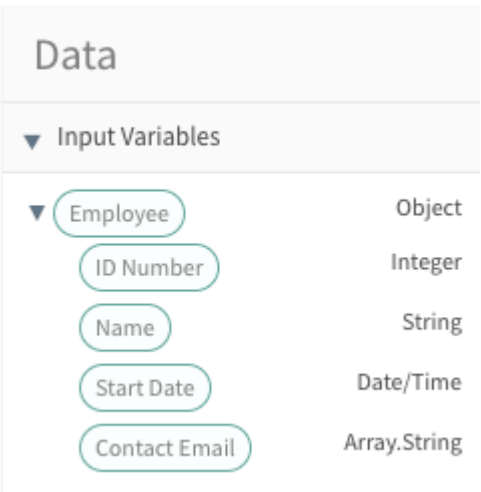
Sample inputs for an Employee object



The data panel displays data structure hierarchy as a tree of collapsible and expandable data pills just like it does with record variables. Parent elements have an arrow icon to collapse or expand the hierarchy.

For example, here is the Employee data structure as seen from the data panel.

Sample data pill for an Employee object



You can use the data panel or Data picker to select specific values from a data structure. Data structures are similar to data pills for records in that you can dot-walk or navigate to specific elements within the structure. When you select a data element, Workflow Studio displays the path to it as a data pill just like any other data element selection. For example, if you select the Start Date data element, the path is **[Input->Employee->Start Date]**.

You can use an element data path the same way you can an XPath or JPath. Sometimes you may even convert the data pill path into one of these path notations.

Complex data types

You build data structures using one or more Array or Object variables. Only these variable data types support child variables.

An Array variable contains values for one type of item. The parent variable is always of an Array data type, and there is always only one child variable, which is one instance of the data type supported by the array. Create Array variables when an input or output accepts multiple values of the same data type.

For example, you could create a Contact Email array to list all the email addresses associated with a given person.

Sample data structure for the Contact Email array

Component label	Data Type	Sample Data
Contact Email	Array.String	beth.anglin@example.com and beth@anglin.com
Contact Email_child0	String	

An Object variable contains any number and arrangement of child variables that each have their own data type and values. Nesting Object variables allows you to create complex data structures similar to a table schema where one table has related records in another table. Create Object variables when an input or output accepts one or more related properties.

For example, you can create an Employee object to define information about the people who work at a company.

Sample data structure for the Employee object

Component label	Data Type	Sample Data
Employee	Object	
ID Number	Integer	20190304000101
Name	String	Beth Anglin
Start Date	Date/Time	March 4, 2019
Contact Email	Array.String	beth.anglin@example.com and beth@anglin.com

Only these variable data types can be parents.

Parent data types

Data type	Description
Array.Array	A container for arrays. Adds a read-only child item of type Array.
Array.True/False	A container for true/false values. Adds a read-only child item of type True/False.
Array.Choice	A container for choice values. Adds a read-only child item of type Choice.
Array.Date/Time	A container for date/time values. Adds a read-only child item of type Date/Time.
Array.Integer	A container for integer values. Adds a read-only child item of type Integer.

Parent data types (continued)

Data type	Description
Array.Object	A container for objects. Adds a read-only child item of type Object, which displays the **Add Child Item** option.
Array.String	A container for string values. Adds a read-only child item of type String.
Object	A container for other data elements. Displays the **Add Child Item** option.

Array and Object variables only support these child data types.

- Array
- True/False
- Choice
- Date/Time
- Integer
- Object
- String

Note:
The data types in array and object variables are not Glide elements. There may not be complete compatibility between these items and ServiceNow AI Platform types.

Advanced options

Object variables have advanced options to save and load data structures. These options allow you to reuse a data structure defined in one location in another. For details on advanced options see action variable data types.

Data structure templates

Data structure templates allow you to reuse Object variables in multiple actions or subflows. For example, you can create a data structure to parse a response and then later reuse that same data structure to format a request. A template stores the list of child variables and their structure within an object. Each Object variable has an Advanced Option to save it as a template.

When you apply a template, you are creating a copy of the original structure. Any changes you make after applying a template do not affect the template, nor do they affect other actions that use the template.

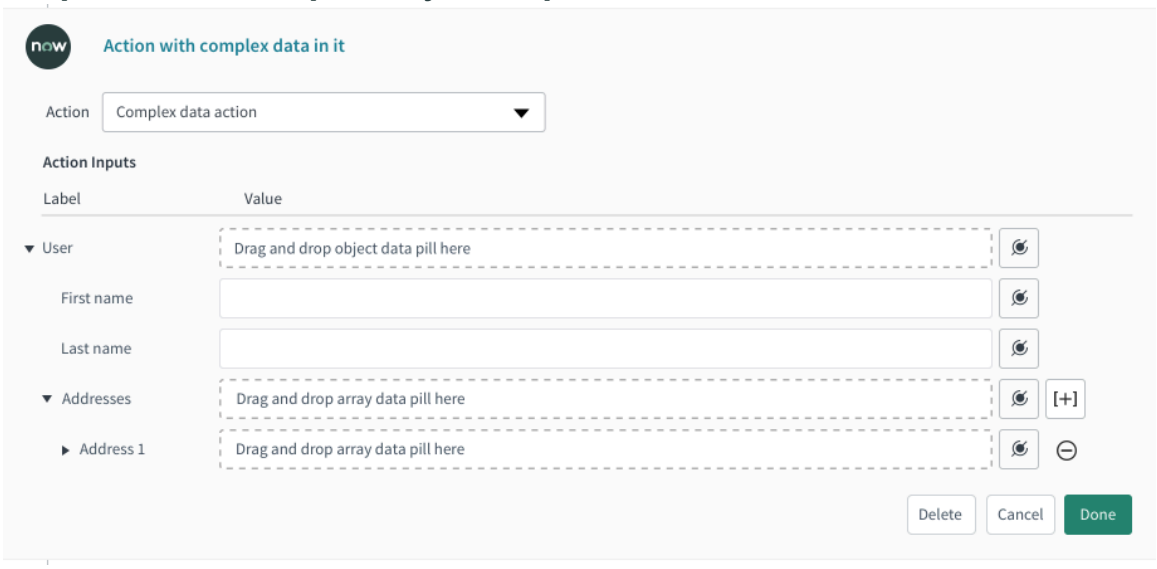
Array data pills

Objects that contain array data may require **For Each** flow logic to process. For example, a user object that contains an array of email addresses would require a **For Each** flow logic loop to send a notification to each email address.

Object data pills

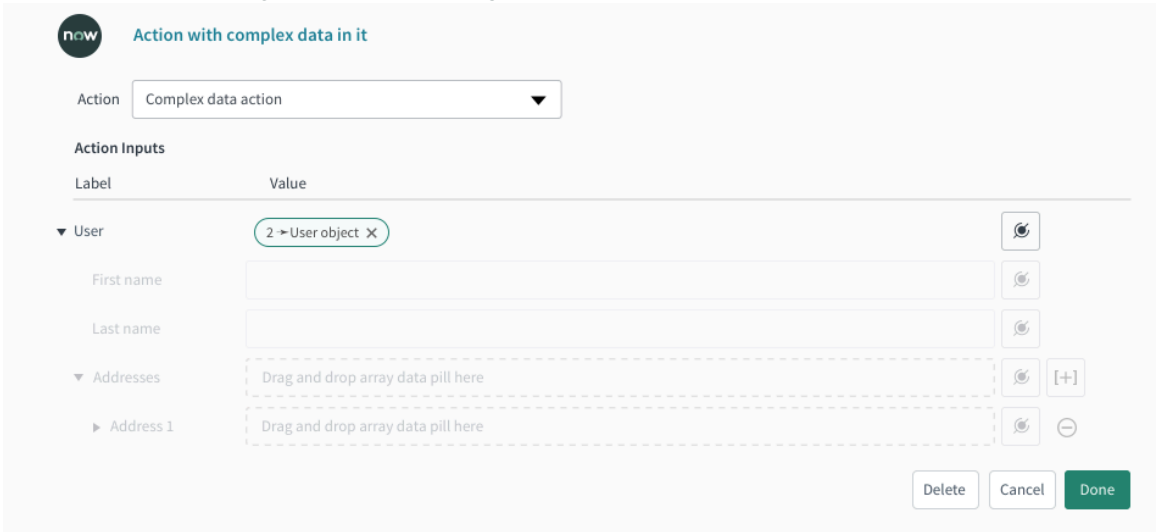
You can design actions that accept object data pills as input values. For example, you might create a notification action that accepts a user object as an input. If the user object consists of values for first name, last name, and addresses, then the notification action has access to all these values. To configure an action input with an object data pill, you must create an object earlier in the flow.

Sample action that accepts an object data pill



You can use an object data pill or any of its child elements to configure an input. When you configure an input value with an object data pill, Workflow Studio makes any child elements of the object read-only, and the action uses the values provided by the object. For example, you can create a flow where one action generates a user object and another action sends a notification to the user specified in the object.

Sample action configured to use an object data pill



When you configure the child elements of an object, you must manually provide data pills for each child element of the object. For example, you can manually configure the user object with record values from an earlier action.

Script support

Create and reference complex data from a script. Use a script when your source data comes from a data stream, a REST step response, or a Look Up Records step. See Script support for complex data for more information about scripting with complex data.

Update set support

Update sets include complex objects as part of the flow, subflow, or action where they are defined. Any change to the parent flow, subflow, or action automatically captures its associated complex data.

General guidelines

Follow these general guidelines to create reusable and maintainable data structures.

Minimize the number of child levels in the hierarchy

The more child levels a data structure has, the more difficult it is to view and select a data variable from the hierarchy. While you can build data structures with any number of child levels, it becomes difficult to navigate and understand data structures with more than seven child levels. For the best user experience, avoid creating data structures that have so many child levels you must scroll horizontally to see and populate them.

Create a separate object for each type of record data

Most Workflow Studio data is record data whether it is from an instance or an external system. This design method ensures that you know what the object contains and where the data came from.

Recreate record data structures

When building objects that receive or transmit record data, review the database dictionary entries for these records and create matching object data structures. For example, suppose that you want an object to contain data from Incident and Configuration Item tables. You might create a string element for the Short description field in the Incident table, and an array of strings element for the Class field in the Configuration Item table.

Create objects to combine different types of records

If you need information from multiple types of records, create an object that contains all the information you need. You can then use the object to format or parse data in Workflow Studio.

Create data structure

Organize multiple data variables into a structure to process them as a unit and identify the individual items within it.

Before you begin

- Role required: action_designer, flow_designer, or admin
- Set up an application in Guided Application Creator to store Workflow Studio content.
- Create an action in Workflow Studio or Create a subflow in Workflow Studio

About this task

A data structure is a collection of related data elements organized into a hierarchy. Each element in a data structure has its own data type and its own unique position in the hierarchy.

Procedure

1. Create a data variable.

2. Set **Type** to **Object**.
   The top level of a data structure hierarchy must be an Object variable.

3. Click the **Add Child Item** icon.
   Workflow Studio adds a child data variable to the bottom of the object list.

   Note:
   You can insert a child item variable between existing variables by hovering your mouse pointer between two variables, and click the insert item icon ( ) that appears.

   Note:
   When hovering your mouse pointer between a child and sibling variable, you will see a add child icon ( ). Click the left side of the icon to add a new sibling variable to the child's parent, or the right side to add another child variable under the current variable.

4. Set the child variable **Label** and **Type**.
   To add another branch to the data structure hierarchy, set the Type to Object.
5. Repeat steps 3-4 for each data variable in the hierarchy.

What to do next

Use the data structure to populate action, step, or subflow inputs. If you can reuse the data structure, save it as a template.

Save data structure

Save the data structure of child variables within an Object variable for later reuse.

Before you begin

- Role required: action_designer, flow_designer, or admin
- Set up an application in Guided Application Creator to store Workflow Studio content.
- Create an action in Workflow Studio or Create a subflow in Workflow Studio

Procedure

1. Create data structure.
2. Expand the Advanced options for the Object variable you want to save.
3. Click Save as Template.
   Workflow Studio displays a pop-up dialog.
4. Enter the template name.
5. Click **Save**.
   If the template name already exists, Workflow Studio displays a confirmation dialog to overwrite the existing template.

What to do next

Load the data structure in another action or subflow. Make updates to the data structure and save them.

Load data structure

Load a data structure of child variables within an Object variable.

Before you begin

- Role required: action_designer, flow_designer, or admin
- Set up an application in Guided Application Creator to store Workflow Studio content.
- Create an action in Workflow Studio or Create a subflow in Workflow Studio

Procedure

1. Create a data variable.
2. From **Type**, select **Object**.
3. Expand the Advanced options for the Object variable whose data structure you want to replace.
4. From **Structure**, select **Start from Template**.
   Workflow Studio displays the **Template** field.
5. From **Template**, select the template containing the template you want to load.
   If the Object variable has no existing data structure, Workflow Studio loads the data structure into it. If the Object variable has an existing data structure, Workflow Studio displays a confirmation dialog to replace the existing structure.

Create a custom action to generate an object from a record

Generate an object from a User record. Learn how to use an Action output to create an object from record values.

Before you begin

Role required: admin

About this task

Use this example to see demonstrations of these operations and steps.

- Create action inputs for the User record fields First name, Last name, and Email.
- Lookup a User record matching the action input values.
- Create an action output for a contact object.
- Save the contact object as a template.
- Map contact object values to User record field values.
- Test the action with a sample user.

Procedure

1. Create an application to store your work.
   Use the Guided Application Creator.

   Example
   For example, create `My Application`.
2. Navigate to **Process Automation** > **Flow Designer**.
   The system displays the Workflow Studio landing page.
3. Select **New** > **Action**
   The system displays the Action Properties dialog.
4. Enter these sample values.

Field	Value
Name	Create Contact Object From User
Application	My Application
Accessible From	All application scopes

5. Select **Submit**.
   The system displays the Workflow Studio interface.
6. From the Action Outline, select **Inputs** > **Create Input**
   The system displays a new action input.
7. Configure the action input with these values.

Field	Value
Label	First name
Type	String
Mandatory	True

8. From the Action Outline, select **Inputs** > **Create Input**
   The system displays a new action input.
9. Configure the action input with these values.

Field	Value
Label	Last name
Type	String
Mandatory	True

10. From the Action Outline, select **Inputs** > **Create Input**
    The system displays a new action input.
11. Configure the action input with these values.

Field	Value
Label	Email address
Type	String
Mandatory	False

12. From Action Outline, select **Outputs** > **Create Output**.
    The system displays a new action ouput.
13. Configure the output variable with these values.

Label	Name	Type	Mandatory
contact	contact	Object	False

14. From the row for the contact Object, select **Add Child Item**.
15. Configure the child item with these values.

Label	Name	Type	Mandatory
First name	first_name	String	True

16. From the row for the contact Object, select **Add Child Item**.
17. Configure the child item with these values.

Label	Name	Type	Mandatory
Last name	last_name	String	True

18. From the row for the contact Object, select **Add Child Item**.
19. Configure the child item with these values.

Label	Name	Type	Mandatory
Email address	email_address	String	False

20. From the row for the contact Object, select **Toggle Advanced Inputs**.
21. From the Advanced Options, select **Save As Template**.
    The system displays the Save As Template dialog.
22. For **Enter a Name**, enter `contact`.
23. Click **Save**.
24. Select **Exit Edit Mode**.
    The System displays the output fields you created.
25. Configure the outputs with these values.

Label	Value
First name	**[step->Look Up Record step->Record->First name]**
Last name	**[step->Look Up Record step->Record->Last name]**
Email Address	**[step->Look Up Record step->Record->Email]**

Note:
You can select data pills from the data panel or from the Data Pill Picker button.

26. Select **Save**.
27. Select **Test**.
    The system displays the Test Action dialog.
28. Enter these test values.

Input	Value
First name	Abel
Last name	Tuter

29. Select **Run Test**.
    The system runs the action with the test values provided.
30. Select **Action has executed. To view the action, click here.**
    The system displays the action execution details.
31. Review the runtime value for the action Output data.
    The system displays output data in JSON format.

Example
For example, sample contact object JSON for the user Abel Tuter.

```
{
    "contact": {
        "email_address": "abel.tuter@example.com",
        "last_name": "Tuter",
        "first_name": "Abel"
    }
}
```

Result
You have a custom action that looks up a User record and converts it into a contact object.

What to do next
Customize the action to use your own logic.

Create a custom flow to generate an object for each record in a list

Generate an object for each User record in a list. Learn how to use flow logic to iterate through a list of records.

Before you begin
- Role required: admin
- Create a custom action to generate an object from a record

About this task
Use this example to see demonstrations of these operations and steps.
- Create a flow that runs on a daily schedule.
- Look up User records filtered by the Department provided as an input.
- Add flow logic that runs for each User record you looked up previously.
- Create a contact object for each User record using the custom action you created previously.
- Create a log message for each User record.

Procedure
1. Navigate to **All** > **Process Automation** > **Flow Designer**.
   The system displays the Workflow Studio landing page.
2. Select **New** > **Flow**
   The system displays the Flow Properties dialog.
3. Enter these sample values.

Field	Value
Name	Create Contact Objects From Users

Field	Value
Application	My Application
Run As	User who initiates session

4. Select **Submit**.
   The system displays the Workflow Studio interface.
5. Select **Click to add a Trigger** > **Date** > **Daily**.
6. Select **Done** to close the trigger.
7. Select **Click to add an Action, Flow Logic, or Subflow** > **Action** > **ServiceNow Core** > **Look Up Records**.
   The system adds the action to the flow.
8. For **Table**, select **User [sys_user]**.
9. For **Conditions**, add these values.
   - **[Department][is][Development] [AND]**
   - **[Email][is not empty]**
10. Configure these field values.

Field	Value
Order by	Name
Sort	a to z
Max Results	1000

11. Select **Done** to close the action.
12. Select **Click to add an Action, Flow Logic, or Subflow** > **Flow Logic** > **For Each**.
    The system adds the flow logic to the flow.
13. For **Items**, select **[1->User Records]**.

    Note:
    You can select the Action 1 **User Records** data pill from the data panel or from the Data Pill Picker button.

14. Select **Done** to close the flow logic.
15. Select the plus icon to add a child item to the For Each flow logic.
16. Select **Action** > **My Application** > **Create Contact Object**.
17. For **userRecord [User]**, select **[2->User Record]**.

    Note:
    You can select the Action 2 **User Record** data pill from the data panel or from the Data Pill Picker button.

18. Select **Done** to close the flow logic action.
19. Select the plus icon to add a child item to the For Each flow logic.
20. Select **Action** > **ServiceNow Core** > **Log**.
21. For **Message**, select **[2.1->contact]**.

Note:
You can select the Action 2.1 **contact** data pill from the data panel or from the Data Pill Picker button.

22. Select **Done** to close the flow logic action.
23. Select **Save**.

Script support for complex data

Create and reference complex data from a script. Use a script when your source data comes from a data stream, a REST step response, or a Look Up Records step.

Use script to create complex data when data comes from these sources.

Data sources requiring script

Data source	Create/map complex data from
Data Stream action response stream	Script Parser step
REST step response	Script step
Look Up Records step	

Data Stream action response stream

Data Stream actions use a parser script to map stream item values to complex object values. When writing a parser script, use JavaScript methods appropriate to the data stream format. For example, use the JSON - Scoped class to parse or encode a JSON data stream.

Parser scripts have access to the data stream input and output objects as well as a targetObject property. See Data Stream actions for more information about parsing a response stream to create complex data.

REST step response

You can convert a REST step response into one or more complex objects by parsing it with a Script step. To access a response from a Script step, you must create an input script variable and map it to the response payload from the prior REST step. See Script step for more information about creating script input variables.

Write a script that maps REST response values to complex object values. When writing REST response script, use JavaScript methods appropriate to the response format such as the JSON parse() method.

Note:
When you use complex data as the source of a string input, Workflow Studio automatically converts it into a JSON string.

You do not need to use a Script step to create a REST request from complex data. You can generate complex data in a prior action or step and then map it to a string input of the REST step. At run time, the action or flow converts the complex data into a JSON representation.

For example, see the script steps used in Get started with dynamic inputs for the data gathering actions. The data gathering actions for getting table and field names both use a Script step to parse a REST response into a JSON object. Both data gathering actions also create output variables that store complex data as JSON objects.

Look Up Records step

While flows can use **For each** flow logic to process a list of records, actions require a Script step. The Script step replaces the **For each** flow logic with JavaScript such as a `For` or `While` loop.

To access record data from a Script step, you must create an input script variable and map it to the record data from the prior look up step. See Script step for more information about creating script input variables.

See Create a custom action to generate an array of objects from a list of records for an example action that converts a list of user records into an array of contact objects.

Note:
The Look Up Records action does not require a Script step to convert record data into complex data. You can create a custom action to convert a record into an object and apply **For each** flow logic to the custom action. See Create a custom action to generate an object from a record and Create a custom flow to generate an object for each record in a list for an example of creating a complex object without using script.

Dot-walking object structures

You can reference elements from the structure of an object by dot-walking the path of the structure. All complex data paths start with the name of the data source, which is either the global object for inputs, the global object for outputs, or the name of the array or object you created in script.

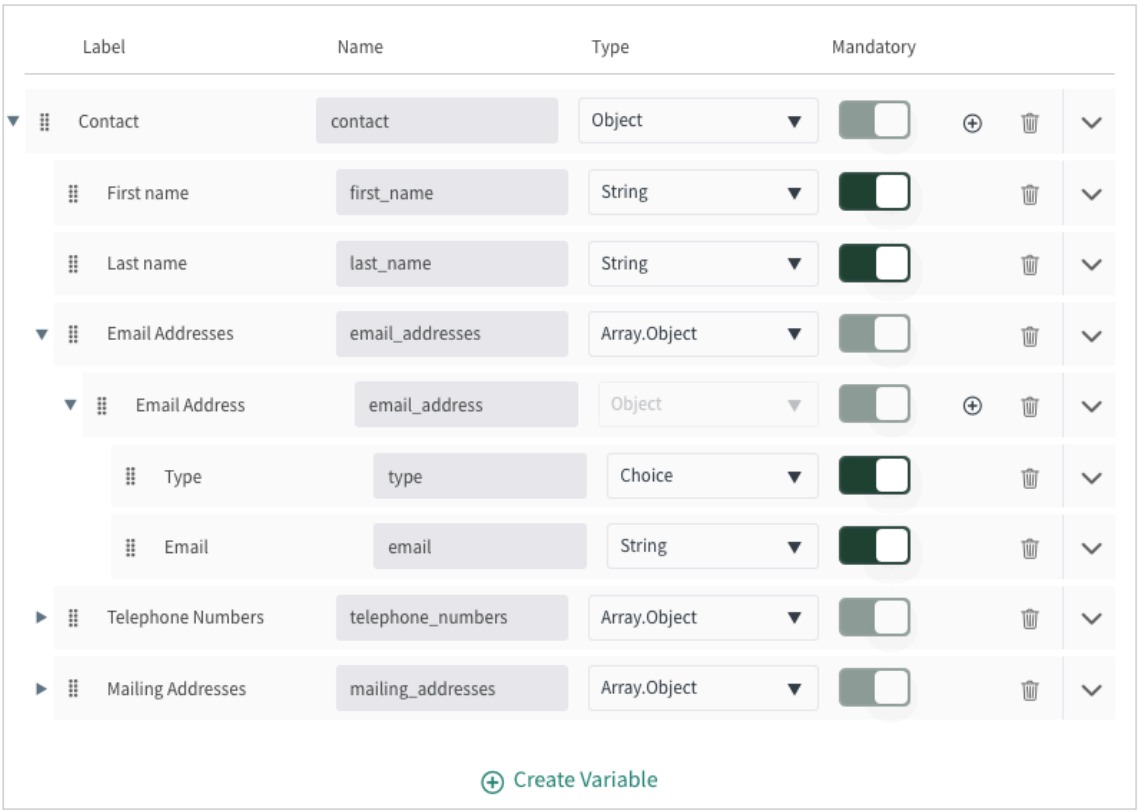
Next in the path are the names of each structural element referenced separated by period characters (also known as dots). Listing the names of structural elements is identical to dot-walking a reference field where you list the table structure to a particular reference field.

Note:
A dot-walk path always lists the name of a structural element rather than its label.

For example, suppose that you define a contact object as an Output variable. The object has the following structure.

Sample Contact object

Output Variables



Place in structure	Label	Name	Type
Parent	Contact	contact	Object
Child	First name	first_name	String
Child	Last name	last_name	String
Child	Email Addresses	email_addresses	Array.Object
Grandchild	Email Address	email_address	Object
Great grandchild	Type	type	Choice
Great grandchild	Email	email	String
Child	Telephone	telephone_number	Array.Object
Child	Mailing Addresses	mailing_address	Array.Object

The dot-walk path to the **First name** structural element would be `outputs.contact.first_name` while the path to the **Email** structural element would be `outputs.contact.email_addresses[0].email` since you must specify an individual element of the array by its JavaScript index value.

Note:
A dot-walk path omits the name of the repeated element within the array. For example, an array of objects does not have to specify the object element name. However, since objects are containers for other elements, you can specify a child element of the object within a dot-walk path.

General guidelines

Keep these general guidelines in mind when scripting with complex data.

Use string inputs to convert complex data into a JSON string

When you map complex data to a string input, Workflow Studio automatically converts it into a JSON string. Instead of writing a script, you can add a string input to a REST step and map it to complex data from a prior action or step.

Save your objects as templates

Save your objects as templates so you can reuse them in other actions, flows, and Script steps.

Create script input variables to access prior data

Create a script input variable for any data you want to access from the action input or a prior step. Map the script input variable to the input or step data pill. For example, map the script input variable to a list of user records you looked up in a prior step.

Create a script output variable to store complex data

Create a script output variable to store any complex data your script creates. The script output variables must match the values defined in the script. For example, create a contacts array of objects to store multiple contact objects. Save the contact object as a template so you can reuse it.

Map the action output to the script output variable

When you want a custom action to output complex data, add an action output and map it to the data pill for your Script step output variable. For example, create a contacts array and load the contact object template you saved earlier. Map the action output to the contacts array produced by your Script step.

Create a custom action to generate an array of objects from a list of records

Generate an array of objects from a list of User records. Learn how to use a Script step to iterate through a list of records.

Before you begin

Role required: admin of flow_designer

About this task

Use this example to see demonstrations of these operations and steps.

- Create an action input for a Department record.
- Look up a maximum of three User records for the Department action input.
- Configure a Script step to process a list of User records.
- Create a script input variable containing the list of User records.
- Write script that creates an empty contacts array.
- Write script that iterates through the list of User records.
- Write script that creates a contact object and maps User record field values to the contact object.
- Write script that populates the contacts array with the current contact object.
- Create script output variables for the contacts array and child contact object.
- Save the contact object as a template.

- Output the generated contacts array of objects as a data pill.
- Test the action with a sample department.

Procedure

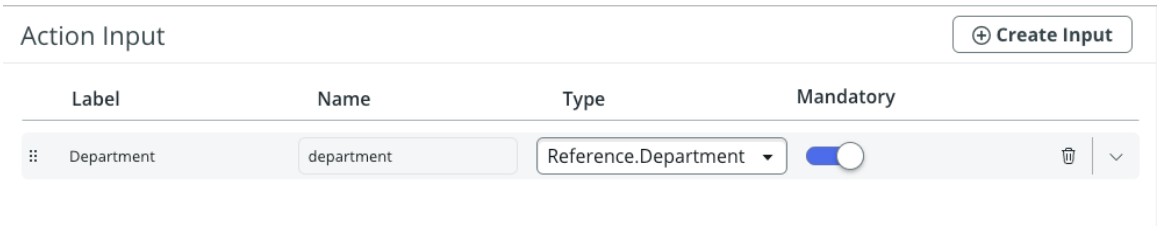
1. **Optional:** Create an application to store your work.
   You can use App Engine Studio to plan, create, and deploy applications. For more information about building a custom application, see Building apps in App Engine Studio.

   Example
   For example, create an application called `My Application`.
2. Navigate to **All** > **Process Automation** > **Workflow Studio**.
3. On the homepage, select **Actions**.
4. Select **New** > **Action**
   The system displays the Action Properties dialog.
5. Enter these sample values.

Field	Value
Name	Create Contacts Array Of Objects
Application	Global
Accessible From	All application scopes

> **Note:**
> If you created an application to store and deploy your custom action, use that application instead of global.

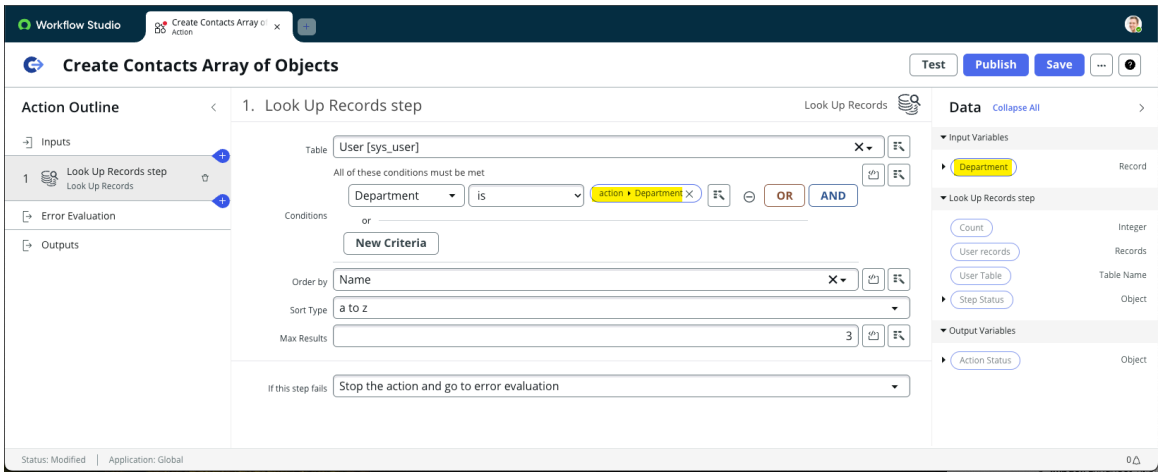
6. Select **Build action**.
   The system displays the Workflow Studio interface.
7. From the Action Outline, select **Inputs** > **Create Input**
   The system displays a new action input.
8. Configure the action input with these values.



Field	Value
Label	Department
Type	Reference.Department [Reference.cmn_department]
Mandatory	True

9. From the Action Outline, select **Add a new step**.
   The system displays a list of available steps.

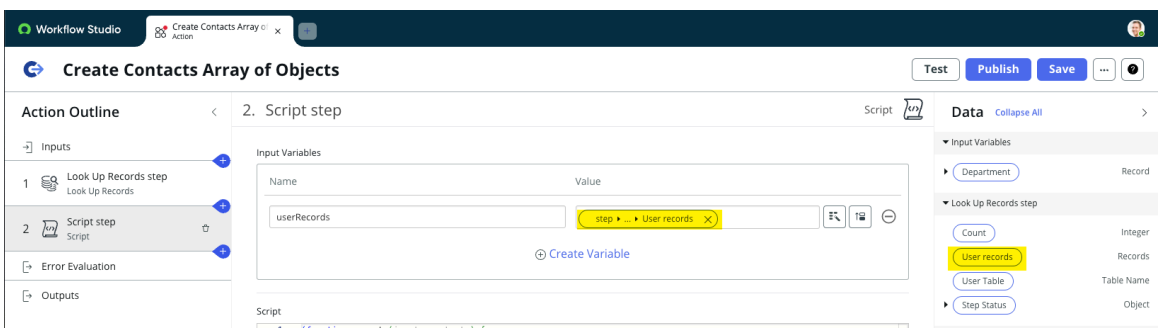
10. Select **Look Up Records**
11. Configure the step with these values.



Field	Value
Table	User [sys_user]
Conditions	[Department][is][action->Department] **Note:** Select the Department data pill from the Input Variables.
Order by	Name
Sort Type	a to z
Max Results	3

Note:
This example limits the **Max Results** setting to three records just for demonstration purposes.

12. From the Action Outline, select **Add a new step**.
    The system displays a list of available steps.
13. Select **Script**.
14. From the Input Variables section, select **Create Variable**.
15. Configure the input variable with these values.



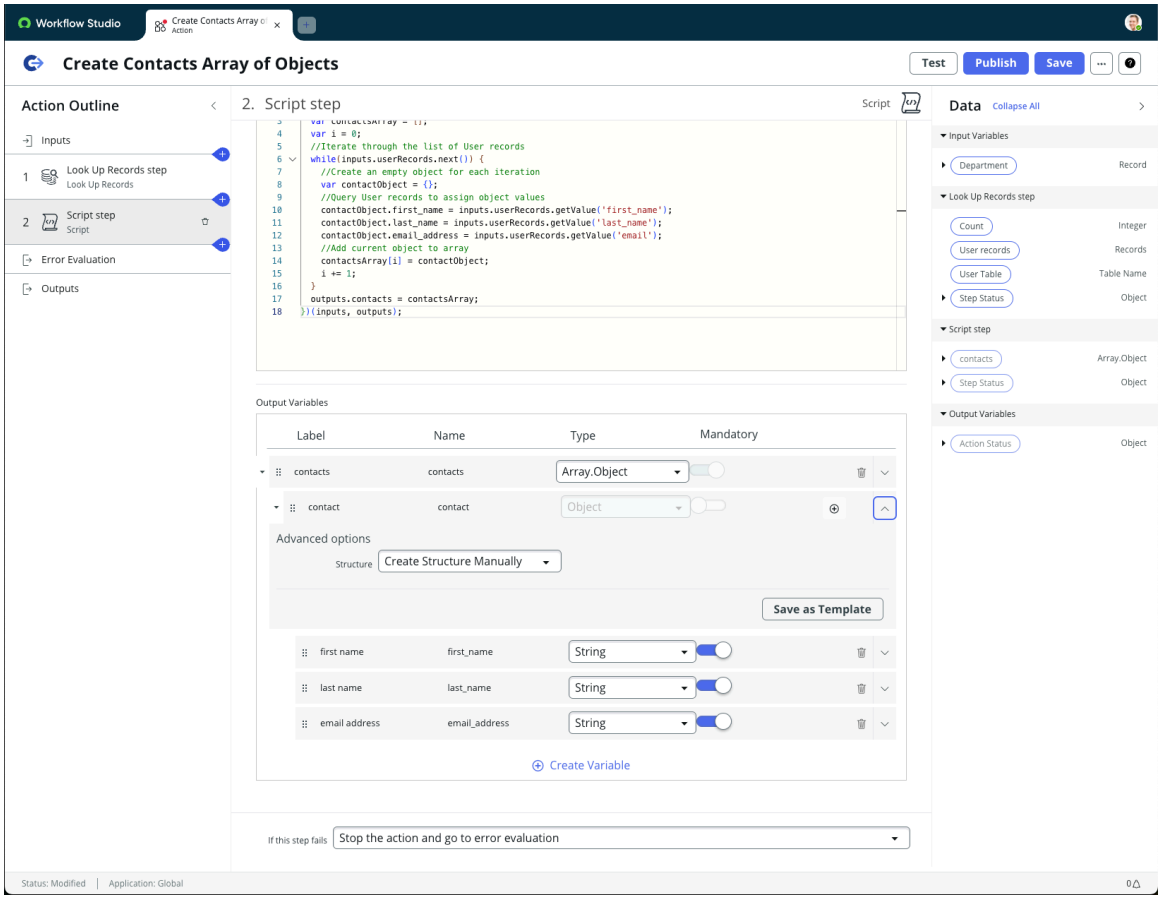
Field	Value
Name	userRecords
Value	[step->Look Up Records step->User Records] **Note:** Select the User records data pill from the Look Up Records step.

Note:
You can select the **User records** data pill from the data panel or from the Data Pill Picker button.

16. For **Script**, enter the following text.

```
(function execute(inputs, outputs) {
  //Create an empty array
  var contactsArray = [];
  var i = 0;
  //Iterate through the list of User records
  while(inputs.userRecords.next()) {
    //Create an empty object for each iteration
    var contactObject = {};
    //Query User records to assign object values
    contactObject.first_name =
 inputs.userRecords.getValue('first_name');
    contactObject.last_name =
 inputs.userRecords.getValue('last_name');
    contactObject.email_address =
 inputs.userRecords.getValue('email');
    //Add current object to array
    contactsArray[i] = contactObject;
    i += 1;
  }
  outputs.contacts = contactsArray;
})(inputs, outputs);
```

17. From Output Variables, select **Create Variable**.
18. Configure the output variable with these values.



Label	Name	Type	Mandatory
contacts	contacts	Array.Object	True

19. Expand the contacts Array.Object, and rename the child object to `contact`.
20. From the row for the contact object, select the **Add Child Item** icon ⊕.
21. Configure the child item with these values.

Label	Name	Type	Mandatory
first name	first_name	String	True

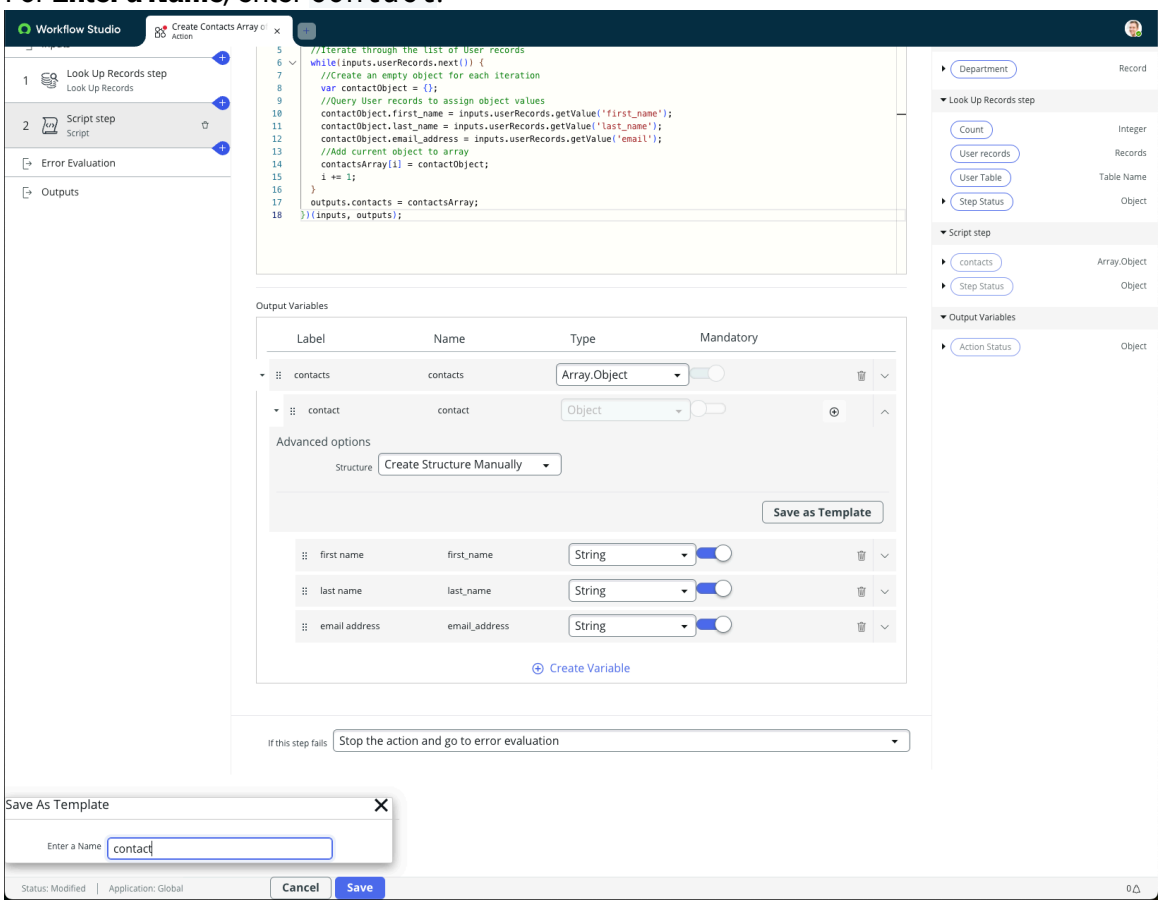
22. From the row for the contact object, select the **Add Child Item** icon ⊕.
23. Configure the child item with these values.

Label	Name	Type	Mandatory
last name	last_name	String	True

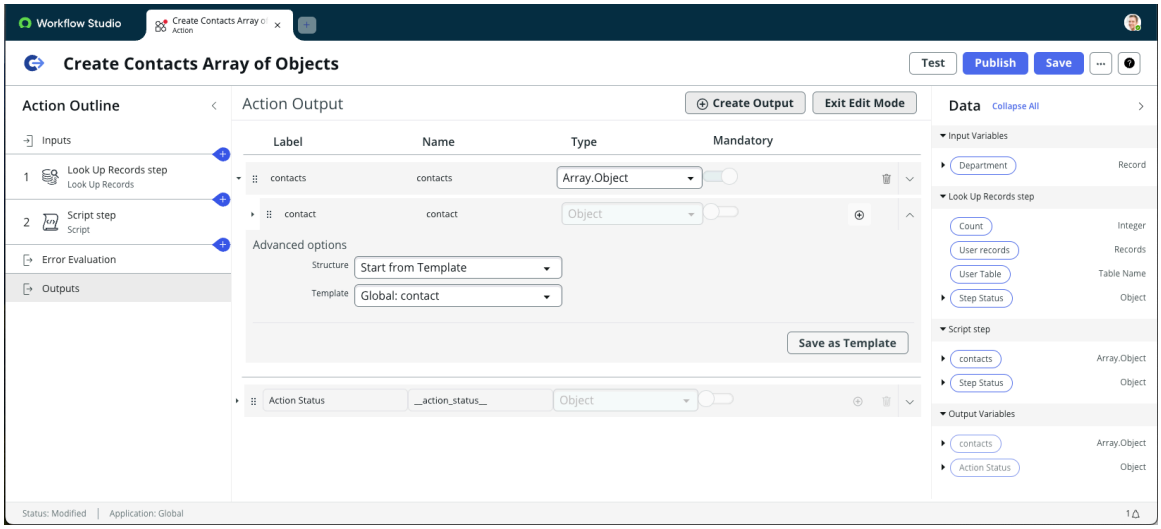
24. From the row for the contact object, select the **Add Child Item** icon ⊕.
25. Configure the child item with these values.

Label	Name	Type	Mandatory
email address	email_address	String	True

26. From the row for the contact Object, select **Toggle Advanced Inputs**.
27. From the Advanced Options, select **Save As Template**.
    The system displays the Save As Template dialog.
28. For **Enter a Name**, enter `contact`.



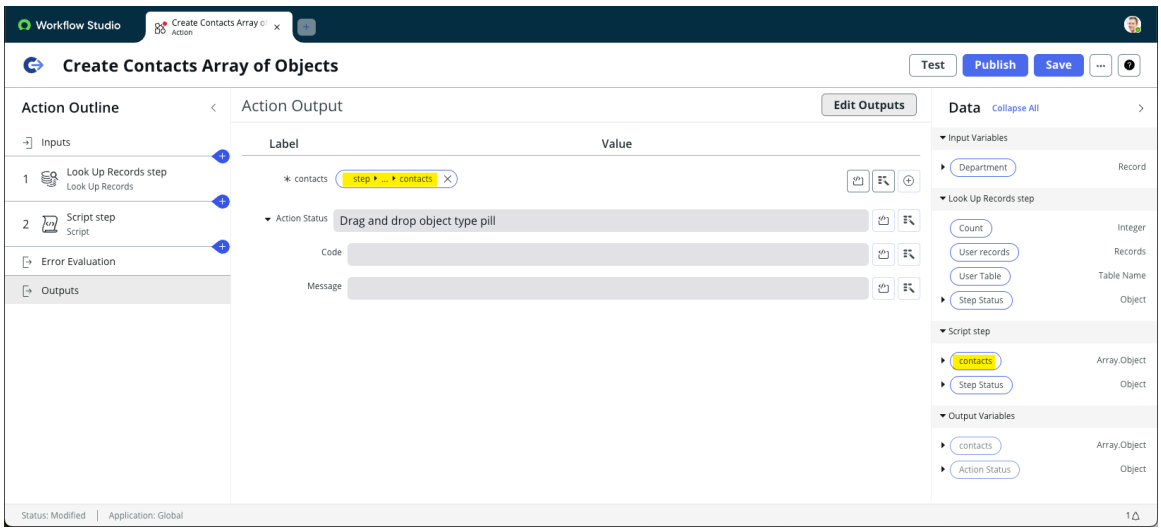
29. Click **Save**.
30. From the Action Outline, select **Outputs** > **Create Output**.
31. Configure the Action Output with these values.



Label	Name	Type	Mandatory
contacts	contacts	Array.Object	True

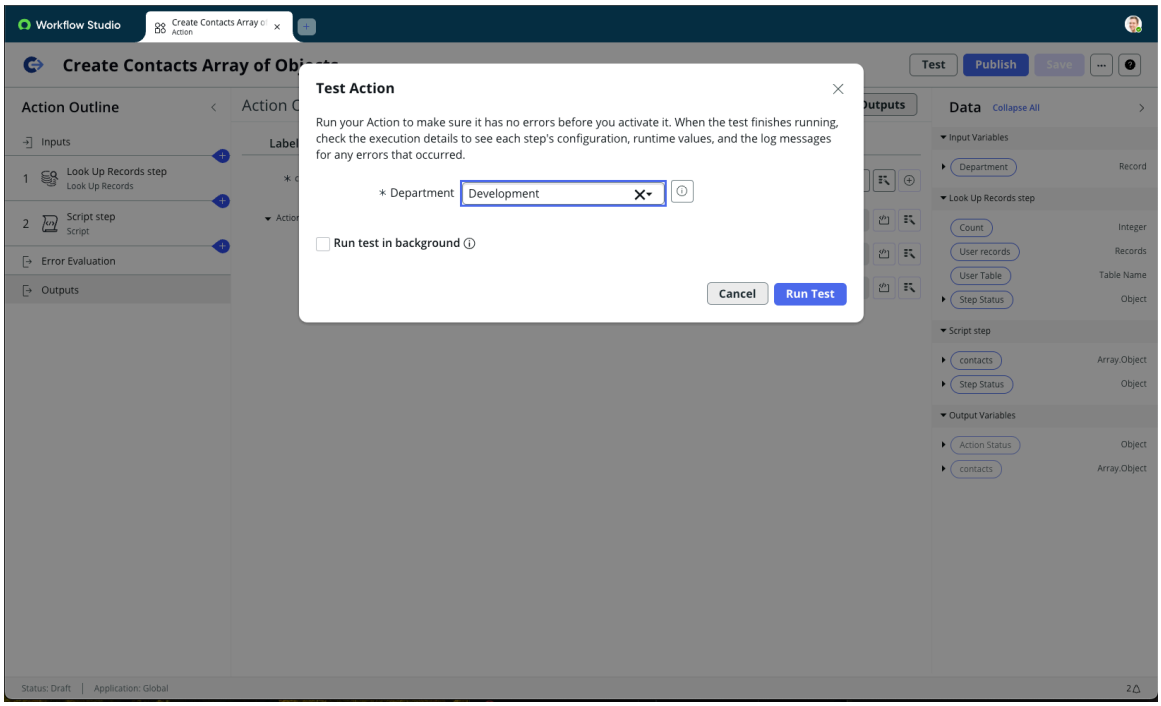
32. Expand the contacts Array.Object.
33. From the row for the contact Object, select **Toggle Advanced Inputs**.
34. From the Advanced Options, select **Structure** > **Start from Template**.
    The system displays **Template**.
35. For **Template**, select the template you previously saved.

    Example
    For example, select **Global: contact**.
36. Select **Exit Edit Mode**.
    The System displays the output fields you created.
37. For **contacts**, select **[step->Script step->contacts]**.



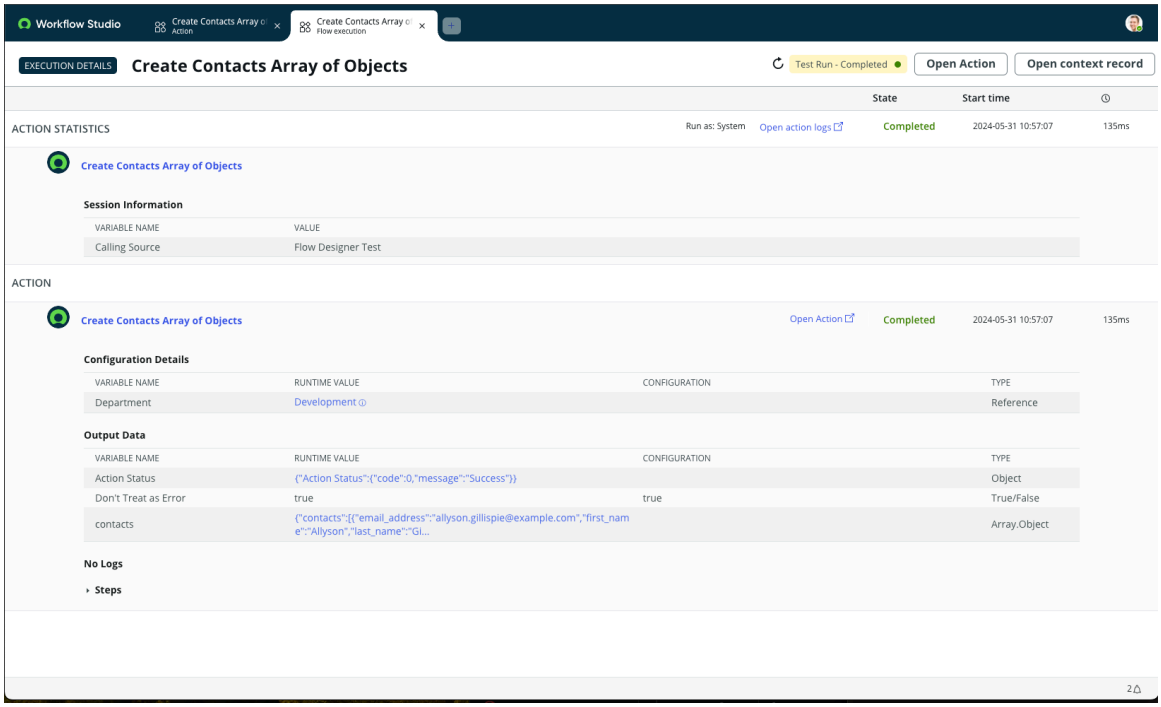
Note:
You can select the Script step **contacts** data pill from the data panel or from the Data Pill Picker button.

38. Click **Save**.
39. Select **Test**.
    The system displays the Test Action dialog.
40. Enter the following test value:



Input	Value
Department	Development

41. Select **Run Test**.
    The system runs the action with the test values provided.
42. Select **Your test has finished running. View the Action execution details.**
    The system displays the action execution details.
43. Review the runtime value for the action Output data.



Although the execution details display the output data as a JSON formatted string, the actual output data type is an array of objects. If you need a string version of your output, you can convert the object into a string using the JSON class. For more information about converting a JSON object into a string, see Scoped JSON - stringify(Object jsonObject).

Example

For this example, the contacts object contains an array of contact objects with first name, last name, and email information for three users in the Development department.

```
{
    "contacts":
        "contact": [
            {
                "email_address":
 "allyson.gillispie@example.com",
                "first_name": "Allyson",
                "last_name": "Gillispie"
            },
            {
                "email_address": "alva.pennigton@example.com",
                "first_name": "Alva",
                "last_name": "Pennigton"
            },
            {
                "email_address": "andrew.och@example.com",
                "first_name": "Andrew",
                "last_name": "Och"
            }
        ]
    }
}
```

Result

You have a custom action that looks up the Users for a given department and converts those users into an array of contact objects.

What to do next
Customize this action to use your own logic.

Create a custom action to generate an array of strings from a list of records

Generate an array of strings from a list of User Role records. Learn how to use a Script step to iterate through a list of records.

Before you begin
Role required: admin of flow_designer

About this task
Use this example to see demonstrations of these operations and steps.

- Create an action input for a Role record.
- Look up a maximum of three User Role records that have the Role action input.
- Configure a Script step to process a list of User Role records.
- Create a script input variable containing the list of User Role records.
- Write script that creates an empty users array.
- Write script that iterates through the list of User Role records.
- Write script that populates the users array with the current user field value.
- Create script output variables for the users array and child user string.
- Output the generated users array of strings as a data pill.
- Test the action with three sample users.

Procedure

1. **Optional:** Create an application to store your work.
   You can use App Engine Studio to plan, create, and deploy applications. For more information about building a custom application, see Building apps in App Engine Studio.

   Example
   For example, create an application called `My Application`.
2. Navigate to **All** > **Process Automation** > **Workflow Studio**.
3. On the homepage, select **Actions**.
4. Select **New** > **Action**
   The system displays the Action Properties dialog.
5. Enter these sample values.

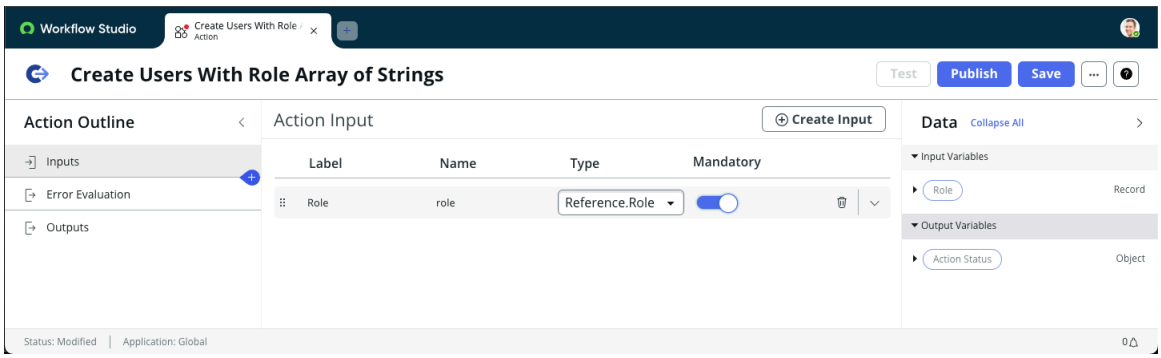
Field	Value
Name	Create Users With Role Array Of Strings
Application	Global
Accessible From	All application scopes

Note:
If you created an application to store and deploy your custom action, use that application instead of global.

6. Select **Build action**.

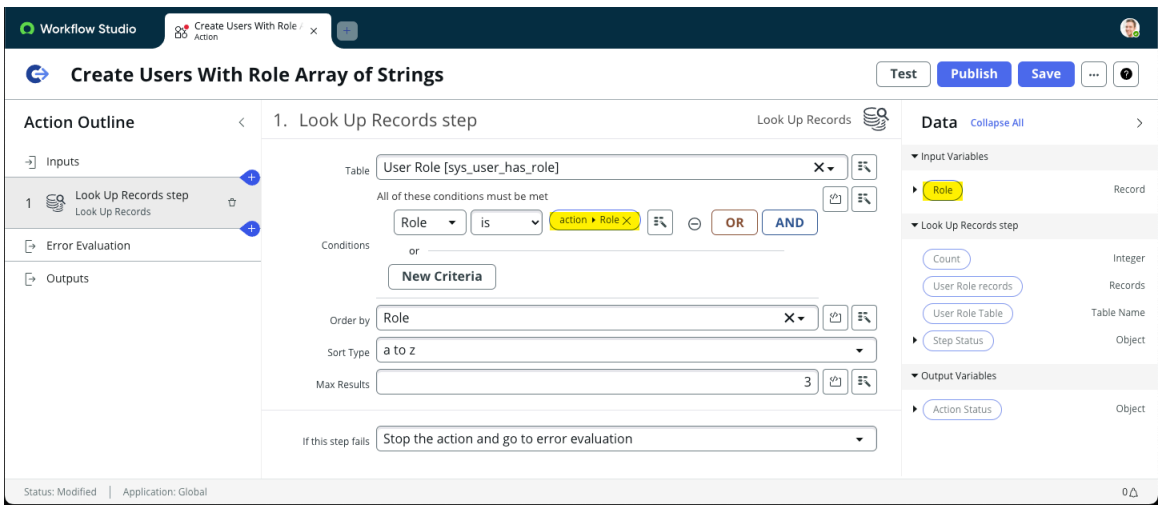
The system displays the Workflow Studio interface.

7. From the Action Outline, select **Inputs** > **Create Input**
   The system displays a new action input.
8. Configure the action input with these values.



Field	Value
Label	Role
Type	Reference.Role [Reference.sys_user_role]
Mandatory	True

9. From the Action Outline, select **Add a new step**.
   The system displays a list of available steps.
10. Select **Look Up Records**
11. Configure the step with these values.

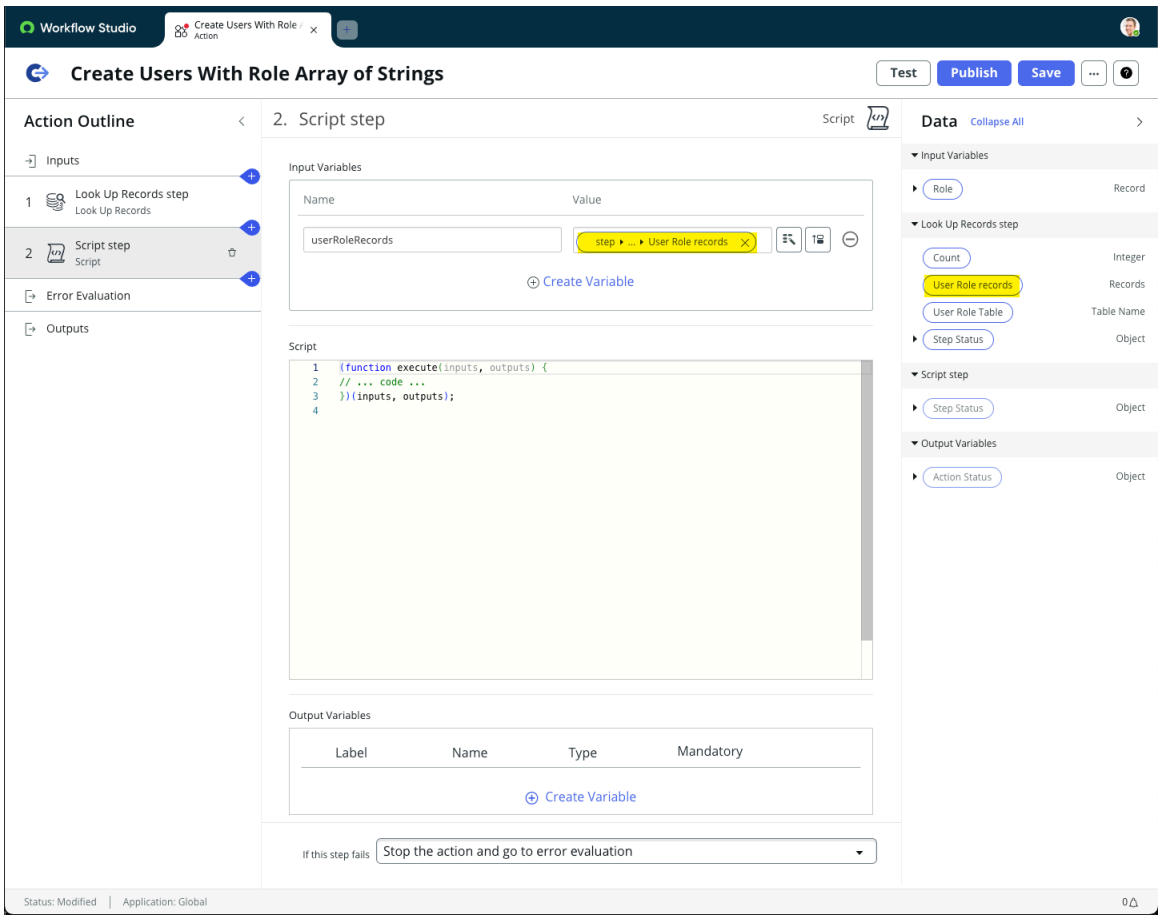


Field	Value
Table	User Role [sys_user_has_role]
Conditions	[Role][is][action->Role] **Note:** Select the Role data pill from the Input Variables.

Field	Value
Order by	Role
Sort Type	a to z
Max Results	3

Note:
This example limits the **Max Results** setting to three records just for demonstration purposes.

12. From the Action Outline, select **Add a new step**.
The system displays a list of available steps.
13. Select **Script**.
14. From the Input Variables section, select **Create Variable**.
15. Configure the input variable with these values.



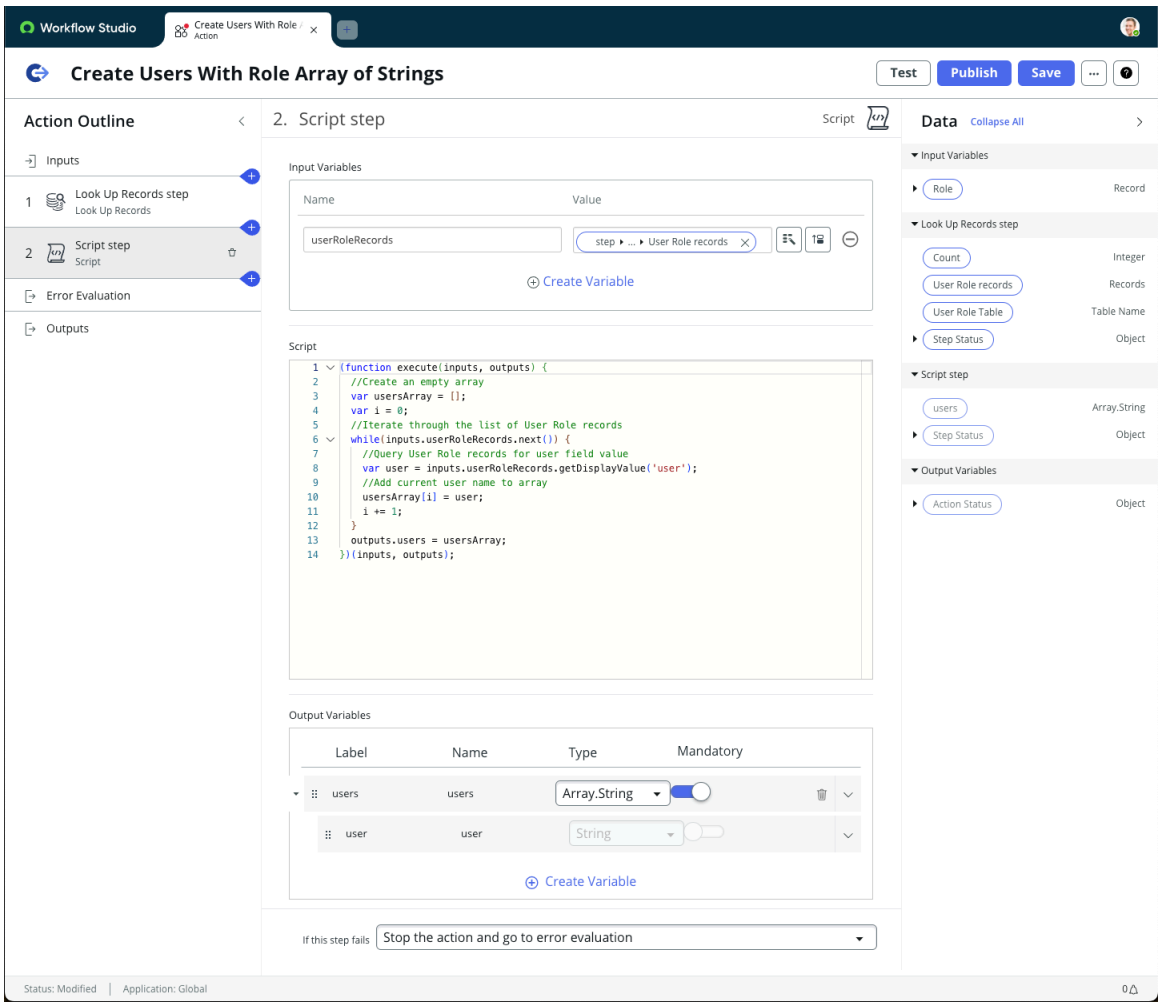
Field	Value
Name	userRoleRecords
Value	[step->Look Up Records step->User Role records] **Note:** Select the User Role records data pill from the Look Up Records step.

Note:
You can select the **User Role records** data pill from the data panel or from the Data Pill Picker button.

16. For **Script**, enter the following text.

```
(function execute(inputs, outputs) {
  //Create an empty array
  var usersArray = [];
  var i = 0;
  //Iterate through the list of User Role records
  while(inputs.userRoleRecords.next()) {
    //Query User Role records for user field value
    var user = inputs.userRoleRecords.getDisplayValue('user');
    //Add current user name to array
    usersArray[i] = user;
    i += 1;
  }
  outputs.users = usersArray;
})(inputs, outputs);
```

17. From Output Variables, select **Create Variable**.
18. Configure the output variable with these values.

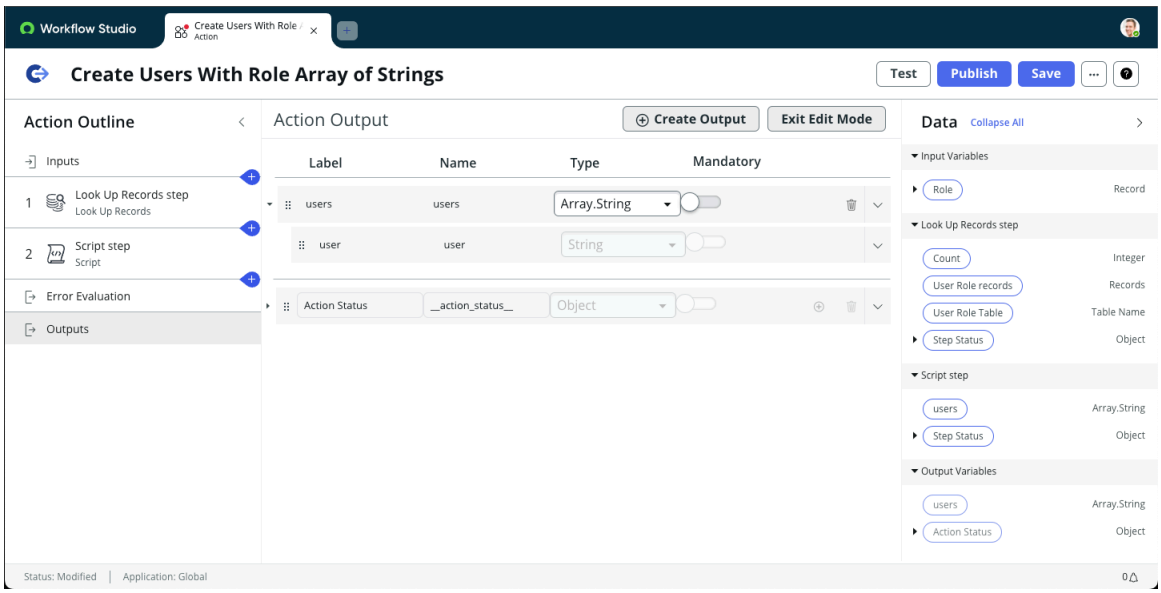


Label	Name	Type	Mandatory
users	users	Array.String	True

19. Expand the users Array.String, and rename the child string to `user`.
20. Configure the child item with these values.

Label	Name	Type	Mandatory
user	user	String	False

21. From the Action Outline, select **Outputs** > **Create Output**.
22. Configure the Action Output with these values.

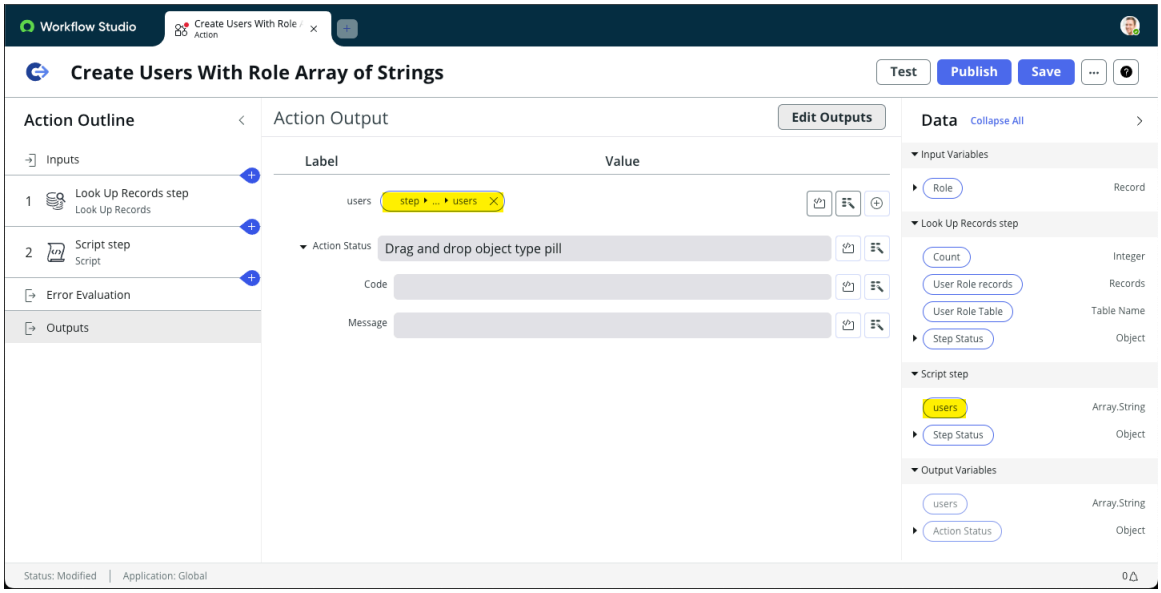


Label	Name	Type	Mandatory
users	users	Array.String	True

23. Expand the users Array.String, and rename the child string to `user`.
24. Configure the child item with these values.

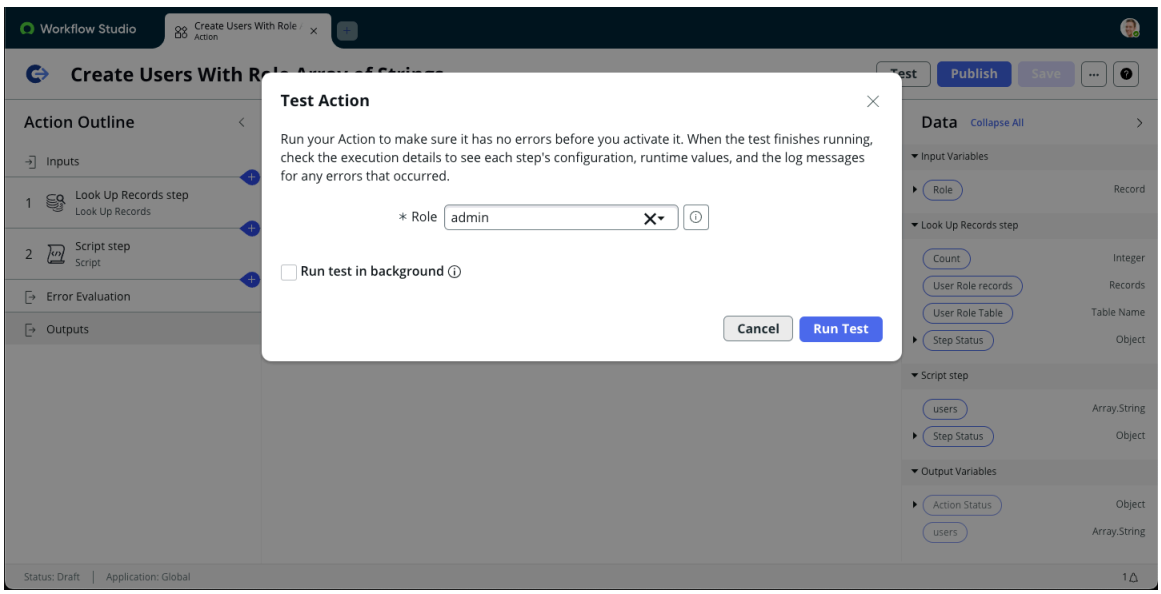
Label	Name	Type	Mandatory
user	user	String	False

25. Select **Exit Edit Mode**.
    The System displays the output fields you created.
26. For **users**, select **[step->Script step->users]**.



Note:
You can select the Script step **users** data pill from the data panel or from the Data Pill Picker button.

27. Click **Save**.
28. Select **Test**.
    The system displays the Test Action dialog.
29. Enter the following test value:

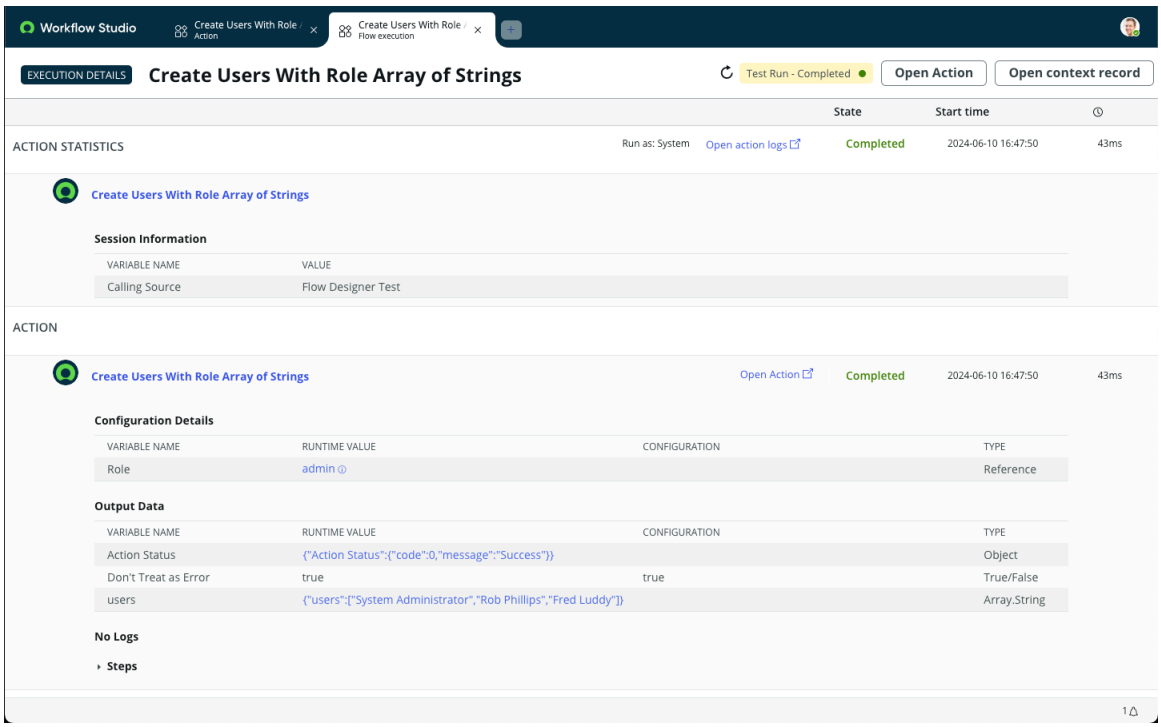


Input	Value
Role	admin

30. Select **Run Test**.
    The system runs the action with the test values provided.
31. Select **Your test has finished running. View the Action execution details.**

The system displays the action execution details.

32. Review the runtime value for the action Output data.



The system displays output data in JSON format.

Example
For this example, the users array contains three users who have the admin role.

```
{
    "users": [
        "System Administrator",
        "Rob Phillips",
        "Fred Luddy"
    ]
}
```

Result
You have a custom action that looks up the Users who have a given role and converts those users into an array of user name strings.

What to do next
Customize this action to use your own logic.

Building playbooks

Fully automate your cross-enterprise business processes by linking flows, actions, and subflows together in a playbook.

A playbook consists of a trigger and activities, and activities are organized into stages.

Triggers

A trigger is a record operation that tells your playbook to start a run. Each trigger has a type and conditions that must be met. Triggers only fire for record operations that are interactive or made by users. For more information, see Triggers.

Stages

Group activities by the stages of your business process, and sequence activities in an order that makes sense for your cross-enterprise workflow. For more information on stages, see Stages and activities.

Activities

An activity represents one step in your business process, and the record for an activity is called an activity definition. For more information on activity instances and activity definitions, see Overview and activity definitions.

Getting started with Playbooks

Learn the basics of designing an automated process for your organization. Get an overview of how Workflow Studio Playbooks work in the ServiceNow AI Platform®.

Playbooks

Playbooks in Workflow Studio are ServiceNow AI Platform representations of your manual cross-enterprise workflows. By creating a Workflow Studio playbook on the platform, you're digitizing these workflows. A playbook in Workflow Studio has two dimensions: the playbook and their associated process executions.

Playbooks

A playbook is where a playbook owner configures and organizes multiple instances of Workflow Studio content into a coherent business process. A playbook consists of a trigger and a sequence of stages, which are made up of a sequence of activities.

Process executions

A process execution is a single, runtime instance of a playbook.

Building a playbook

When you design your business process in Playbooks, you're creating a playbook.

Each playbook consists of a trigger as well as stages and activities. Triggers define the conditions that, when met, start running your playbook. Stages and activities represent stages and individual steps in your overall business process. In the Playbooks design environment, you can organize these stages and Playbooks into a sequence that reflects how your business process runs.

When you're done creating your playbook, activate it so that it runs when triggered.

What happens when a playbook runs

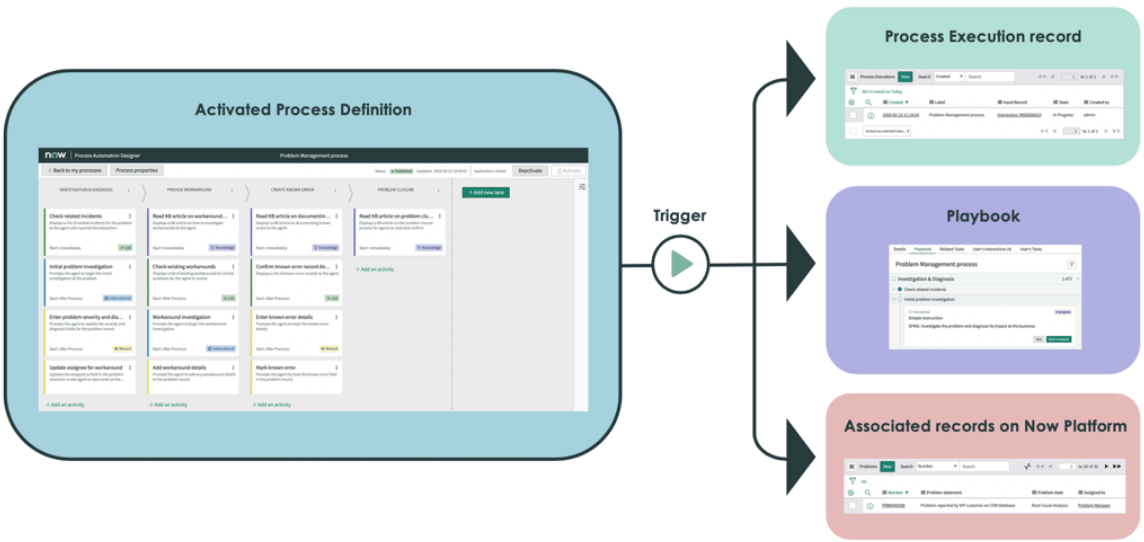
When your activated playbook is triggered, the system creates a process execution, which represents a single, running instance of your playbook.

When your playbook starts running, a process execution is created. The system runs each activity in your playbook until the process execution is complete. As each activity runs, automated record operations occur on the ServiceNow AI Platform.

If an administrator has set up Playbook Experience for your playbook, then agents and fulfillers can run the playbook. Agent or fulfillers go through each activity in the playbook, which is associated with a record in a Workspace.

In summary, your playbook runs when:

1. The system creates a process execution.
2. Playbook Experience is configured and the playbook is rendered for agents.
3. Automated operations on records that are associated with your process execution occur on the ServiceNow AI Platform.



Learn more about Playbooks

If you're ready to start digitizing your business process with Playbooks in Workflow Studio, try checking out these resources:

- Design your first automated process
- Create a playbook

Design an automated process

Transform an example manual business process into a well-designed, automated process that runs on the ServiceNow AI Platform®.

Before you begin

- Enable the **Process Automation Designer for App Engine [com.glide.pad.license]** plugin with a subscription to the ServiceNow AI Platform App Engine. For more information, see Activate playbooks.
- Follow the steps to Configure a Playbook user experience.
- Ensure that your current application is set to **Global**. For more information, see Application picker.
- Role required: admin or playbook.admin

About this task

In the following procedure, you can step through an example of how to digitize a manual business process on the ServiceNow AI Platform using Playbooks in Workflow Studio. In this example, you can use Playbooks to standardize and automate how Service Desk agents handle chat interactions with VIP users.

The manual business process for this example consists of the following stages:

1. *`Identify and Log`*: A Service Desk agent learns of an issue that a VIP user is facing while chatting with the user in a messaging application. The Service Desk agent creates an interaction record to track this issue.
2. *`Classify and Diagnose`*: The Service Desk agent associates an incident record with the interaction and sets the incident's priority to High. The agent then enters the Assigned To user for the incident.
3. *`Communicate Work in Progress`*: The assignee updates the incident record's state to Work in Progress and emails the VIP user when progress is made on the incident.
4. *`Resolve`*: When the incident is resolved, the assignee emails the resolution information to the VIP user.

Procedure

1. Create a playbook named *`Handle Interactions with VIPs`*.

   a. Navigate to **Process Automation** > **Workflow Studio** > **Playbooks**.

   b. In the main header, click **Create a new process**.

   c. On the form, fill in the fields.

Field	Action
Label	Enter `Handle Interactions with VIPs`.

Field	Action
Description	Enter `This process defines how Service Desk agents can handle interaction records that are created for VIP users.`
Application	Select **Global**.

d. Click **Select a trigger**.

e. Click the **Define your own trigger conditions for when your process runs** option.

f. From the list of trigger options, select **Record Create**.

g. Click **Set your trigger conditions**.

h. In the Table list, select **Interaction [interaction]**.

i. Use the condition builder to add the following condition to your trigger: **[Opened for->VIP] [is] [True]**.

j. Click **Go to Designer**.

The Playbooks design environment appears.

2. Add a stage for each stage in your process.

a. Click **+ Add stage** to add the first stage to your process.

b. In the stage properties panel's **Label** field, enter `Classify and Diagnose`.

> **Note:**
> Because the *`Identify and Log`* stage triggers this process, don't add it as a stage.

c. In the **Description** field, enter `Associate an incident with the interaction`.

d. In the **When to start** field, leave **Immediately** selected, and then click **Save**.

e. Click **+ Add stage** to add another stage to your process.

f. In the stage side panel's **Label** field, enter `Communicate Work in Progress`.

g. In the **Description** field, enter `Notify VIP of work in progress`.

h. In the **When to start** field, leave **After Previous** selected, and then click **Save**.

i. Click **+ Add stage** to add the final stage to your process.

j. In the stage side panel's **Label** field, enter `Resolve`.

k. In the **Description** field, enter `Resolve incident and share resolution details`.

l. In the **When to start** field, leave **After Previous** selected, and then click **Save**.

3. Add the *`Create incident from interaction`* activity to the **Classify and Diagnose** stage.
   a. Under the **Classify and Diagnose** stage, click **+ Add an activity**.
   b. In the activity picker, select **Common Activities**, and then select **Automated Create Record** under Non-Interactive.
   c. In the activity properties panel's **Label** field, enter `Create incident from interaction`.
   d. In the **When to start** field, leave **Immediately** selected, and then click **Save**.
   e. Click the **Create incident from interaction** activity card.
   f. In the activity properties panel, click **Configure activity**.
   g. On the Configure your activity screen, locate the Variables section under Inputs.
   h. In the Table Name list, select **Incident [incident]**.
   i. From the Fields list, select **Assigned To**.
   j. Next to the **Assigned To** field, select the data pill picker icon ( ).
   k. Dot-walk to the Interaction record's **Assigned To** field by selecting **Context** > **Input Record - interaction** > **Assigned To**.
   l. In the Fields list, select **Impact** and then select **2 - Moderate**.
   m. Under **Fields**, select **Urgency** and then select **1 - High**.
   n. In the Fields list, select **Short Description**.
   o. Dot-walk to the Interaction record's **Short description** field by selecting **Context** > **Input Record - interaction** > **Short Description**.
   p. In the Fields list, select **Caller**.
   q. Dot-walk to the Interaction record's **Opened for** field by selecting **Context** > **Input Record - interaction** > **Opened for**.
   r. In the **Wait for user input** field, leave **No** selected.
   s. In the **Fields to show after creation** field, enter `priority`.
   t. Click **Update** to finish updating the inputs for the activity.
   The **Create incident from interaction** activity automatically maps the Assigned To and Short Description fields from the interaction record to the incident record when your process runs.
4. Add the **Wait for assignee to update** activity to the **Communicate Work in Progress** stage.
   a. Under the **Communicate Work in Progress** stage, click **+ Add activity**.
   b. In the activity picker, select **Common Activities**, and then select **Wait For Condition** under Interactive.
   c. In the activity properties panel's **Label** field, enter `Wait for assignee to update`.
   d. In the **When to start** field, leave **Immediately** selected, and then click **Save**.

e. Click the **Wait for assignee to update** activity card.

f. In the activity properties panel, click **Configure activity**.

g. On the Configure your activity screen, locate the Variables section under Inputs.

h. Next to the **Record** field, select the data pill picker icon ( ).

i. Dot-walk to the **Create incident from interaction** activity's **record** output by selecting **Activities** > **1:1 - automated_create_record** > **Outputs** > **record**.

j. In the Table list, select **Incident [incident]**.

k. Use the condition builder to add the following condition to your activity: **[Updated by] [is] [Activities** > **1:1 - automated_create_record** > **Outputs** > **record** > **Assigned to]**.

l. Click **Update** to finish updating the inputs for the activity.

The **Wait for assignee to update** activity pauses the process until the Assigned To user for the Incident record updates the record.

5. Add the *`Send update to VIP`* activity to the **Communicate Work in Progress** stage.
   a. Under the **Communicate Work in Progress** stage, select **+ Add activity**.
   b. In the activity picker, select **Common Activities**, and then select **Instruction** under Default.
   c. In the activity properties panel's **Label** field, enter `Send update to VIP`.
   d. In the **When to start** field, leave **After Previous** selected, and then select **Save**.
   e. Click the **Send update to VIP** activity card.
   f. In the activity properties panel, select **Configure activity**.
   g. On the Configure your activity screen, locate the Variables section under Inputs.
   h. In the **Message** field, enter `Notify the VIP user that work on their issue is in progress`.
   i. Leave the **Wait for user input** field's value as **Yes**.
   j. Click **Update** to finish updating the inputs for the activity.
   The **Send update to VIP** activity prompts the agent to send an email to the VIP user when the assignee for the incident record makes an update.
6. Add the *`Wait for incident resolution`* activity to the **Resolve** stage.

   a. Under the **Resolve** stage, select **+ Add activity**.

   b. In the activity picker, select **Common Activities**, and then select **Wait For Condition** under Interactive.

   c. In the activity properties panel's **Label** field, enter `Wait for incident resolution`.

   d. In the **When to start** field, leave **Immediately** selected, and then click **Save**.

e. Click the **Wait for incident resolution** activity card.

f. In the activity properties panel, click **Configure activity**.

g. On the Configure your activity screen, locate the Variables section under Inputs.

h. Next to the **Record** field, select the data pill picker icon ( ).

i. Dot-walk to the *Create incident from interaction* activity's **record** output by selecting **Activities** > **1:1 - automated_create_record** > **Outputs** > **record**.

j. In the **Table** field, select **Incident [incident]**.

k. Use the condition builder to add the following condition to your activity: **[State] [is] [Resolved]**.

l. Select **Update** to finish updating the inputs for the activity.

The **Wait for incident resolution** activity pauses the process until the Incident's state becomes **[Resolved]**.

7. Add the *Share resolution details with VIP* activity to the **Resolve** stage.
   a. Under the **Resolve** stage, select **+ Add an activity**.
   b. In the activity picker, select **Common Activities**, and then select **Instruction** under Default.
   c. In the activity properties panel's **Label** field, enter `Share resolution details with VIP`.
   d. In the **When to start** field, leave **After Previous** selected, and then click **Save**.
   e. Click the **Share resolution details with VIP** activity card.
   f. In the activity properties panel, click **Configure activity**.
   g. On the Configure your activity screen, locate the Variables section under Inputs.
   h. In the **Message** field, enter `Provide the Resolution Notes from the Incident record in an email to the VIP user.`
   i. Leave the **Wait for user input** field's value as **Yes**.
   j. Click **Update** to finish updating the inputs for the activity.
   The **Share resolution details with VIP** activity prompts the agent to send the issue resolution details to the VIP user.
8. In the main header, click **Activate** so that your process runs when triggered.
9. View your activated process as a playbook.

a. Close the Playbooks tab and navigate to **Playbook Experience**.

b. In the side menu, click the lists icon ( ).

c. In the **Lists** tab under Interactions, click **My Interactions**.

d. In the form header, click **New**.

e. On the form, fill in the fields:

Field	Action
Type	Select **Chat**.
Opened for	Select a VIP user.
Assigned to	Select a user that can make updates to Incident records.
Short description	Enter `Testing out the Handle Interactions with VIPs playbook`.

f. In the form header, click **Save**.

g. In the Contextual side panel, click the playbook icon ( ).

Result
Your process appears in as a playbook. Here, agents and fulfillers can get a task-oriented view of the automated business process. Agents can step through the activities that you set up to see where the record is in the overall process.

Triggers

Triggers specify when to start running your playbook.

In Playbooks, triggers indicate when your playbook should start running. Each trigger has a type and conditions that, when met, start running your activated playbook.

You can choose a trigger when you create a playbook in Workflow Studio. Start by adding a trigger, which defines the trigger type. Then, set conditions and other options to refine your trigger so that it fires in a way that makes sense for your business process. For more information, see Create a process definition.

If there are no triggers that fit your use case, you can create your own trigger definition instead. For more information, see Create a trigger definition.

How triggers work

Trigger types

In your Trigger Definition [sys_pd_trigger_definition] record, you can choose a trigger type, which determines when your trigger fires. These trigger types represent record operations that can occur in the ServiceNow AI Platform®. The following trigger types are available in your instance by default:

Record Created

The playbook runs when a user creates a record anywhere in the ServiceNow AI Platform.

Record Updated

The playbook runs when a user updates an existing record anywhere in the ServiceNow AI Platform.

Record Created or Updated

The playbook runs when a user creates a record or updates an existing record anywhere in the ServiceNow AI Platform.

Note:
Triggers only fire for record operations that are interactive, or made by users. Triggers don't fire for non-interactive record operations. For more information, see Non-interactive sessions.

Conditions to run

After you add a trigger to your playbook, you can then set conditions and other options that determine when and how your trigger fires.

Option	Action
Conditions	Use the condition builder to create field conditions for when your playbook runs. See Condition builder.
Run my process	Choose an option for when your playbook runs. Options include: • **Once**: Triggers the playbook once for the life of the triggering input record. • **For each unique change**: Triggers the playbook for every unique update to a non-system field even if the flow is currently running. The system stores a history of every change to a record and determines whether the change is unique. For example, if an incident record's **State** field changes from In Progress to On Hold, the playbook runs. However, if the **State** field then changes back to In Progress, the playbook doesn't run. **Note:** Playbooks that have a trigger that runs **For each unique change** can produce recursions when run in a non-interactive session. When such playbooks make a change to the trigger record, the change meets the playbook's trigger conditions and causes a recursion. • **Only if not currently running**: Triggers the playbook for every unique change if a process execution is not currently running.

Option	Action
	• **For every update**: Triggers the playbook every time the input record is updated, regardless of whether there has already been or there currently are any running process executions.
Run on extended	Select this option to trigger the playbook on tables that extend from your selected table. For example, if you enable this option and select the Configuration Item [cmdb_ci] table, your playbook runs when record operations occur on the Server [cmdb_ci_server], Computer [cmdb_ci_computer], and other extended tables. For more information, see Table extension and classes.

Design considerations

Refer to these design considerations when working with triggers:

Create unique filter conditions for record triggers on the same table

To prevent playbooks from overwriting each other, create unique filter conditions for each playbook that runs on the same table. If multiple playbooks on the same table have the same filter, there is no way to know the order in which the playbooks will run.

Avoid duplicating triggers used in Workflow Studio flows

Playbooks triggers do not override Workflow Studio triggers. For both applications, when the trigger conditions are met, the automated processes run.

Ignore records added or updated by import and update sets

Record triggers ignore records that were added or updated by applying an update set or importing an XML file. These operations apply to the entire application or table instead of an individual record.

Create a trigger definition

Define the type of trigger that determines when to start running your playbook.

Before you begin

- Make sure to set your current application to the application that you want your playbook to run in. For more information, see Application picker.
- Role required: admin, playbook.admin, or pd_trigger_author

Procedure

1. Navigate to **All** > **Process Automation** > **Process Automation Administration** > **Trigger Definitions**.
2. In the context header, click **New**.
3. On the Trigger Definition form, in the **Label** field, enter any label for your trigger definition. This label appears as a trigger option when you Create a playbook.
4. Next to the **Trigger Type** field, click the lookup using list icon ( ).
5. In the Trigger Types list, select a trigger type to use for your trigger definition.

Options include:

Record Created

The playbook runs when a user creates a record anywhere in the ServiceNow AI Platform.

Record Updated

The playbook runs when a user updates an existing record anywhere in the ServiceNow AI Platform.

Record Created or Updated

The playbook runs when a user creates a record or updates an existing record anywhere in the ServiceNow AI Platform.

Note:
Triggers only fire for record operations that are interactive, or made by users. Triggers don't fire for non-interactive record operations. For more information, see Non-interactive sessions.

6. Click **Next** to go on to the next step.
7. In the Table list, select a table whose record operations you want to trigger your playbook.
8. Under Condition, use the condition builder to add field conditions for when you want to trigger your playbook.
9. To trigger your playbook for tables that extend your selected table, select the **Run On Extended** check box.
   For more information, see Table extension and classes.
10. Click **Update** to finish creating your trigger definition.

Result

Your trigger definition is added to the Trigger Definition [sys_pd_trigger_definition] table. You can now select your preset trigger when you Create a playbook.

Add and configure a trigger in a playbook

Begin building your playbook by adding and configuring the trigger.

Before you begin

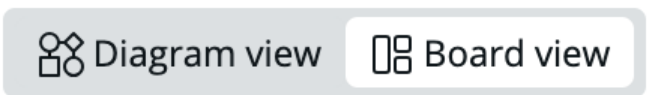
Role required: playbook.admin or pd_author

Review Triggers.

Create a trigger definition if needed.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio** > **Playbooks** > .
2. Select the view you want to build in.
   The builder displays in **Diagram view** by default, but you can select **Board view** to switch views. Switch between views anytime as you build your playbook.



3. Open the configuration modal for the trigger.

4. **Note:**
   The playbook cannot be activated without a trigger, and you will see an error in the notification tray.

   Under the **Schedule** tab, choose how you want the playbook to be triggered.

   - **Define your own conditions for when your process runs**: If you want to create your own custom conditions for when your playbook should run, select this option, choose a trigger type, and then select **Set your trigger conditions**. On the next screen, select a **Table** to trigger your playbook and the **Conditions** that cause your playbook to run. Finally, you can choose to run your trigger on Table extension and classes. When you're done adding conditions to your trigger, click **Done**.
   - **Choose an existing trigger**: If you want to use a trigger that has all the conditions you need for your playbook, select this option. Then, choose an existing trigger from the list and select **Done**.

   Note:
   Playbooks can be triggered off of any table that the customer is entitled to use.

   The trigger is configured.

What to do next

Add and configure your stages.

Stages and activities

In Playbooks, an activity represents one step in your overall business process. You can sequence many activities together in the stages of your process.

Stages

Group activities by the stages of your business process, and sequence activities in an order that makes sense for your cross-enterprise workflow. A stage is made up of many sequenced activities that are grouped in a logical way. In the Playbooks Board view of the design environment, you can add a stage to your playbook by clicking **+ Add stage**. In the Diagram view, select the **+** icon on connectors (lines) to add a stage.

Activities

In Playbooks, an activity represents one step in your overall business process. In the system, an activity is one instance of an activity definition. For more information on activity instances and activity definitions, see the Playbooks User Experience and activity definitions.

Adding an activity to your playbook

In the Playbooks Board view of the design environment, select **+ Add activity** to open the activity picker. In the Diagram view, select the **+** icon on connectors (lines) to open the activity picker. In the activity picker, you can search for an activity to add or select one from the list of Common Activities. To choose an activity for a custom application, first select the application and then select the activity from the resulting list within the picker.

Additionally, you can add an activity just using an automation asset.

If there isn't an activity that fits your use case, you can create your own activity definition to add to the activity picker. For more information, see create an activity definition.

Interactive and Non-Interactive activities

Activity categories include:

Interactive Activities

When an interactive activity runs, it prompts a user for input in your playbook as it runs. For more information, see Interactive activities.

Non-Interactive Activities

When a non-interactive activity runs, it runs entirely behind-the-scenes and requires no user input. For more information, see Non-Interactive activities.

Optional Activities

An optional activity can be inserted by agents and fulfillers during a playbook run. For more information, see Optional activities.

Stage and activity details

When you select an activity, the side panel should open. Under the **Details** tab of the side panel, you can add names, descriptions, and start rules to the stages and activities in your playbook. Click the show or hide additional options button to show or hide additional properties for your currently selected stage or activity. The basic details for each stage and activity include:

Label

You can enter a display name for your stage or activity. This name appears during playbook runtime.

> **Note:**
> Keep your stage and activity names brief, as the system truncates long names.

Description

Optionally, enter a description for your activity or stage. This description only appears within Playbooks and isn't visible during playbook runtime.

Start Rule

Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running:

- **When process starts**: Your stage starts running as soon as the playbook starts.
- **After specific stages**: Your stage starts running after specified stage(s) have finished running.

Under **Schedule** > **Start Rule**, select a start rule for when your activity should start running:

- **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered.
- **After specific activities**: Your activity starts running after specified activities have finished running.

Run condition

Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions.

Activity inputs

Under the **Automation** tab of the side panel, each activity has inputs you can configure:

- To change default inputs and outputs for the activity, open and edit the source flow in Workflow Studio by selecting the **Open link** icon ( ) next to the **Automation** field.

  For more information on how to work with the default inputs and outputs for activities in Playbooks, see Create an action as an activity automation plan.

  Note:
  You must have the appropriate user roles to access Workflow Studio and Workflow Studio. For more information, see User access to Workflow Studio flows.

- **Inputs** are data that you provide so that the activity runs and performs its function appropriately. For example, a Record activity that notifies a customer of an application rejection will have message inputs for the email address, subject, body, and more.
- Expected **Outputs** are displayed at the bottom of the Automation tab.

Creating static and dynamic values for activity inputs

You can add dynamic data to an activity input. Click the data pill picker icon ( ) and navigate, or dot-walk, to the data pill whose dynamic value you want to use when your activity runs. You can select dynamic data from fields in the input record. If your activity has a start rule of **After specific activities**, you can also select dynamic data from the outputs of other activities in your playbook. For more information on dot-walking to related fields and records on the ServiceNow AI Platform, see selecting fields on related tables using dot-walking.

Note:
You can dot-walk to fields in a Reference but not in a Document ID or Sys ID.

The inputs for interactive activities typically provide data that renders in a playbook for an agent to interact with. The inputs for non-interactive activities can render in a playbook but don't require any user input in order to run. For more information, see Interactive activities and Non-Interactive activities.

Design considerations

Refer to these design considerations when working with stages and activities:

Keep ServiceNow AI Platform state models in mind when designing your playbook

Some record types already have state models that describe their life cycle. Use any existing state model as a template for the design of your playbook. For more information, see State Management.

Add and configure a stage in a playbook

Add and configure a stage in your playbook.

Before you begin

Role required: playbook.admin or pd_author

Review Stages and activities.

Procedure

1. Add a stage.
   Your new stage appears, and the Stage properties panel opens.

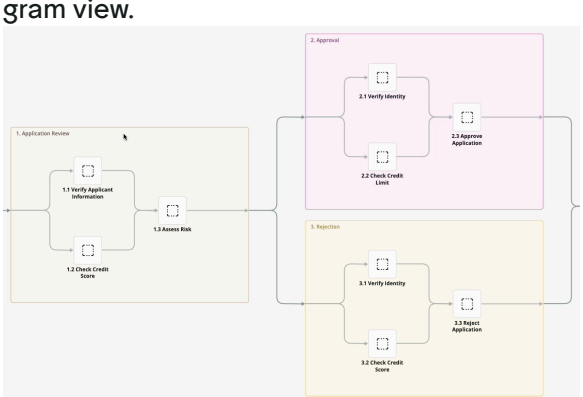
2. Fill in the following fields.

a. Provide the basic details of the stage.

Required inputs	
Label	Enter a unique, user-facing name for your stage. This name appears to agents and fulfillers during runtime of your playbook.
Description	Optionally, enter some descriptive details about your stage.
Run condition	After the stage starts, the stage runs only if specific conditions are met.
Start Rule	Choose when you want your stage to start running. Options include: ▪ **When process starts**: Your stage starts running as soon as the playbook starts. ▪ **After specific stages**: Your stage starts running after specified stage(s) have finished running.

b. **Optional: Show additional options** in the activity for even more granular control.

Warning:
Changing the advanced property fields of an activity can potentially break your automation. Make sure you understand how the playbook and its activities flow before you make changes.

Option	Description
Display order	When there are multiple stages running at the same time, define the order in which stages appear during a playbook run. **Note:** In Workflow Studio, this can also be help ful when viewing parallel activities in Dia gram view. 
Start with delay	Specify a duration of time to wait before run ning an stage whose start rule and conditions

Option	Description
	have been met. For more information, see Start with delay input properties.

3. Click **Save and close** to finish adding the stage to your playbook.
   You can continue to add more stages to your playbook and fill in fields as described in the previous steps. In the Board view, use the **Stage actions menu** ⋮ to add stages between existing stages.
   Stages are created.

What to do next

Add and configure your activities.

Activity definitions

Activity definitions describe how the activities in your playbook get the data that they need when your playbook runs.

Activity definitions provide default configurations and values for your activities so that they can run properly when your playbook is triggered. Each activity definition contains some basic configuration details, as well as an automation plan and activity experience.

The Activity Definition [sys_pd_activity_definition] table lists the definitions for the activities that you can add to a playbook in Workflow Studio. To access these activity definitions, navigate to **Process Automation** > **Process Automation Administration** > **Activity Definitions**.

Fields

Each activity definition record has these basic fields:

Field	Description
Label	Name of the activity to display to users in Workflow Studio.
Table	Name of the table whose records the activity can access as inputs. Typically, this table is either the Task [task] or Global [global] table.
Application	Application scope that the activity can run in.
Accessible From	Options include: • **All application scopes** - Users can access this activity from any application scope. • **This application scope only** - Users can access this activity from the application scope that you specify in the **Application** field only.
Description	Optional description for the activity.
Required Roles	A list of user roles that are allowed to access activities that use this activity definition.

Automation plan

Each activity definition has an automation plan. The automation plan for an activity definition specifies:

- The Workflow Studio flow or action, which drives the activity's automation
- The activity's inputs, which are the data that the activity needs to run yourplaybook

Activity designers can configure the visibility of each activity input.

Include in standard modal

Hides the input from the properties panel. Playbook designers can only see the input from the standard modal when they select the **Show advanced properties** option.

Include in standard modal and configuration panel

Displays the input in the properties panel. Playbook designers can also see the input from the standard modal when they select the **Show advanced properties** option.

Admin visibility only

Hides the input from users who do not have the admin or playbook.admin roles.

Activity experience

Each activity also has an optional activity experience. The activity experience specifies an experience type, associated record, and details for what data to render in the activity's associated playbook card. Activity experience configurations only apply to activities that you add to a playbook which has an associated Playbook user experience. For more information, see Set up Playbook Experiences.

Experience type

An experience type defines the data, or properties, that describe how the activity renders as a playbook card at runtime. For example, a Record experience type tells the system that the activity can display a title, tagline, description, footer, and service level agreement (SLA) information in the Playbook card when your activated playbook runs. For more information, see UI Layouts.

Associated record

The associated record defines the record whose data can render within a Playbook card at runtime. The associated record is dynamic, which means that it changes frequently as the playbook progresses. Because of this dynamic nature, you may want to use the data pill picker () to map the associated record to output record data within the underlying subflow or action specified in the automation plan.

Data to render in the Playbook card

You can specify the data to render in the Playbook card in the sections under the Associated Record section. To add dynamic data to fields that render in this user-facing view, use the data pill picker () next to a data field and navigate, or dot-walk, to the appropriate data pill. The data pill should point to data within the subflow or action specified in the activity definition's automation plan.

Note:
An activity experience contains many sections where you can specify the data to appear within the Playbook card. These sections vary depending on the experience type that you select. For example, a Record experience type has Details, Form, Attachments, and Features sections, while a Knowledge experience type has Knowledge, Details, and Features sections. For more information, see UI Layouts.

Actions to render in the Playbook card

You can specify the Playbook actions that you want to render in an activity's Playbook card using the Playbook Experience Action Assignment Map related list. A Playbook action displays as a button in the Playbook card's footer. Playbook actions can run server scripts, dispatch client actions, or render UI components. For more information, see Custom Playbook actions.

To add a Playbook action to your activity definition, select **New** in the Playbook Experience Action Assignment Map related list. Then, choose a Playbook action from the **Action Assignment** list. Next, choose a Playbook user experience that you want the Playbook action to appear in from the **Playbook Experience** list, and then click **Submit**.

Design considerations

Refer to these design considerations when working with activity definitions:

Avoid calling triggered Workflow Studio flows in an activity's automation plan

To prevent unintentionally running a flow outside of Playbooks, you can use only subflows or actions in activity automation plans. Alternatively, you can set the flow's trigger to only run if not already running. For more information, see Workflow Studio flow trigger types.

Specify default input values in your activity definitions

Preconfiguring default input values for your activity definitions reduces the time and complexity needed for a playbook owner to create a playbook.

Create an activity definition

Specify the action or subflow you want an activity to run. Configure the inputs you want playbook designers to set when adding the activity to a playbook. Select the experience you want end users to have when the activity runs.

Before you begin

- Create a Workflow Studio subflow or action that you want to use as the automation plan for your activity. For example, see Create an action as an activity automation plan.
- Make sure to set your current application to the application that you want your activity to run in. For more information, see Application picker.
- Role required: admin, playbook.admin, or pd_content_author

Procedure

1. To start creating a new activity definition, do one of the following:
   - Navigate to **Process Automation** > **Process Automation Administration** > **Activity Definitions**. Then in the context header, click **New**.
   - Follow the steps to Create a playbook. Then in the Playbooks activity design space, click **Add an activity** > **Create a new activity**.

The Activity Definition form view appears.

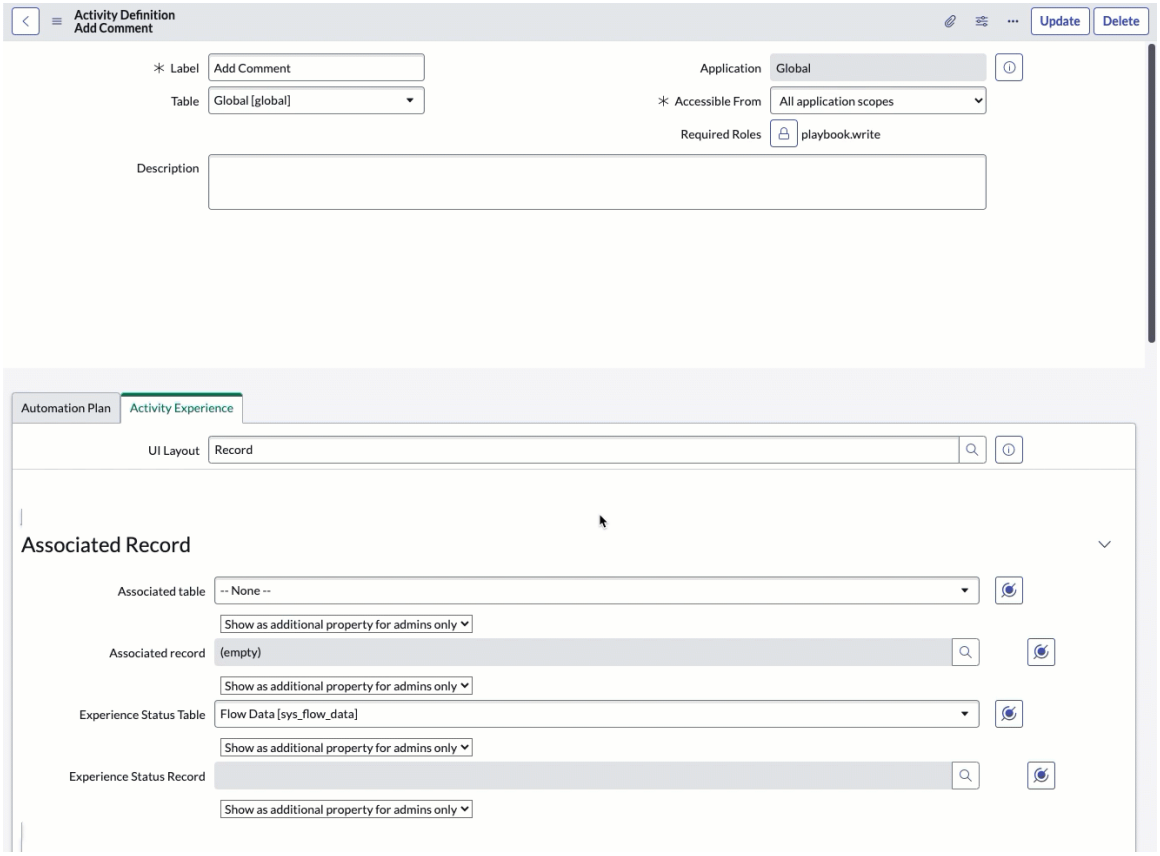
2. Fill in the Activity Definition form fields.

3. Under the Automation Plan section next to the **Flow or Action** field, click the lookup documents using list icon ( ).
   The Select the document screen appears.
4. In the Table Name list, select one of the following options:
   - To use a Workflow Studio subflow to automate your activity, select **Flow**.
   - To use a Workflow Studio action to automate your activity, select **Action Type**.

   Note:
   You can only use published actions or subflows for an activity definition's automation plan.

5. Next to the **Document** field, click the lookup documents using list icon ( ).
   The Flows or Action Types screen appears.
6. From the list, select the subflow or action that you want to use to automate your activity.
7. Select **OK**.
8. From the list, select an UI Layout for the properties and components that you want your activity to use when it renders in a user-facing view of your playbook.
   For more information, see UI Layouts.
9. Click **Submit** to save and create your activity definition record.
   The Activity Definitions list view appears.
10. Under the **Label** column in the list, select your activity definition.
    The Activity Definition form view appears.
11. Select the Automation Plan section.
    The system displays the available variables for the action or subflow. The Workflow Studio Playbooks builder displays a variable for each action or subflow input.
12. For each variable, configure the default value you want each variable to have.
    Leave a variable blank when you want a playbook designer to configure the value when adding the activity to a playbook.
13. For each variable, select where it is visible.
    Playbook designers can only set values for variables that they have access to.
14. Select the Activity Experience tab.
15. Next to the **UI Layout** field, select the list icon ( ).
    The Activity UI Layouts list appears.
16. Select the UI Layout you want to use.
17. Right-click in the header of the activity definition, and select **Save**.
18. Under the Associated Record section, select values for the **Associated table** and **Associated record** fields.
    These values are typically Record and Table Name outputs for the Workflow Studio subflow or action specified in your activity's automation plan. For example, you can click the data pill picker icon ( ) next to the **Associated record** field and dot-walk to the Table Name output by selecting **VL** > **Add Comment** > **Outputs** > **task** > **Approval**.

The system associates a record with your activity so that, when the activity runs, it knows which record's data to output.



19. If you want to set up the default activity data that renders in your playbook during runtime, enter the values for that data in the other sections under Activity Experience.
    The sections and fields that appear under Activity Experience vary depending on the UI Layout that you select. For more information, see UI Layouts.
20. Click **Update** to finish creating your activity definition.

Result

You can now select your custom activity from the activity picker in the Workflow Studio Playbooks design environment. Select the appropriate application scope for your activity to view it in the picker.

Create an action as an activity automation plan

Create an example action to configure and run as an activity from Playbooks.

Before you begin

Role required:

- This task requires some knowledge of creating flows in Workflow Studio environment. For more information, see Building flows.
- This task requires some knowledge of server-side scripting. For more information, see Server-side scripting.
- admin

About this task

Each activity definition requires an automation plan to run Workflow Studio content. The automation plan tells the Workflow Studio Playbooks builder what input values to use when

running an action or subflow. An automation plan can specify static default values or can prompt playbook owners to provide dynamic values when they add an activity to a playbook.

In this example, you create a reusable Workflow Studio action to use as an activity's automation plan. The action you create associates a Task [task] record with a parent record.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. In the main header, click **+ New** > **Action**.
   The Action Properties screen appears.
3. In the Action Properties screen, fill in the following fields:

Field	Action
Name	Enter `Associate Record with Parent`.
Application	Leave **Global** selected.
Accessible From	Leave **All application scopes** selected.

4. Click **Submit**.
   The Workflow Studio design environment appears.
5. Under the Action Outline, click **Inputs**.
6. In the Action Input header, click **+ Create Input**, and then fill in the following fields:

Field	Action
Label	Enter `Record`.
Name	Enter `record`.
Type	Select **Reference.Task[task]**

7. In the Action Input header, click **+ Create Input**, and then fill in the following fields:

Field	Description
Label	Enter `Parent Record`.
Name	Enter `record`.
Type	Select **Reference.Task[task]**

8. Click the add a new step icon ( ).
   The Select Step to add screen appears.
9. Under ServiceNow Data, select the **Update Record** step.
   The **Update Record** step appears in the Action Outline under Inputs.
10. In the **Record** field, click the data pill picker icon ( ), and then dot-walk to **Inputs** > **Record**.
11. Under Field Values, click **+Add Field Value**, and then select **Parent** from the list.

12. Next to the **Parent** field, click the data pill picker icon ( ), and then dot-walk to **Inputs** > **Parent Record**.
13. Under the Action Outline, click **Outputs**.
14. Under Action Output, click **+ Create Output**, and then fill in the following fields:

Field	Action
Label	Enter `Record`.
Name	Enter `record`.
Type	Select **Reference.Task[task]**

15. In the Action Output header, click **+ Create Output**, and then fill in the following fields:

Field	Action
Label	Enter `Parent Record`.
Name	Enter `record`.
Type	Select **Reference.Task[task]**

16. In the Action Output header, click **Exit Edit Mode**.
17. Next to the **Record** output's Value column, click the data pill picker icon ( ), and then dot-walk to **Inputs** > **Record**
18. Next to the **Parent Record** output's Value column, click the data pill picker icon ( ), and then dot-walk to **Inputs** > **Parent Record**.
19. In the main header, click **Save** > **Publish** to make your action available to add to an activity definition's automation plan.

Result

You can now add your custom Workflow Studio action to an automation plan for your custom activity definition.

What to do next

Add your custom action to an automation plan when you Create an activity definition.

UI Layouts

A UI Layout describes what properties and components to use when an activity renders as a card in a Playbook Experience.

When creating an activity definition, you must select an UI Layout to associate your activity definition with. Playbooks come with the following UI Layouts that are available in your instance by default:

- Create Record
- Questionnaire
- Instructional
- Knowledge
- List

- Record
- Record generator

Depending on the UI Layout associated with the activity, different sections and fields appear under the activity definition's Activity Experience. These sections and fields let you set up the activity data that renders during the runtime Playbook Experience.

Note:
All UI Layouts, except for Record generator, display an Associated Record section. For information on how to set up or edit the Associated Record section in an activity definition, see Create an activity definition.

Create Record

If you choose the Create Record UI Layout, the following sections appear in the Activity Experience section for your activity definition:

Associated Record

Displays the following fields:

- Associated table
- Associated record
- Experience Status Table
- Experience Status Record

Details

Displays the following fields:

- Tagline
- Icon
- Title
- Description
- Pending State Title
- Pending State Description
- Record fields
- Footer

Form

Displays the following fields:

- Form View
- Form fields
- Template Fields

Attachments

Displays the following fields:

- Attachment source
- Attachments read only

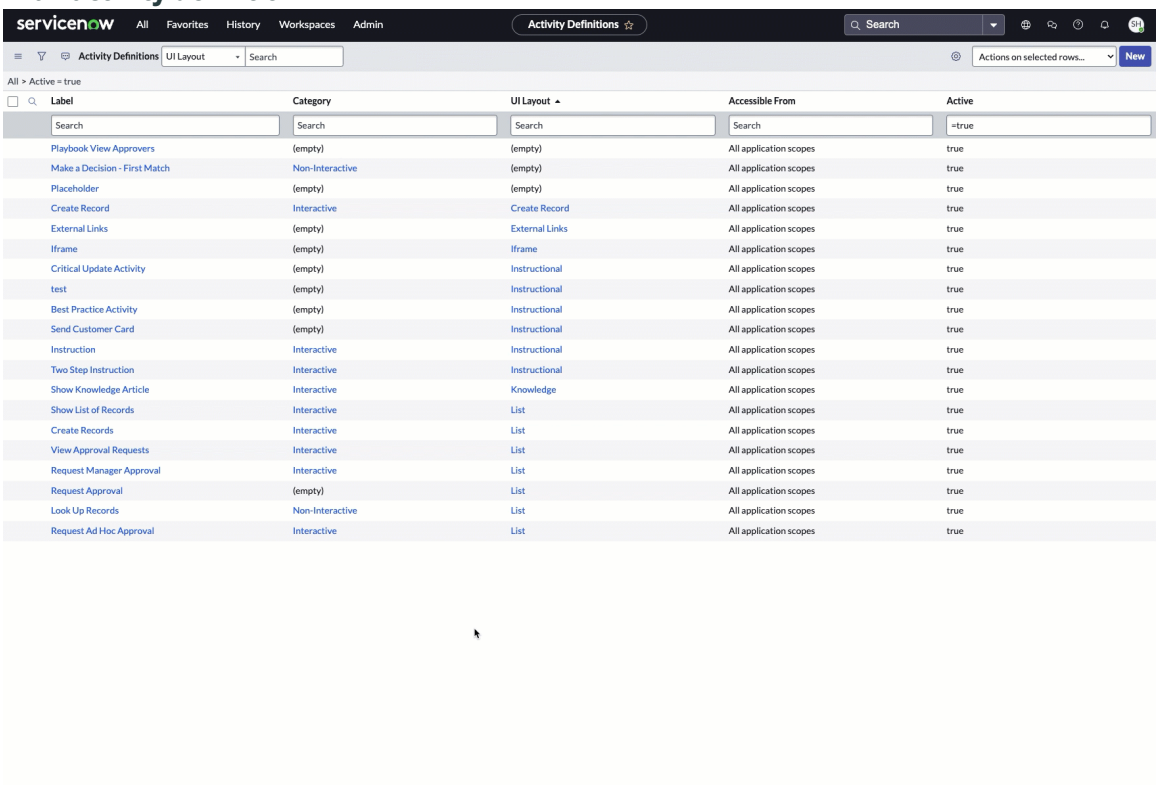
Features

Displays the following fields:

- Show SLA
- Show Checklist
- Is Automated

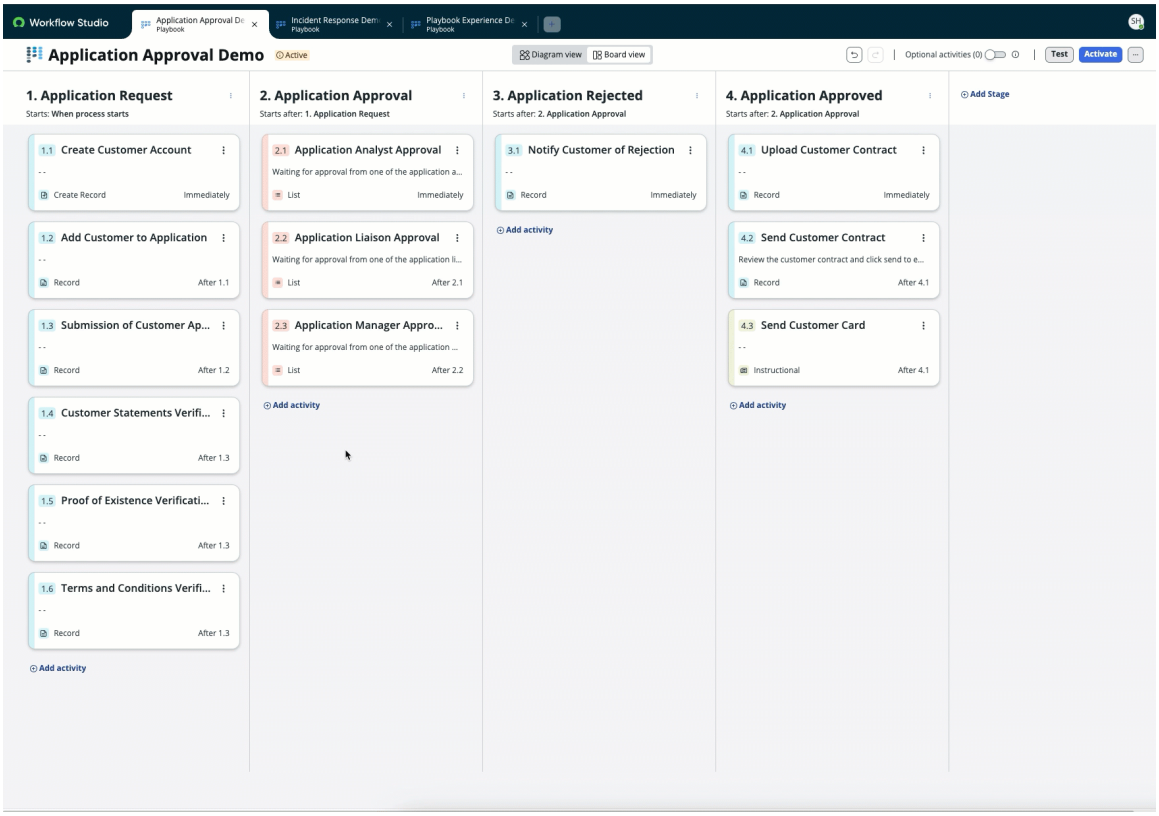
The Create Record activity that comes with the application uses this UI Layout.

In an activity definition



In Workflow Studio, the fields are configured on the UI Layouts tab of the side panel:

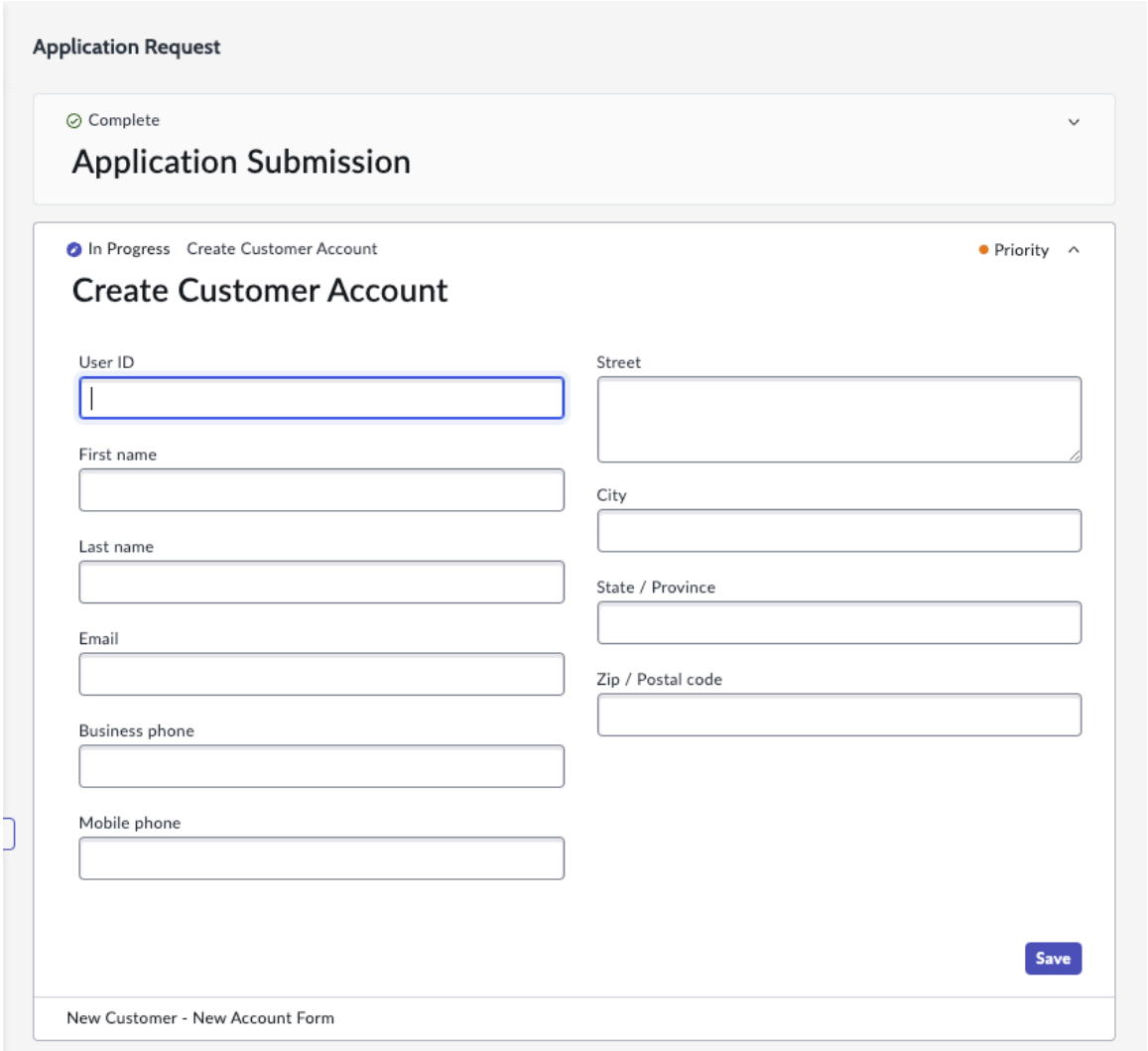
In Workflow Studio



Note:
To learn more about configuring out-of-the-box activities in Workflow Studio, find activity reference documentation in Playbooks reference.

In a running playbook, your configurations are reflected:

In a running playbook, also called a Playbook Experience



Questionnaire

If you choose the Questionnaire UI Layout, the following sections appear in the Activity Experience section for your activity definition:

Associated Record

Displays the following fields:

- Associated table
- Associated record
- Experience Status Table
- Experience Status Record

Details

Displays the following fields:

- Tagline
- Icon
- Title
- Description

- Pending State Title
- Pending State Description
- Record fields
- Footer

Form

Displays the following fields:

- Form View
- Form fields

Attachments

Displays the following fields:

- Attachment source
- Attachments read only

Features

Displays the following fields:

- Show SLA
- Show Checklist
- Is Automated

Instructional

If you choose the Instructional UI Layout, the following sections appear in the Activity Experience section for your activity definition:

Associated Record

Displays the following fields:

- Associated table
- Associated record
- Experience Status Table
- Experience Status Record

Details

Displays the following fields:

- Tagline
- Icon
- Title
- Description
- Footer

Features

Displays the following field: Is Automated

Knowledge

If you choose the Knowledge UI Layout, the following sections appear in the Activity Experience section for your activity definition:

Associated Record

Displays the following fields:

- Associated table
- Associated record
- Experience Status Table
- Experience Status Record

Knowledge

Displays the following fields:

- Knowledge Table
- Knowledge Record

Details

Displays the following fields:

- Title
- Footer

Features

Displays the following field: Is Automated

List

If you choose the List UI Layout, the following sections appear in the Activity Experience section for your activity definition:

Associated Record

Displays the following fields:

- Associated table
- Associated record
- Experience Status Table
- Experience Status Record

Details

Displays the following fields:

- Tagline
- Icon
- Title
- Description
- Record fields

List Details

Displays the following fields:

- List Title
- Table
- List Query
- UI View
- Columns
- Max Columns
- Row Count

Features

Displays the following field: Is Automated

Record

If you choose the Record UI Layout, the following sections appear in the Activity Experience section for your activity definition:

Associated Record

Displays the following fields:

- Associated table
- Associated record
- Experience Status Table
- Experience Status Record

Details

Displays the following fields:

- Tagline
- Icon
- Title
- Description
- Pending State Title
- Pending State Description
- Record fields
- Footer

Form

Displays the following fields:

- Form View
- Form fields

Attachments

Displays the following fields:

- Attachment source
- Attachments read only

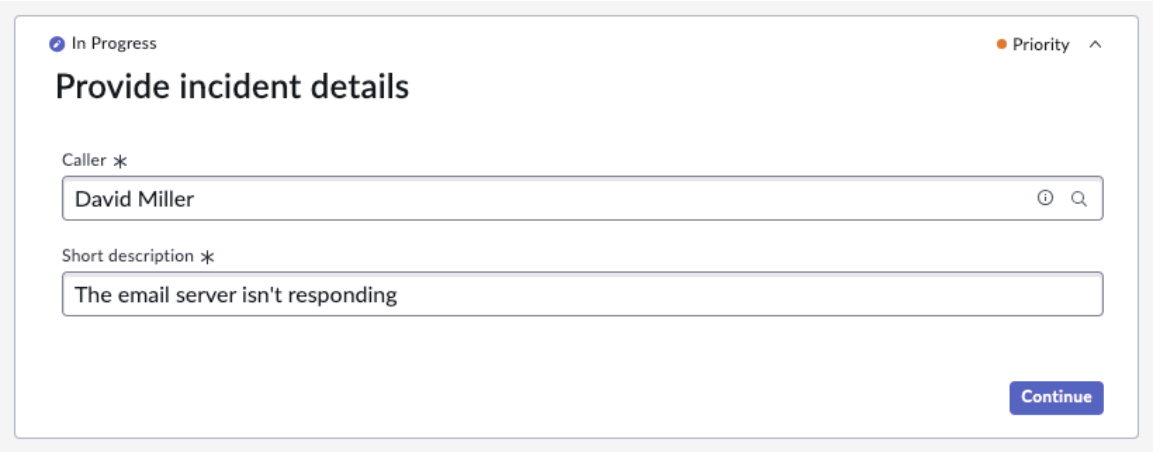
Features

Displays the following fields:

- Show SLA
- Show Checklist
- Is Automated

Record generator

Activities with a Record generator UI Layout create a record during runtime, and redirect users to the record. For example, in this activity, after an agent selects the **Continue** button, a record is created and they are taken to that new record.



If you choose the Record generator UI Layout, the following sections appear in the Activity Experience section for your activity definition:

- Template Fields
- Process Definition Scoped Name
- Associated Table
- Form View

Guided Decision

Note:
The Guided Decision UI Layout is available with a subscription to App Engine or Customer Service Management (CSM). For more information on how to enable this activity for use in Playbooks, see Activate Playbooks for Customer Service Management (CSM).

If you choose the Guided Decision UI Layout, the following fields appear in the Activity Experience section for your activity definition:

- Decision Tree Execution
- Decision Tree

Add and configure an activity in a playbook

Add and configure an activity in your playbook.

Before you begin
Role required: playbook.admin or pd_author

Review Stages and activities.

Procedure

1. Add an activity.
   Your new activity appears in the stage, and the Activity properties panel appears.
2. Under the **Details** tab, fill in the details of your activity.

   a. Provide the basic details of the activity.

Required inputs	
Label	Enter a unique, user-facing name for your activity. This name appears to agents and fulfillers during runtime of your playbook.
Description	Optionally, enter some descriptive details about your activity.
Start Rule	Choose when you want your activity to start running. Options include: ▪ **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. ▪ **After specific activities**: Your activity starts running after specified activities have finished running.

   a. **Optional: Show additional options** the activity for even more granular control.

> ⚠ **Warning:**
> Changing the advanced property fields of an activity can potentially break your automation. Make sure you understand how the playbook and its activities flow before you make changes.

Option	Description
Display order	Define the order in which this activity will ap pear during a playbook run.
Run condition	After the activity starts, the activity runs only if specific conditions are met.
Start with delay	Specify a duration of time to wait before run ning an activity whose start rule and condi tions have been met. For more information, see Start with delay input properties.
Restart rules	Choose what this activity does when a play book is restarted:

Option	Description
	○ **Skip on restart**: Skip this activity when the run is due to a restart. ○ **Run always**: Always run this activity, including first runs. ○ **Skip on first run**: Skip this activity during the first run. For more information, see Enable and Configure Restart for Playbooks.

3. Open the **Automation** tab.
   Automation, inputs, and outputs sections appear.

   Automation

   See or open the action, subflow, or flow that drives the activity.

   Inputs

   Configure inputs for the action, subflow, or flow that is driving the activity. Not all activities require user input.

   Outputs

   Outputs of the action, subflow, or flow that is driving the activity.

4. Under the Inputs section, define the values of the activity's inputs.

   Configure inputs with a hardcoded value, or by dot-walking to a data pill () to use data from previous activities or the playbook trigger.

5. Open the **UI Layout** tab.

   Depending on the UI Layout associated with the activity, different sections and fields appear under the activity definition's Activity Experience. These sections and fields let you set up the activity data that renders during the runtime Playbook Experience.

   Note:
   All UI Layouts, except for Record generator, display an Associated Record section.
   For information on how to set up or edit the Associated Record section in an activity definition, see Create an activity definition.

6. Click **Save and close** to save the details of your activity.
   At any point while in the Playbooks builder, you can add more activities to a stage as described in the previous steps.
7. **Optional:** If the activity you want to add is not in the activity picker, create the activity definition.

   a. Under your stage, click **+ Add activity**.
   The activity picker appears.

   b. Click the **Create a new activity** button.

   c. Fill in the following fields.

Option	Description
Label	Enter a unique, user-facing name for your activity. This name appears to agents and fulfillers during runtime of your playbook.
Accessible From	Where the activity is accessible from. Make the activity accessible to other applications by setting the **Accessible from** field to **All application scopes**. Restrict access by setting the field to **This Application Scope Only**.
Table	Optionally, set to the table associated with the activity.
Application	Read-only field that indicates which applications can use this activity.
Description	Optionally, enter some descriptive details about your activity.
Automation Plan	Select the subflow or action that you want to use to automate your activity.
Activity Experience	Optionally, set the activity's experience type, which helps to define how the activity renders in user-facing views of a playbook card at run time.

8. Click **Submit** to finish creating your activity definition.
   Your activity is defined.

What to do next

Set up the Playbook Experience for you agents and fulfillers.

Automation Assets

Include all automation assets in the activity picker to add a subflow, flow, or action directly to your playbook without having to create an activity definition.

Use cases

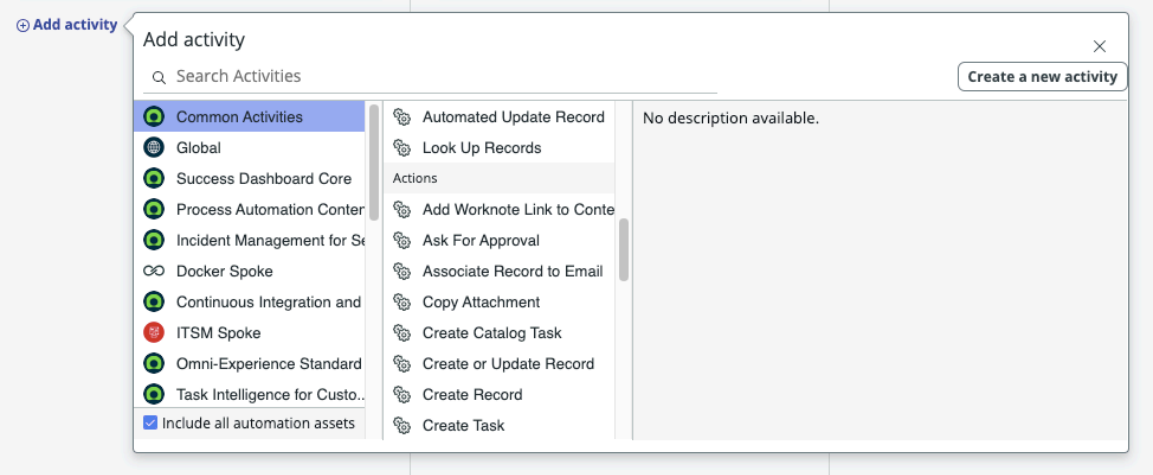
Use automation assets in your playbook if:

- You need an activity that only requires automation.
- The activity doesn't need to be reusable.
- You don't need a UI.

If you need a UI or an activity that is reusable, create an activity definition.

Asset types

When adding an activity, select **Include all automation assets** in the activity picker to add flows, subflows and actions.



When adding a flow, subflow, or action, a non-usable activity definition is created behind-the-scenes.

Start with delay input properties

Specify a duration of time to wait before running an activity or a stage whose start rule and conditions have been met. Give users time to act during automated playbooks. Give users time to wait for a specific date and time to complete actions.

Roles and availability

This input is available for all stages and common activities, except for the Placeholder activity. Users with the admin, playbook.admin, pd_author, or pd_content_author can configure the properties of this input.

Input properties

Open the activity or stage properties panel, show additional options, and enable the **Start with delay** option to configure the following input properties.

Input	Type	Description
Duration Type	Choice	Option to specify how long of duration to wait. Options include • **Explicit Duration**: Wait for a specific time period, such as 5 minutes. • **Relative Duration**: Wait for a specific time period from a selected Duration data pill or date/time value, such as 5 minutes after the playbook starts. • **Percentage Duration**: Wait for a specific percentage of a time period between the start of the activity or stage

Input	Type	Description
		and a specified end time, such as 50%. **Note:** The percentage value must be from **0** through **100** only.
Wait for	Template Value	Option to set a duration value manually or to select a Duration data pill from the data pill picker (). • **Explicit Duration**: Wait duration in hours, minutes, and seconds. • **Relative Duration**: Wait duration in hours, minutes, and seconds before or after a specific time. Select **Relative Duration** to specify a wait duration from a specific date. **Note:** Past dates don't affect the wait duration. You can enter a wait value of up to 999 hours. **Note:** The actual wait duration can vary due to the instance processing time. The playbook always waits for the time that you specify for this field, but other work in the queue may add to the wait time.
Wait for Percentage	Integer	Percentage of time to wait from a specified end date before running an activity or stage. You can manually enter an end date or select a date/time data pill from the data pill picker (). If you select an end date in the past, the wait duration is set to **0**. This field is only available when

Input	Type	Description
		you set the **Duration Type** to **Percentage Duration**.
During the following schedule	Reference.Schedule [cmn_schedule]	Schedule used to calculate an end date that occurs during your hours of operation. For example, the calculated end date for a 10-hour duration that occurs during a 8-5 weekdays schedule will always be one or more business days in the future. If you leave this field blank, the calculated end date does not follow a schedule. For information on creating schedules, see Define a schedule.

Optional activities

Enable your agents and fulfillers to add additional activities as they go through a playbook.

Overview

Admins enable and add optional activities in the Workflow Studio Playbooks builder, while agents and fulfillers add and complete optional activities in Playbook Experience.

As an example, if you have a playbook for a Security Incident, a playbook admin could add an optional activity for a virus scan. When the activity runs, the optional virus scan activity would take the parent Security Incident record and run third-party virus scanning integrations on Configuration Management Database (CMDB) assets associated with the parent record. The security analyst can decide at runtime whether to run a virus scan based on their assessment of the situation.

The flow of working with optional activities:

1. Turn on and add optional activities to a playbook in Workflow Studio.
2. Optional activities are configured like other activities. To see design an automated process.
3. Agents add optional activities in Playbook Experience.

Add an Optional Activity to a playbook

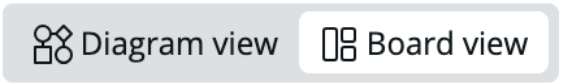
As a Playbooks administrator, add an optional activity to a playbook that Playbook Experience agents and fulfillers can choose to add and complete during the playbook runtime.

Before you begin

Role required: playbook.admin or pd_author

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio** > **Playbooks**.
2. Open the playbook that you want to add an optional activity to.
3. Switch to **Board view**.



4. In the upper right-hand corner, toggle on **Optional activities**.

The optional activities swimlane opens.

5. Add an optional activity.
   Global optional activities can be inserted by the agent or fulfiller anywhere in the playbook, while a stage-specific optional activity can only be inserted within its specific stage.
6. Configure your activity the way that you would configure a regular activity.

Decision activities

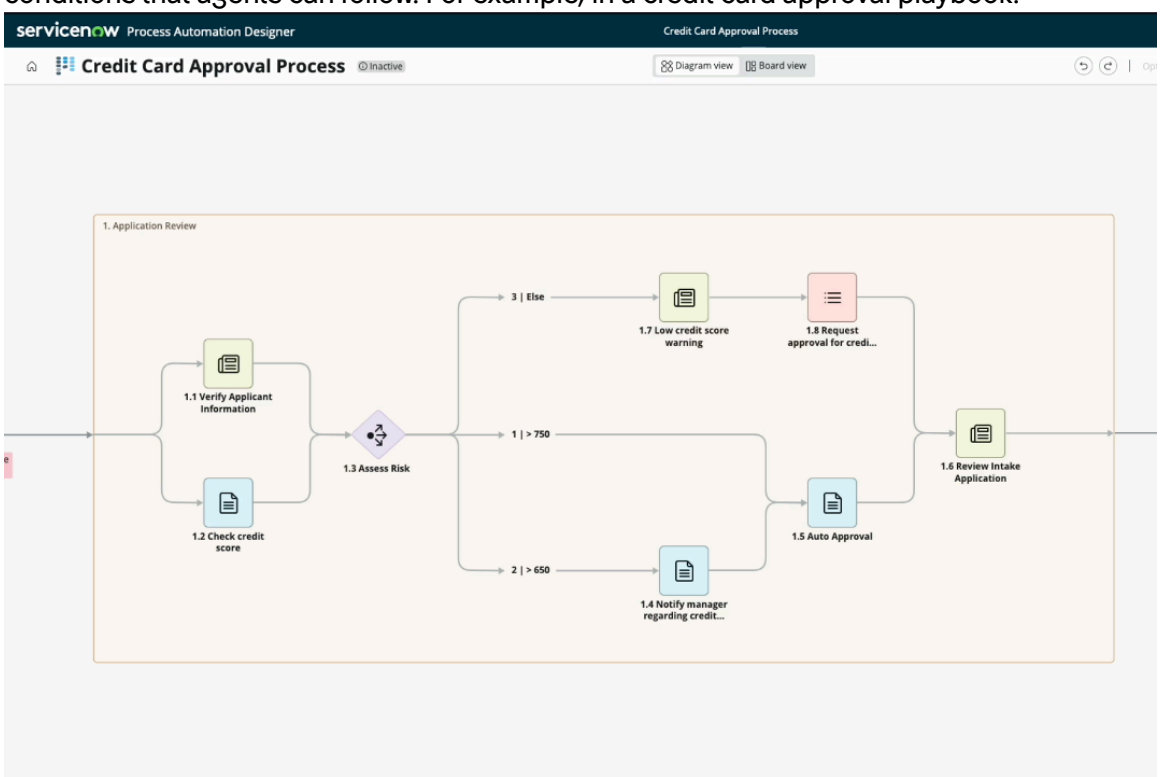
Create and define branches with different conditions for different paths that an agent can follow in playbook.

Before you begin

Role required: playbook.admin

About this task

Add a decision activity to create a decision tree so that agents in Playbook Experience can troubleshoot if-then situations during runtime. Branches are the different paths with different conditions that agents can follow. For example, in a credit card approval playbook:



In this example, an agent can take different actions for different credit scores during a decision activity to assess the risk of a credit card applicant.

Decisions are not currently supported for stages, but you can create different stages and set different **Run conditions** to get achieve the same outcome.

Procedure

1. In Diagram view, hover on the object that you want to insert a decision activity next to, and select the **+** icon to add an activity.

   Note:
   Decision activities can only be added in Diagram view, but can be modified and viewed in Board view.

   The mini-picker displays.

2. Select the diamond icon ◇ to add a decision.
   A decision is added with two branches and the side panel opens for configuration.
3. Under the **Details** tab, fill in the following fields.

Decision details

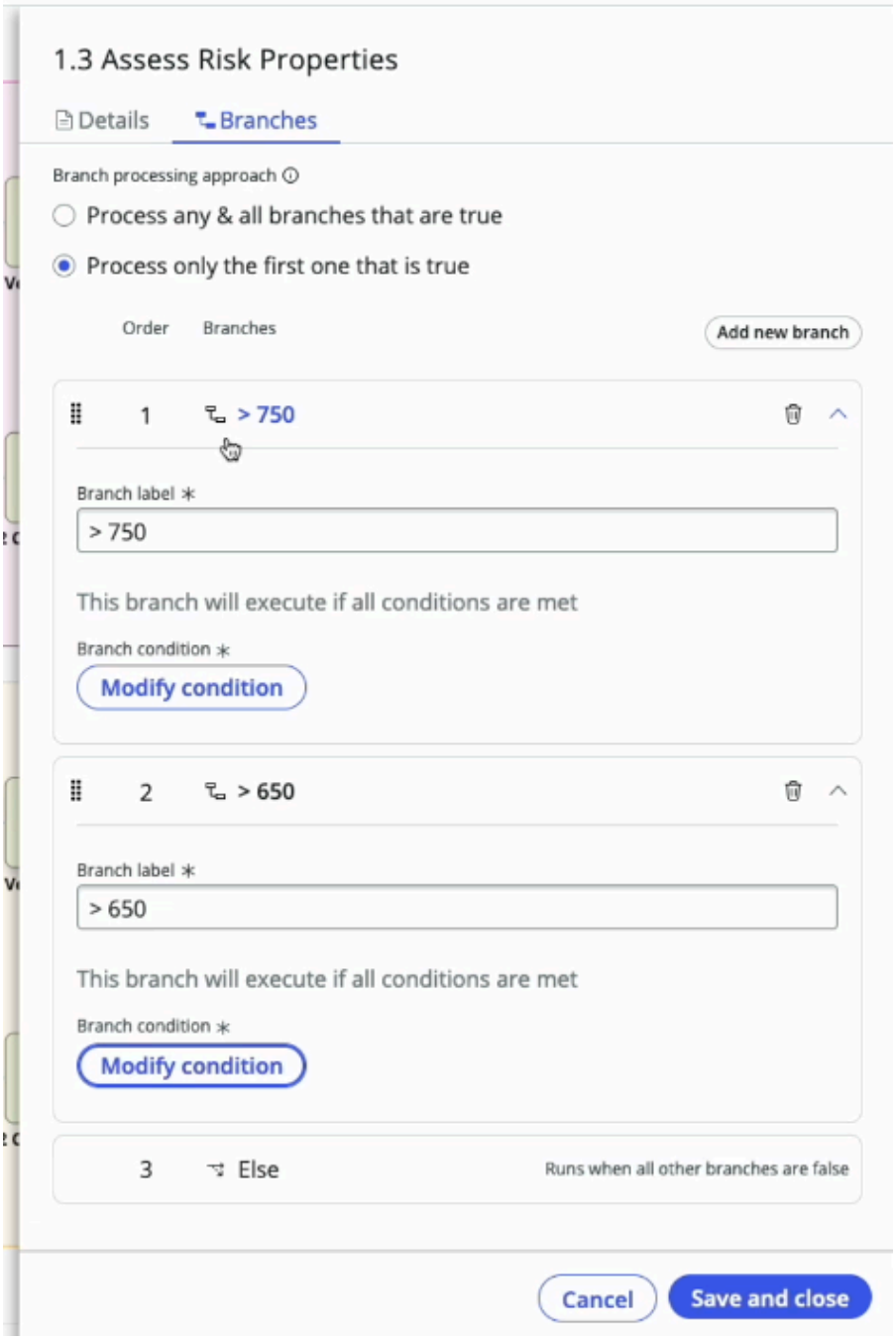
Label	Enter a unique name for your activity. This name appears in your playbook during runtime.
Description	Optionally, enter some descriptive details about your activity.
Start Rule	Choose when you want your activity to start running. Options include: ○ **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. ○ **After specific activities**: Your activity starts running after specified activities have finished running.
Display order	Define the order in which this activity will appear during a playbook run.
Run condition	After the activity starts, the activity runs only if specific conditions are met.
Start with delay	Specify a duration of time to wait before running an activity whose start rule and conditions have been met. Give users a specific amount of time to complete actions. For more information, see Start with delay input properties.
Restart rules	Choose what this activity does when a playbook is restarted: ○ **Skip on restart**: Skip this activity when the playbook run is due to a restart. ○ **Run always**: Always run this activity, including first runs. ○ **Skip on first run**: Skip this activity during the first run. For more information, see Restart a playbook.

4. Under the **Branches** tab, select your new branch to begin configuring it.

   a. Give your branch a unique label.

   b. Select the **Add Condition** button.
   The condition builder displays.

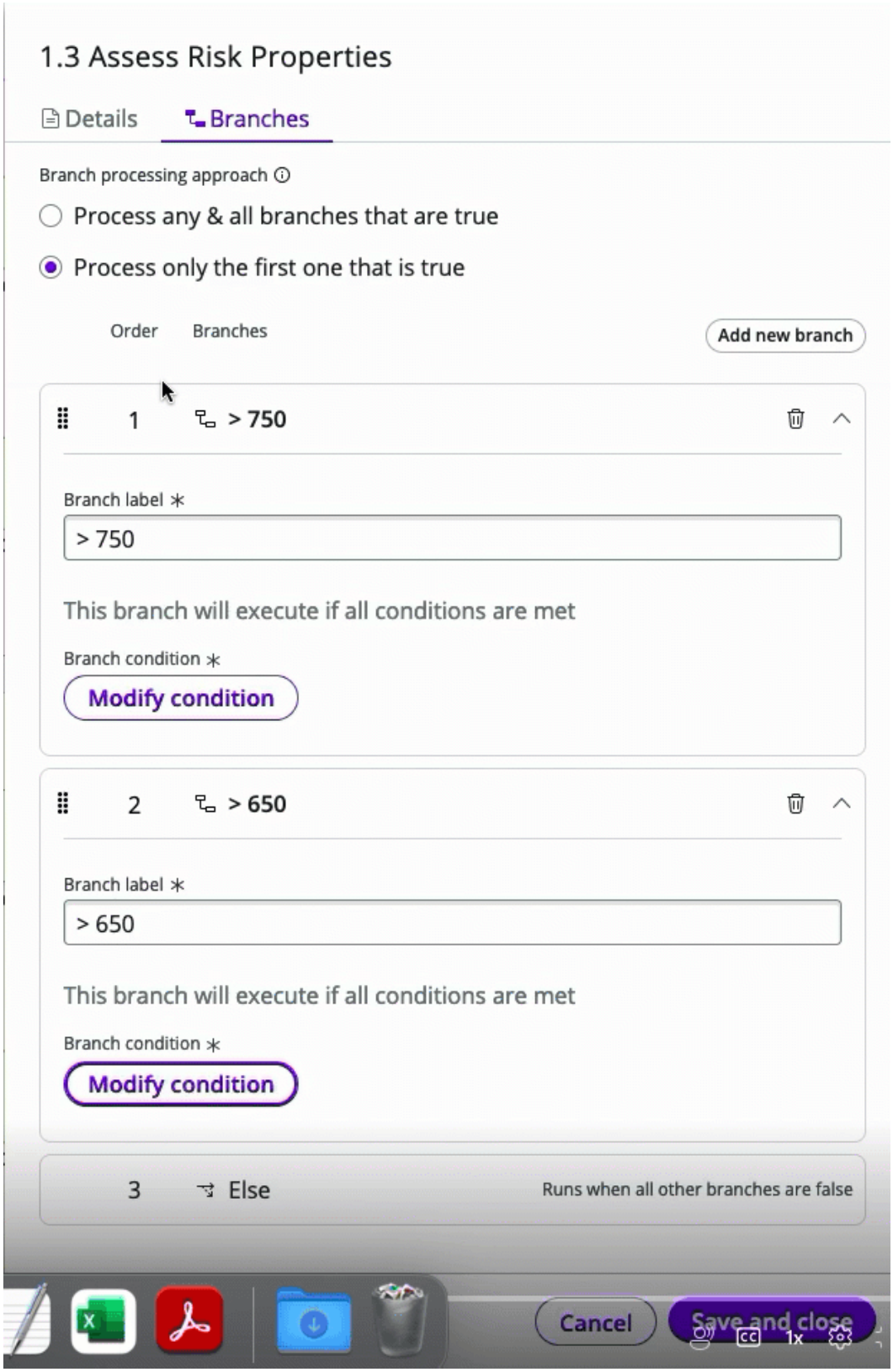
   Note:
   Branches can only be added in the side panel.



   c. Select or enter a field, operator, and value.

5. Add more branches as needed.
   Branches can only be added via the side panel.

6. If you add two or more branches, select whether to process all branches with conditions that are met, or just to process the first listed branch with conditions met.
7. If you selected to **Process only the first one that is true**, drag and drop the branch that you want to be evaluated to the top.



Result
You've added and configured decision branches.

Questionnaire activity

Collects inputs from a user during a playbook run to use later in the playbook.

The questionnaire activity replaces the Collect User Data activity, but does not require you to create a data definition. Use the questionnaire activity if:

- You don't have a table already,
- You don't need to run reports on the collected data,
- And you don't need to use the data outside of the playbook.

If you already have a table to store the collected data, use the User Form activity.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

During a playbook run, you can use data definitions to potentially:

- Collect a shipping address, then reference the address when generating a shipping label.
- Ask the user "yes" or "no" questions, and determine subsequent activities based on the user's responses.

Common properties

These properties are common to all to activities in Playbooks.

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running:

Input	Type	Description
		• **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.

Questionnaire

In the **Questionnaire** tab, you can:

- Add questions for agents to respond to,
- Edit existing questionnaires.

To learn more about adding or configuring questions, see Create a new questionnaire.

Inputs

Many of these inputs are common to activities in Playbooks.

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assignment Group for this Process Step	Reference.Group [sys_user_group]	Assignment group allowed to perform this playbook activity. If you don't set any values for **Assignment Group** or **Assigned To**, any user can read and edit the collected data. The Assignment group and Assigned to fields limit who has access to do so. To specify only individual users, use the **Assigned To** field. The same users do not need to be specified in both fields. **Note:** By default, these fields are mapped to the **Assignment Group** and **Assigned To** fields of the trigger record. This means that users assigned to work on the parent record have access to submit, view and edit the collected data by default.
Assigned to this Process Step	Reference.User [sys_user]	User allowed to perform this playbook activity. If you don't set any values for **Assignment Group** or **Assigned To**, any user can read and edit the collected data. The Assignment group and Assigned to fields limit who has access to do so. To specify only individual users, use the **Assigned To** field. The same users do not need to be specified in both fields. **Note:** By default, these fields are mapped to the **Assignment Group** and **Assigned To** fields of the trigger record. This means that users assigned to work on the parent record have access to submit, view and edit the collected data by default.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Record	Reference.Flow Data	Reference to record containing collected data. Use the pill-picker to dot-walk to **Outputs** > **Record** > **Vars** to see all collected data. To learn more about the pill-picker, see Dot-walking examples.

Create a questionnaire

Create and insert a new questionnaire for agents to respond to.

Before you begin

Role required: admin, playbook_admin, playbook_author, or playbook_content_author

Familiarize yourself with the questionnaire activity inputs and outputs.

Procedure

1. In diagram view, hover on the object that you want to insert a questionnaire activity next to, and select the **+** icon to add an activity.
   The mini-picker displays.
2. Select the questionnaire icon [insert inline icon image] to add a questionnaire.
   A questionnaire activity is added and the side panel opens for configuration.
3. Under the **Details** tab, fill in the following fields.

Label	Enter a unique name for your activity. This name appears in your playbook during runtime.
Description	Optionally, enter some descriptive details about your activity.
Start Rule	Choose when you want your activity to start running. Options include: ○ **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. ○ **After specific activities**: Your activity starts running after specified activities have finished running.
Display order	Define the order in which this activity will appear during a playbook run.
Run condition	After the activity starts, the activity runs only if specific conditions are met.

Start with delay	Specify a duration of time to wait before running an activity whose start rule and conditions have been met. Give users a specific amount of time to complete actions. For more information, see Start with delay input properties.
Restart rules	Choose what this activity does when a playbook is restarted: ○ **Skip on restart**: Skip this activity when the playbook run is due to a restart. ○ **Run always**: Always run this activity, including first runs. ○ **Skip on first run**: Skip this activity during the first run. For more information, see Restart a playbook.

4. Under the **Questionnaire** tab, select **New questionnaire**.
5. Add questions.

> **Note:**
> In the canvas, hover over the activity to see additional actions to edit a questionnaire.

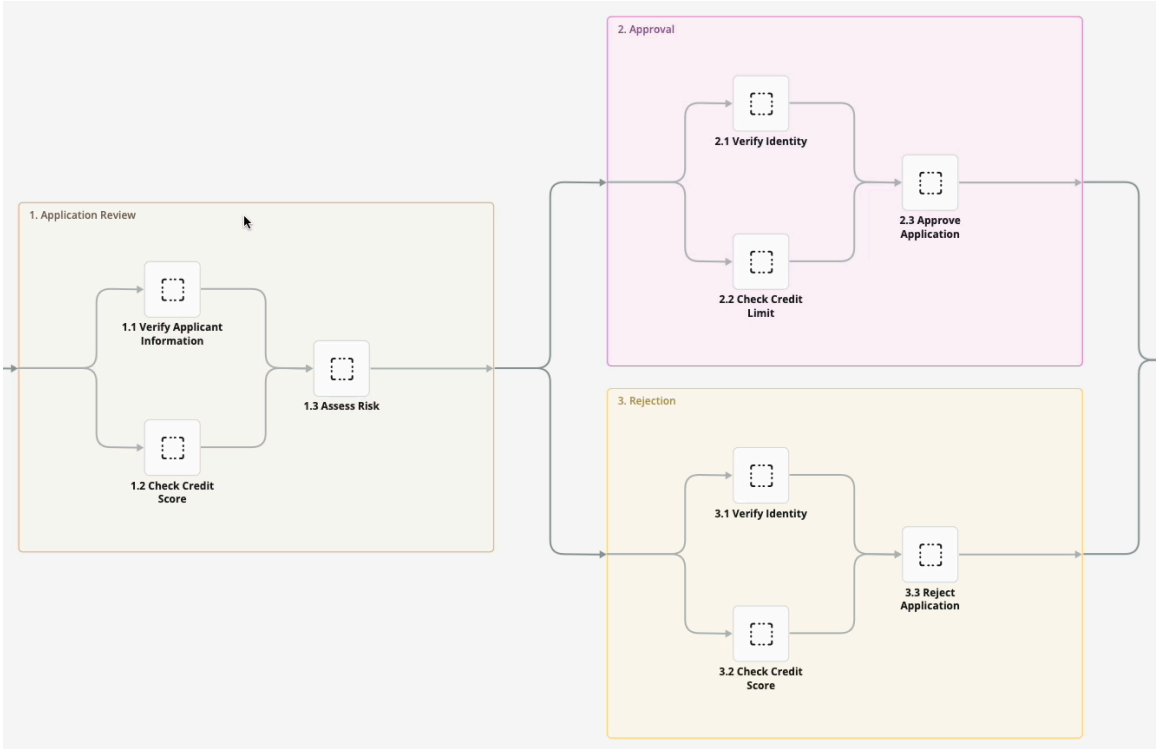
Parallel branches

Add branches for activities and stages that run in parallel to another branch of activities and stages.

Before you begin
Role required: playbook.admin or pd_author

About this task
Parallel activities create branches like decision activities. Unlike decision activities, the activities and stages on a parallel can run on the same conditions as other branches. For example, in this credit card approval playbook:

Parallel activities and stages



In this example, agents verify information or identity at the same time as when they check credit scores. All activities and all stages will run.

Procedure

1. In Diagram view, hover on the object that you want to insert a parallel activity next to, and select the **+** icon to add an activity.
   The mini-picker displays.
2. Select the parallel icon to add a parallel branch.
   A parallel branch is added.

What to do next

Add and configure the stage or activity that will run at the same time as the other branches of stages or activities.

Add dynamic inputs to an activity

Configure your activity to show a certain set of fields based on the value of another input, such as a selected catalog item, selected decision table, or even a REST API response.

Before you begin

Role required: playbook.admin, pd_author, action_designer, flow_designer, admin

Familiarize yourself with the other Workflow Studio components. Dynamic inputs are created with actions and subflows:

1. In the ServiceNow AI Platform, you will create a new data definition for the dynamic input fields you want to add to an activity.
2. In Workflow Studio, you will create a data gathering action.

3. Still in Workflow Studio, you will create a subflow or another action with a dynamic input to consume the first action. This subflow or new action creates a JSON schema that represents the field(s) you want to add to an activity.
4. In the ServiceNow AI Platform, you will create an activity definition using the subflow or action with the dynamic input.

Once your activity definition is created, Playbooks authors can add and configure activities with the dynamic inputs.

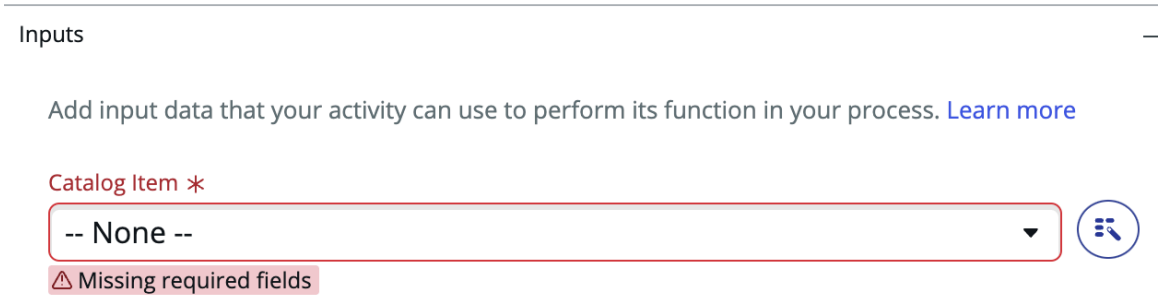
About this task

A dynamic input is an input that changes based on another input. In Workflow Studio Playbooks, you can present multiple dynamic inputs based on another input.

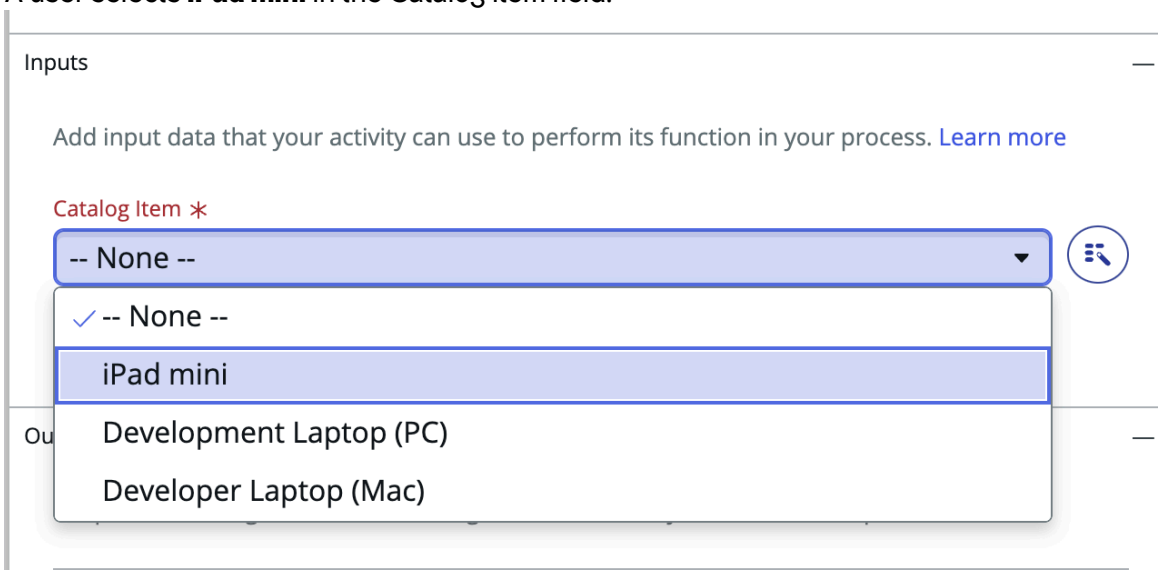
Example

When a user requests catalog items, you can dynamically present a list of catalog variables based on the selected catalog item.

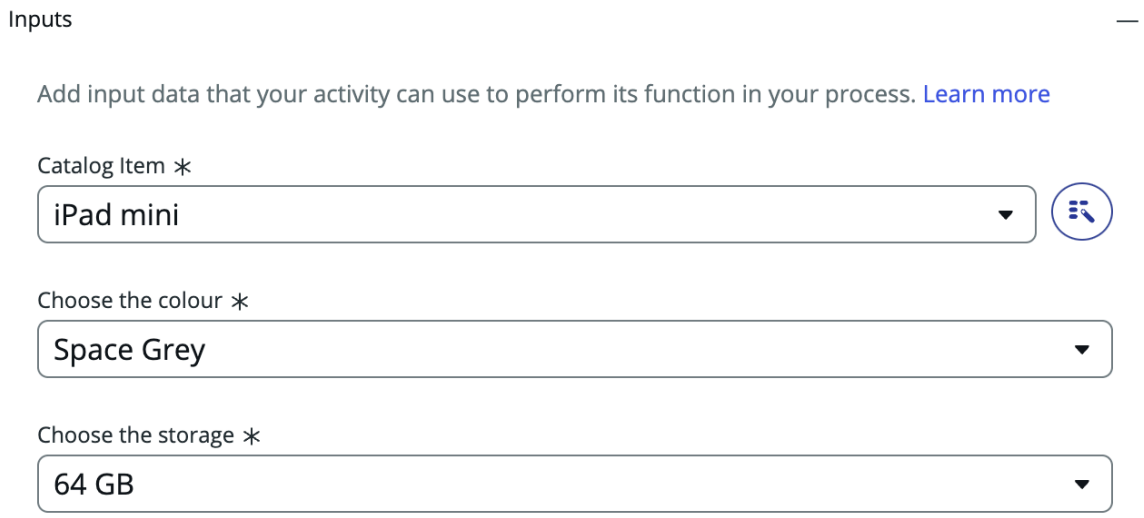
1. The first input required is for the Catalog Item field.



2. A user selects **iPad mini** in the Catalog item field.



3. Two (2) additional fields for color and storage options appear in response to the user selecting an iPad mini as the Catalog Item.



Procedure

1. Navigate to **All** > **Workflow Studio** and select **Actions**.
2. Create an action to add an input.
   The input appears under the **Script step** > **Input Variables** section. The JSON under the **Script** section should include the new input.
3. In the **Outputs** section, click **Edit Outputs** to make sure that the value of the **Name** field is **output**, and that **JSON** is the selected in the **Type** drop-down field.
4. Create a subflow with the new input.
5. Navigate to **All** > **Process Automation Administration** > **Activity Definitions**.
6. Open or create the activity definition to add the new input to.
7. Under the **Automation Plan** tab, make sure the action or flow you created with the new input is the underlying **Flow or Action**, and that the new input appears in the **Variables** section.
8. Change the visibility of the new input from **Show as additional property** to **Always show**.
9. Save your changes.
   Once your activity definition is created, Playbooks authors can add and configure activities with the dynamic inputs.

Playbooks in Workflow Studio

Playbooks are ServiceNow AI Platform® representations of cross-enterprise processes for your organization. Create and activate a playbook to run your digitized business process on the ServiceNow AI Platform.

Each playbook that you design in Workflow Studio has a trigger, a sequence of stages, and a sequence of activities.

You can view your list of Playbooks by navigating **Process Automation** > **Workflow Studio** > **Playbooks**. Opening a playbook allows you to edit it. If there are no playbooks in the list, you can create a new one by clicking **New** and selecting **Playbook**. For more information on creating a playbook, see create a process definition.

Properties

After you create a playbook, access its properties by opening it, and selecting **Properties** in the **More actions** menu in the upper right corner of the header. In the **Additional properties** modal, you can edit the following information:

Field	Description
Label	Name of the playbook to display in Workflow Studio and in a Playbook Experience.
Description	Description of what your playbook does.
Conditions	Conditions that must be met to run your playbook.
Run my trigger	Option that defines how many times your trigger can run for your playbook. Choices include: • **Once**: Triggers the playbook once for the life of the triggering input record. • **For each unique change**: Triggers the playbook for every unique update to a non-system field even if the flow is currently running. The system stores a history of every change to a record and determines whether the change is unique. For example, if an incident record's **State** field changes from In Progress to On Hold, the playbook runs. However, if the **State** field then changes back to In Progress, the playbook doesn't run. **Note:** Playbooks that have a trigger that runs **For each unique change** can produce recursions when run in a non-interactive session. When such playbooks make a change to the trigger record, the change meets the playbook's trigger conditions and causes a recursion. • **Only if not currently running**: Triggers the playbook for every unique change if a process execution is not currently running. • **For every update**: Triggers the playbook every time the input record is updated, regardless of whether there has already been or there currently are any running process executions.
Run on extended	Option to trigger your playbook when record operations occur on tables that extend the input table. For example, if your selected table

Field	Description
	is the Task [task] table and you select this option, your playbook triggers when a Problem [problem] record is created or updated. For more information, see Table extension and classes.

Note:
After you create a playbook, you can't change the trigger's input table or trigger type. For more information, see Triggers.

Design considerations

Refer to these design considerations when working with playbooks:

Avoid duplicating business logic used in Workflow Studio, Workflow, and business rules

Replace separate business logic such as business rules, flows, and workflows with a consolidated playbook. Make sure that you deactivate any external business logic you replace to avoid duplication of effort.

Ignore records added or updated by import and update sets

Record triggers ignore records added or updated by applying an update set or importing an XML file. These operations apply to the entire application or table rather than an individual record.

Create a playbook

Enable playbook owners to configure and organize multiple instances of Workflow Studio content into an automated business process on the ServiceNow AI Platform®.

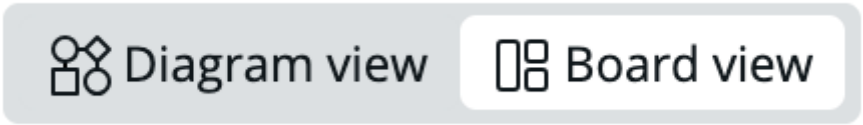
Before you begin

- Activate playbooks for your appropriate application.
- Familiarize yourself with the tables and relationships that your application uses for the playbook that you want to create.
- Make sure to familiarize yourself with any features that your business uses to automate operations on the ServiceNow AI Platform, such as flows, subflows, and actions.
- Learn how to get started with ServiceNow® Process Automation.
- Role required: admin, playbook.admin, or playbook.write.

About this task

Procedure

1. Navigate to **All** > **Workflow Studio** > **Playbooks**.
   The Workflow Studio landing page appears. Playbooks are shown by default, but you can toggle to flows, subflows, actions, and decisions.
2. In the upper right corner, click **New** and select **Playbook** from the drop-down menu.
   The Playbooks builder details screen for a new playbook opens in a new tab.
3. Fill in the following fields.
   The builder displays in **Diagram view** by default, but you can select **Board view** to switch views. Switch between views anytime as you build your playbook.



4. Configure your trigger.
5. Add a stage.
6. Add an activity.
7. Keep adding stages and activities according to your manual playbook.
   For an example of how to design an entire digitized process with Playbooks, see Design an automated process.
8. **Optional:** If you don't see the activity you need to add in the activity picker, create an activity definition.
9. **Optional:** Add a decision activity.
10. **Optional:** Add parallel activities.
11. **Optional:** Add optional activities.
12. After you've added all appropriate stages and activities to your playbook, select **Activate** in the header.
    Activating your playbook publishes it so that it runs when triggered.

> **Note:**
> If you change your playbook after activating it, the system saves your changes but deactivates your playbook. You must click **Activate** again to publish any new changes to your playbook. For more information, see Playbook statuses and activation states.

Result

When your playbook's trigger conditions are met, your playbook runs. As a result, the system creates a Process Execution record and renders user-facing configurations for Playbook Experience. For an example of how to digitize a manual business process that renders as a playbook, see Design an automated process.

What to do next

Set up the Playbook Experience for your agents and fulfillers.

Playbook variants

Use one playbook for multiple scenarios.

Create different variations on top of a base playbook for multiple use cases instead of duplicating and modifying playbooks, or relying on one-time workarounds that use complex run conditions and branching.

Consider using playbook variants if you have similar business processes that change based on specific factors. For example:

- Processes and requirements for different locations (regions, countries, municipalities, organizations, etc.).
  - Banking SLAs differ by region, USA is 30 days and EMEA is 45 days with potential for further variation by country.
  - Card disputes with variance by network: Visa, Mastercard, Discover, etc.
- Managing hiring flows that vary by industry, department, and job titles.

- Business processes and requirements for different users and roles.
- Business processes and requirements for different kinds of applications, such as licenses and permits for different kinds of businesses, etc.

To get started, see Create a playbook variant.

General Guidelines

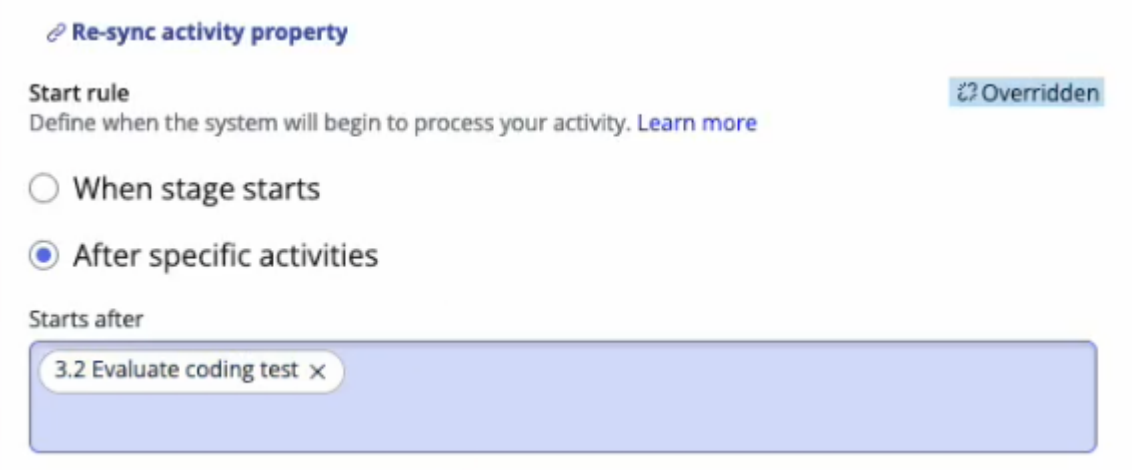
Create, run, troubleshoot, and monitor your playbook variants more effectively. Use these guidelines to optimize the performance of your playbook variants.

Order your variants

Playbook variants are evaluated from the top down at every level. The first variant that meets conditions will run. Make sure your variants are in the order that you would like for them to be evaluated.

Pay attention to property overrides

As you make changes to your variants, related activity configurations that are inherited from parent variants (such as start rules or display order) may be overridden. The overridden link and label appear next properties that are no longer synced with the parent variant or playbook.



Check all properties to ensure they are still configured as you want them. Re-sync properties to be the same as in the parent, if needed.

Don't create variants for playbooks with decision activities

Decision activities are not currently supported in playbook variants.

Don't create variants if you want to change stage properties

Stage overrides are not currently supported in playbook variants.

Create a playbook variant

Create variations of a playbook for different use cases.

Before you begin

Role required: admin or playbook.admin

Procedure

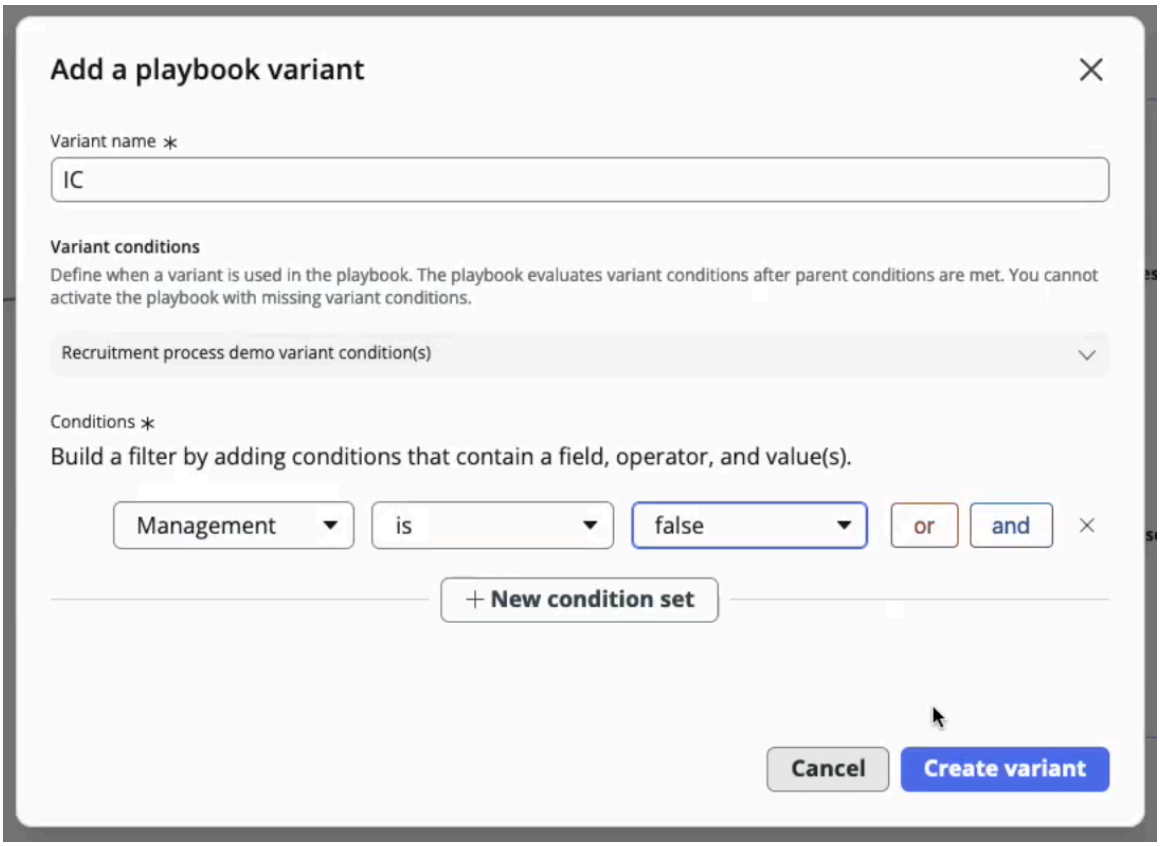
1. Navigate to **All** > **Workflow Studio**.
2. Open the playbook that you want to create variant for.

3. Select the variant icon ( ) to open the variant panel on the left.
4. Select **Add a variant**.
5. Fill in the following fields.

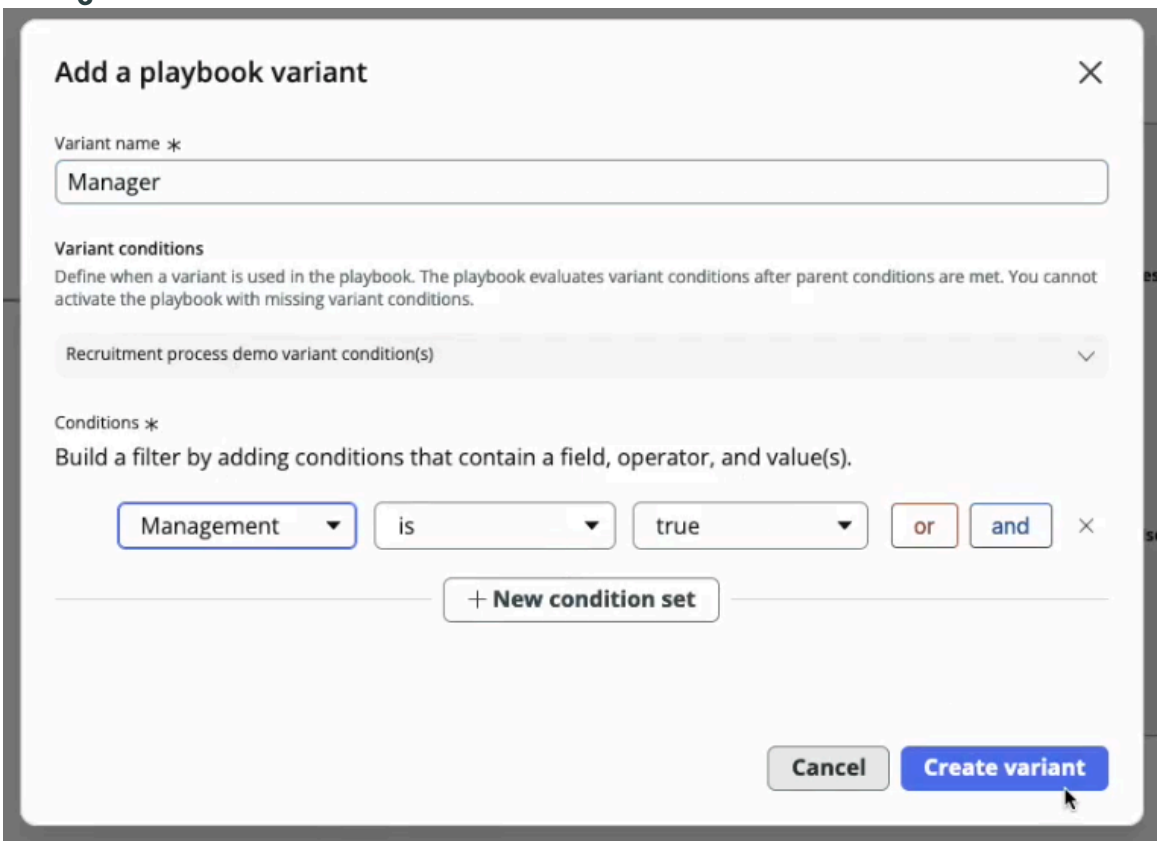
Option	Description
Variant name	Enter a unique, user-facing name for your play book variant. This name appears to agents and fulfillers when this variant runs.
Conditions	In the condition builder, select or enter a field, operator, and value. **Note:** To learn more about condition builders, see Add more conditions for this variant to run, as needed.

Example
In this example, we want to create different versions of a base recruitment process playbook for individual contributors (IC) and managers. In the condition builder, we indicate whether the variant is for management roles or not.

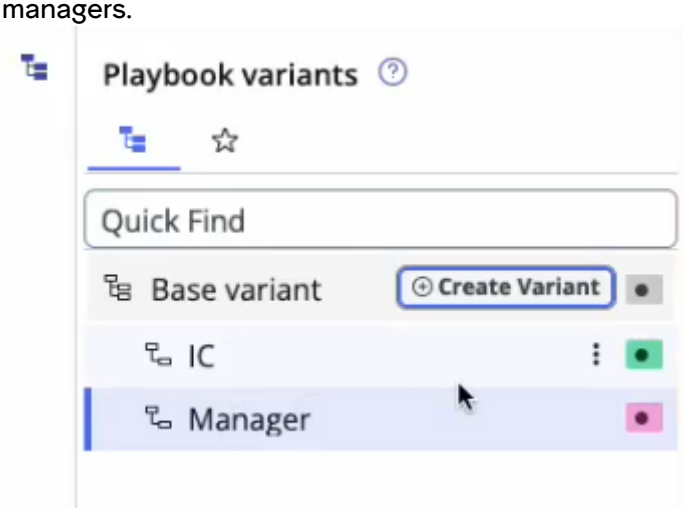
IC variant



Manager variant



6. Select **Create variant**.
   You have two new variants of the base recruitment playbook, one for ICs and one for managers.



   The color of the canvas border corresponds to the color of the variant that is open.

> **Important:**
> The order of your variants is important. Variants are evaluated in the order that they are listed in the panel, from top to bottom. To learn more about reordering your variants, see Re-order playbook variants.

7. As you make changes to your variants, make sure all activity properties are still configured as needed.

Override properties for an activity if they should be different from the parent playbook.

Activities that are the same as in a parent playbook are grayed out. If you change or add an activity, it is shown in full color.

Note:
For more general guidelines, see General Guidelines.

8. **Optional:** Add a child variant to a variant.

a. **Note:**
You can nest child variants up to 6 levels.

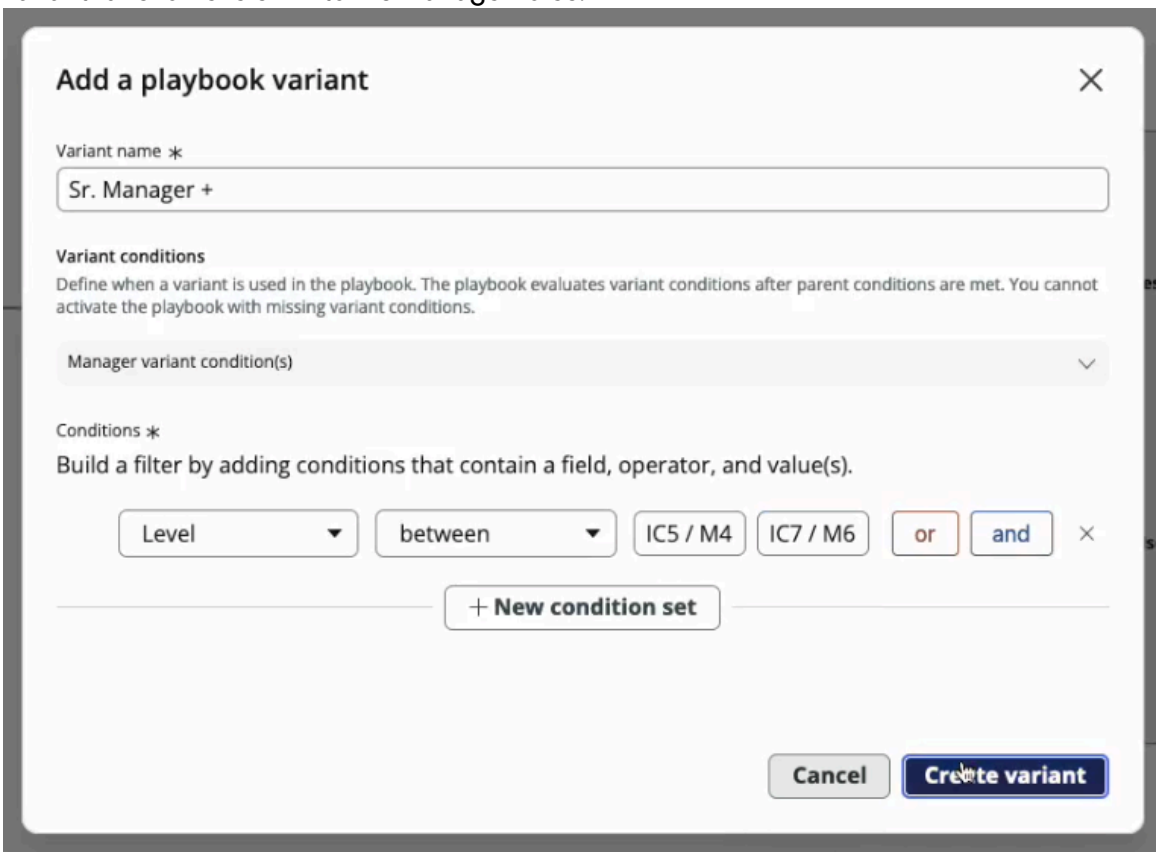
Hover over the variant in the variant panel, and select the kebab menu to open the action menu.

b. Select **Add child variant**.

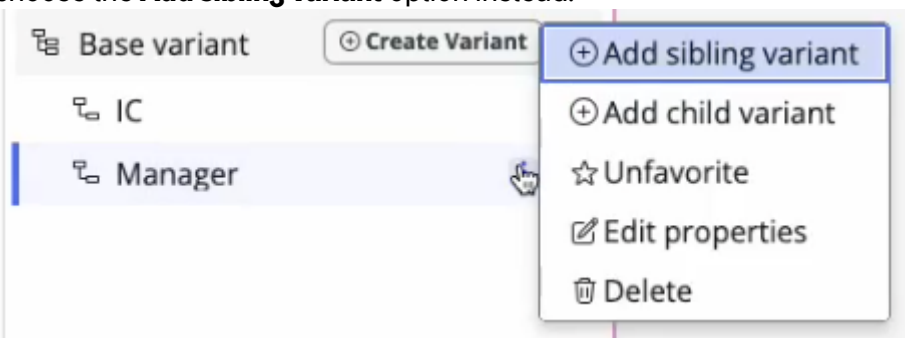
c. Fill in the following fields.

Field	Description
Variant name	Enter a unique, user-facing name for your playbook variant. This name appears to agents and fulfillers when this variant runs.
Conditions	(Optional) In the condition builder, select or enter a field, operator, and value. To learn more about condition builders, see Condition builder. Add more conditions for this variant to run, as needed. **Note:** Child variants automatically inherit the conditions of the parent variant.

Example
In this example, we want to create a child variant of the Manager variant for recruiting senior managers and above. We add an extra condition to specify that this particular Manager child variant runs for levels M4 to M6 manager roles.



9. **Optional:**
   Follow step 8 to add a variant at the same level as another variant (a sibling variant), but choose the **Add sibling variant** option instead.



10. **Optional:** Edit a playbook variant's conditions.
    a. Hover over the variant in the variant panel, and select the kebab menu to open the action menu.
    b. Select **Edit properties**.
11. **Optional:** To save a variant as a favorite, see Save a playbook variant as a favorite.

What to do next
Re-order your playbook variants. To learn more about reordering your variants, see Re-order playbook variants.

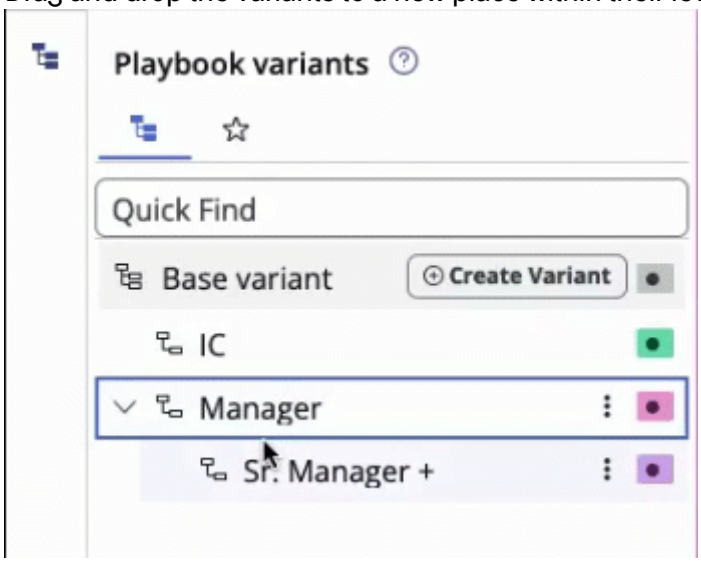
Re-order playbook variants

Change the order in which playbook variants are evaluated.

Before you begin
Role required: admin or playbook.admin

About this task
Playbook variants are evaluated from the top down at every level. The first variant that meets conditions will run.

Procedure
1. Navigate to **All** > **Workflow Studio**.
2. Open the playbook with the variants that you want to re-order.
3. Select the variant icon ( ) to open the variant panel on the left.
4. Drag and drop the variants to a new place within their level.



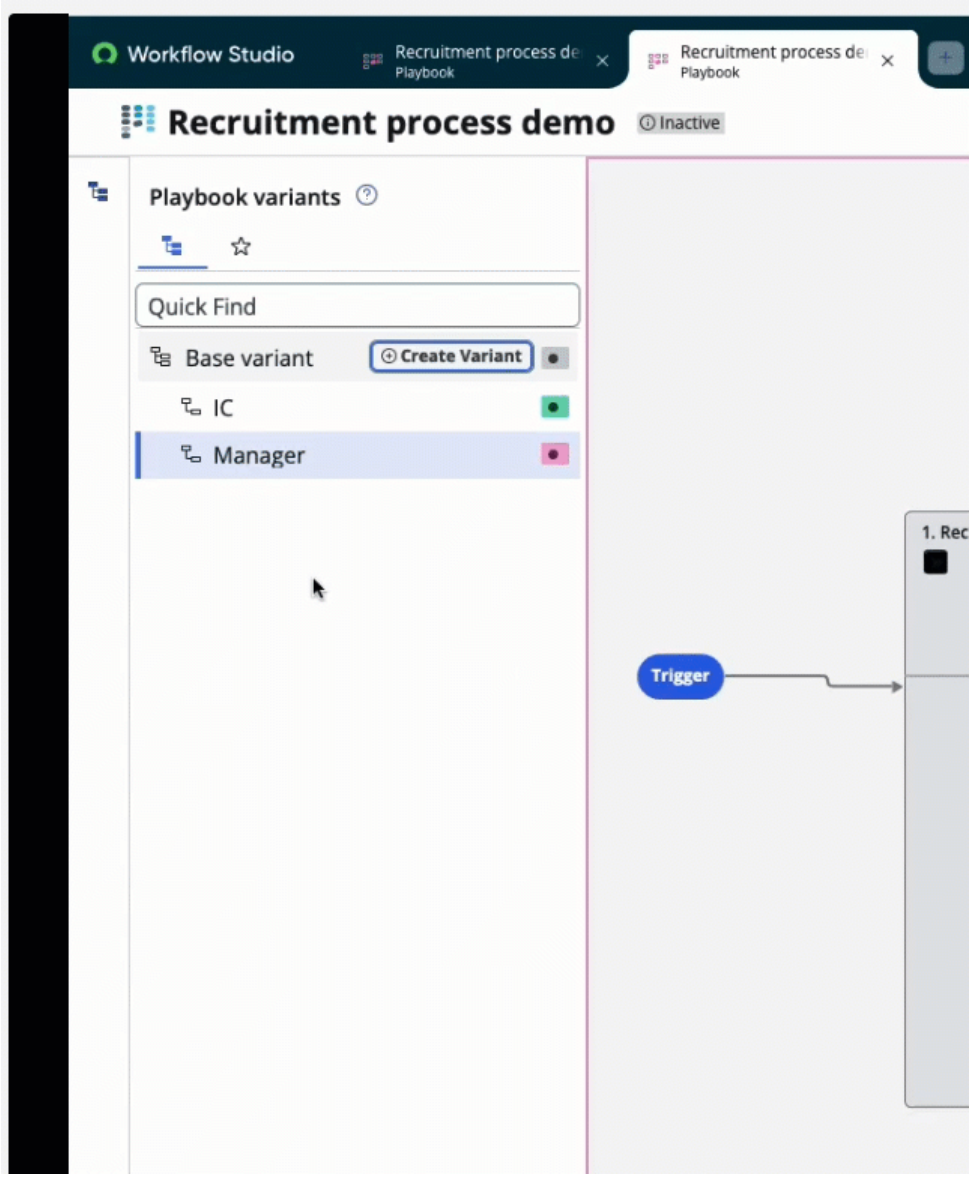
Variants cannot be moved to different levels. They can only be reordered at the sibling level.

Save a playbook variant as a favorite

Save a playbook variant as a favorite for quick reference.

Before you begin
Role required: admin or playbook.admin

Procedure
1. Navigate to **All** > **Workflow Studio**.
2. Open the playbook with the variant that you want to save as a favorite.
3. Select the variant icon ( ) to open the variant panel on the left.
4. Hover over the variant in the variant panel, and select the kebab menu to open the action menu.



5. Set the variant as a **Favorite**.

> **Tip:**
> You can also just search for variants in the **Quick Find** search bar.

Test a playbook

Verify that your playbook works as expected by running the playbook with test trigger data. Identify and resolve all errors before activating your playbook.

Before you begin

- Set up an application in Guided Application Creator to store Playbooks content.
- Create a playbook
- Role required: the admin, playbook.admin, or pd_operator roles grant users access to test playbooks and to view process execution records.

- The Playbook Experience plugin is required to preview a playbook in runtime.
- Plugin required: the Playbook Experience (sn_playbook_exp) plugin grants users access to test playbooks and to preview playbooks in runtime. Users need a role to view process execution records.

About this task

Testing a playbook bypasses the normal playbook trigger to run it with the sample data you provide. Every test you run creates or changes records on your instance. To avoid unwanted record changes, test playbooks on a non-production instance that contains relevant demonstration data.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio** > **Playbooks**.
   The Playbooks landing page appears.
2. Open the playbook you want to test.
   The playbook details screen appears.
3. Select **Test**.
   The system displays the Test your process modal. The contents of the modal depend on the type of trigger record your playbook uses.
4. Select a playbook trigger record to use for testing.

   Note:
   Testing ignores the playbook trigger conditions. You can select any existing record for testing. If there are no existing records to select, you can create a test record.

   The playbook runs as if the selected test record met the playbook trigger conditions.
5. **Optional:** Create a test record.

   a. In the **Test your process modal**, select **Create a new record**.
   In a new browser tab, the system opens a create record form in the ServiceNow AI Platform.

   b. Enter the new record values.
   The fields of a record vary depending on what table your playbook is triggered from.

   c. Select **Submit**.
   The system creates the record with the values you specified.

   d. Return to the Workflow Studio browser tab with the **Test your process** modal open in the Playbooks builder.

   e. Close or cancel the **Test your process** modal.

   f. Repeat steps 2-4 and select the test record you created.

   Note:
   The system only displays the **Create a new record** option when there are no records in the table.

6. Select **Run Test**.

The system runs the playbook using the record you selected as test data. When the test is complete, the system displays options to view process execution details and a playbook preview.

7. **View** the **Process execution details** for information about the playbook state, activities run, and log messages produced.

   Note:
   This option is only visible to users with the admin, playbook.admin, pd_operator roles.

   The system opens a Process Execution form in a new tab.

8. To see the **Playbook preview**:

   a. Select a playbook experience type.

   b. Select **View**.

   Note:
   This option is only visible on instances where the Playbook Experience plugin is activated.

   The system opens a sample playbook in a new tab.

9. Identify and resolve any errors in your playbook.

10. Update and test your playbook until it is ready for release.

What to do next

Publish your playbook to a production instance and activate it.

Restart

Give agents and fulfillers the ability to restart a playbook, stage, or activity.

Overview

In the Workflow Studio Playbooks builder, Playbooks administrators enable restart for Playbook Experience agents and fulfillers. Playbooks can be restarted from the beginning, or from certain activities or stages during runtime. Playbooks administrators also define what each activity and stage does when an agent restarts.

Workflow Studio

In Workflow Studio, Playbooks admins enable restart for playbooks, and define the rules for what an activity or stage does during restart:

- **Skip on restart**: The stage or activity only runs during a playbook's initial run. It does not run on restart.

  Note:
  This setting is helpful if you don't want new tasks or records to be created during a restarted run, because the original execution and resulting record is still relevant.

- **Run always**: The stage or activity always runs, whether during an initial or restarted run.
- **Skip on first run**: The stage or activity runs only on restart. It never runs during an initial run.

Playbook Experience

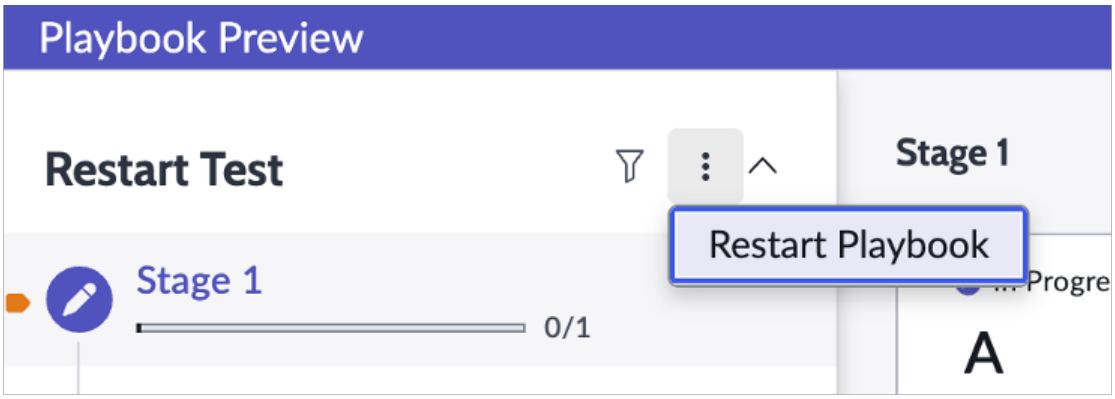
During runtime, agents and fulfillers can restart playbooks from the beginning, or from certain stages or activities.

Note:
Playbooks in an active state can be restarted:

- **In Progress**

Playbooks in a terminal state cannot be restarted:

- **Complete**
- **Error**
- **Cancelled**



The opposite is true for activities and stages. Activities and stages must be complete before they can be restarted.

Design considerations

Follow these design considerations when configuring restart for your playbook, stages, and activities.

Last stages and activities

Avoid setting the last stage or activity of a playbook to **Skip on first run** if there are no parallel stages or activities. If the playbook is restarted before the last stage or activity can run, the last stage or activity never runs.

Stages

Avoid grouping all activities that are configured to **Skip on first run** in one (1) stage. If you do so, the stage is completely hidden the first time that it runs. The stage must run twice to become visible.

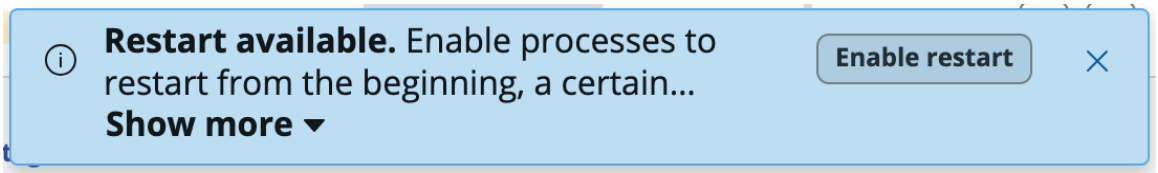
Enable and Configure Restart for Playbooks

Configure your playbook so that agents and fulfillers in Playbook Experience can restart a playbook from the beginning, or from a specific stage or activity.

Before you begin
Role required: pd_author

When you open an existing playbook in Workflow Studio for the first time after upgrading to the Process Automation Designer 25.1.2 ServiceNow Store app, a banner message notifies you to

enable restart for your playbook. You must enable the restart feature before you can perform the task below.



This message only displays for existing playbooks. Restart is automatically enabled for new playbooks and does not change any other features and functions.

Note:
Once enabled, restart cannot be disabled. If you don't want agents to be able to restart a playbook or the activities and stages in a playbook, do not perform the following task.

Procedure

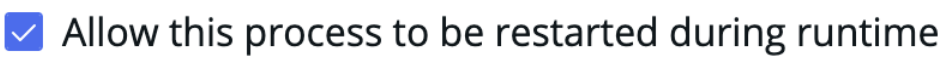
1. Enable restart for an entire playbook.

a. Navigate to **All** > **Process Automation** > **Workflow Studio** and select **Playbooks**.
The Playbooks list displays.

b. Open the playbook you want to enable restart for.
The Playbooks builder displays.

c. In the upper right-hand corner, open the **More actions menu** ··· , and select **Properties**.
The **Additional Properties** modal displays.

d. At the bottom of the **General** tab, check the **Allow this process to be restarted during runtime** box.



Agents can now restart the whole playbook.

Note:
Restart can be enabled for activities and stages even if the whole playbook cannot be restarted.

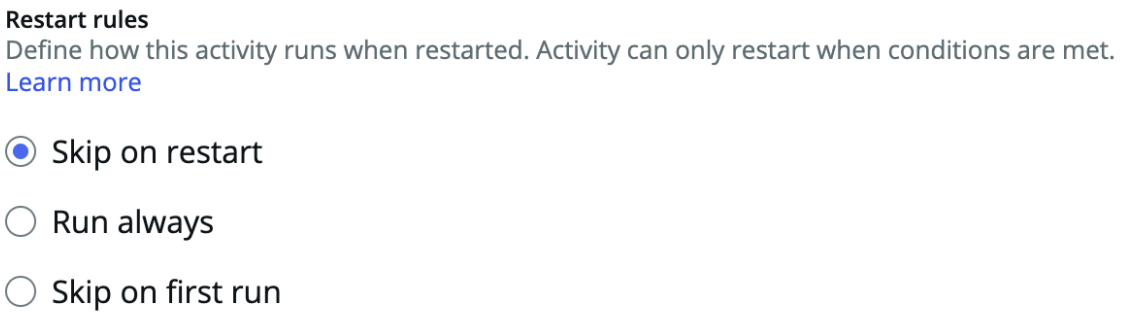
2. Define restart rules for each stage and activity.

Note:
Restart settings for a stage do not apply to its activities. Each activity also has its own restart settings.

a. Navigate to **All** > **Process Automation** > **Workflow Studio** > **Playbooks**.
The Playbooks list displays.

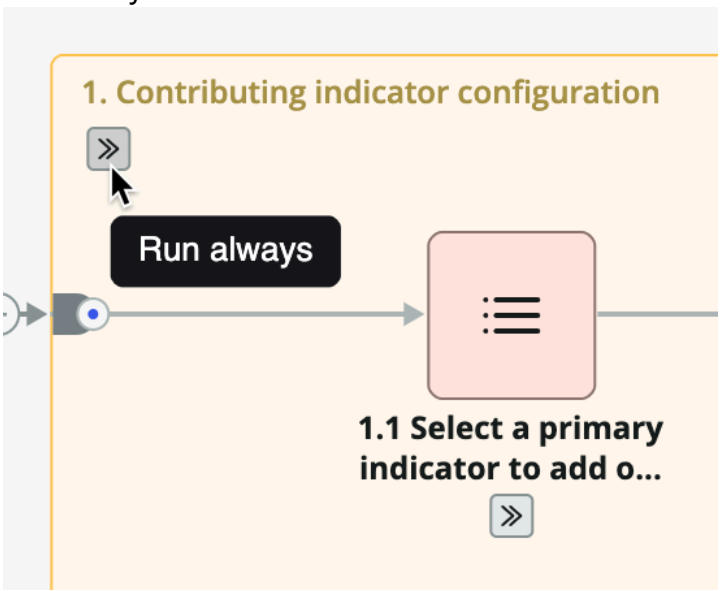
b. Open the playbook you want to configure stage or activity restart settings for.
The Playbooks builder displays.

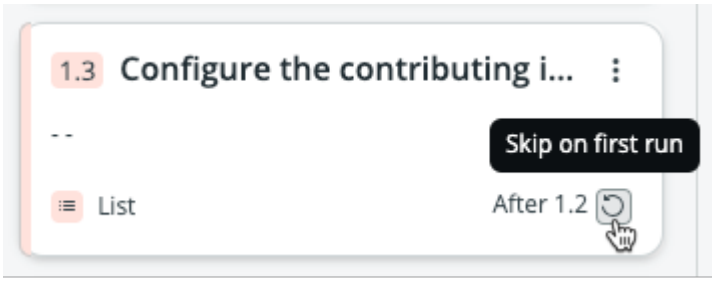
c. Open the stage or activity you want to configure restart settings for.



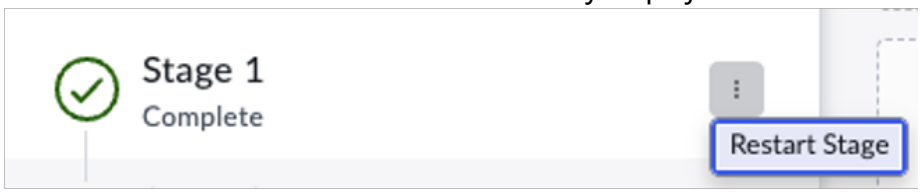
d. Select what you want the stage or activity to do when restarted.
Your restart rules are applied.

- Restart settings are reflected in both the Diagram view and Board view of the Workflow Studio Playbooks builder.



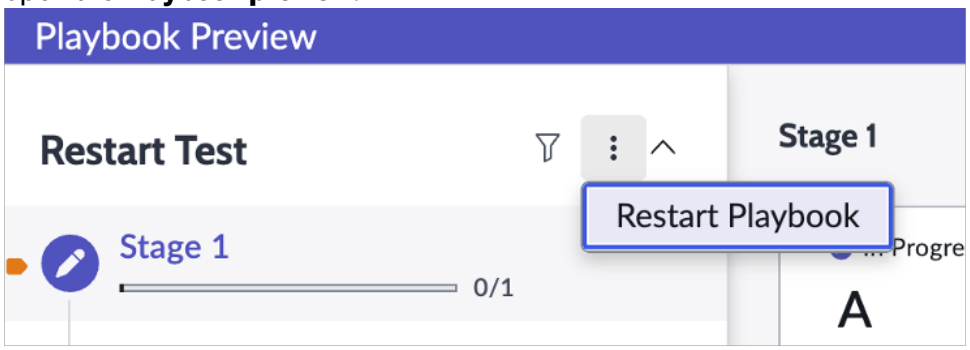


- Restart buttons are added to context menus in your playbook.



3. Optional:

To test your playbook with the restart options, select **Test** in the upper right-hand corner and open the **Playbook preview**.



Duplicate Playbooks

Make a copy of an existing playbook with the same trigger, stages, activities, and experience configurations as the original. Edit the duplicated playbook to quickly create a working variation.

Before you begin

- Activate playbooks for your appropriate application.
- Familiarize yourself with the tables and relationships that your application uses for the playbook that you want to create.
- Make sure to familiarize yourself with any features that your business uses to automate operations on the ServiceNow AI Platform, such as flows, subflows, and actions.
- Learn how to get started with ServiceNow® Process Automation.
- Role required: admin or playbook.admin

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio** > **Playbooks**.
   The Playbooks builder appears.
2. Select the playbook you want to duplicate.
3. Check the box next to the playbook that you want to duplicate.
   Only one playbook can be duplicated at a time.
   The **Duplicate** button displays.
4. Select **Duplicate**.
   The system makes a copy of the selected playbook and opens **Duplicate playbook** modal for the new playbook.
5. Fill in the playbook properties.

Option	Description
Label	Enter a unique, user-facing name for your play book. This name appears during playbook run time.
Application	Choose an application scope that you want your playbook to run in. Selecting **Global** lets your playbook run in any application scope. For more information, see Application

Option	Description
Description	Optionally, enter some descriptive details about your playbook.
Trigger type	Specify what causes your trigger to fire.
Table	The table with the record operations that you want to trigger your playbook. This field can not be edited.

6. Edit the playbook stages and activities to fit the new playbook.
7. Select **Save**.

Designing playbooks

After building playbooks in Workflow Studio, Playbook Experience administrators can configure the layouts and further customize playbook functions during runtime.

There are many different ways that a Playbook Experience admin can customize the layout of a playbook during runtime. Playbooks can be designed and customized in UI Builder for:

- A configurable workspace,
- Service Portal,
- or the ServiceNow Mobile Platform.

Once customized to your satisfaction, mobile playbooks must be embedded. For playbooks that in a configurable workspace or Service Portal page, there are no additional steps needed before end users can run your playbook.

Customize the Playbook Experience

Customize the layout of your Playbook Experience with base system modular components, templates, and more via integrated UI Builder functionality.

Build custom experiences for your agents and fulfillers with the modular components in the Playbook Experience apps. To speed up the development process, add the Custom Layout UI Controller to pages where you want presets to auto-populate inputs for the provided components.

Note:
You can also create your own custom components with UI Builder, but presets cannot auto-populate custom components with data.

Components

Determine how to display these modular components in your Playbook Experience:

Playbook stage picker

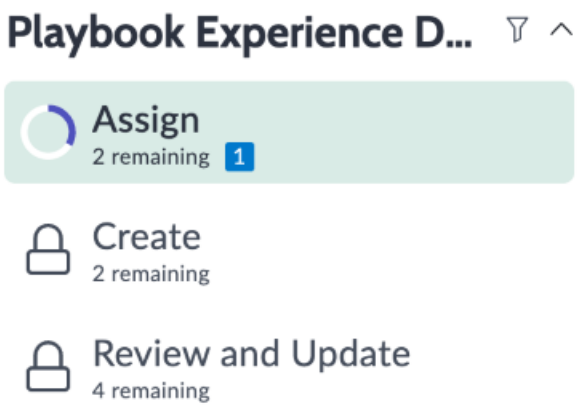
The stages of a playbook can be displayed horizontally or vertically:

- The vertical orientation shows:
  - All playbooks for a parent record,
  - The stages of each playbook,

- How many activities remain to be completed in each stage, and
- How many activities are in progress for each stage.

In this example, a playbook named **Playbook Experience Demo** with stages named **Assign**, **Create**, **Review and Update** is shown. The **Assign** stage has 2 activities left to complete, with 1 in progress. The **Create** and **Review and Update** stages don't have blue numbers next to them, so they don't have any activities that are in progress.

Vertical stage picker



- The horizontal orientation only shows the stages of a selected playbook. To see the activities of a stage, select the stage.

Horizontal stage picker

Playbooks with more than 5 stages paginate.

Note:
The horizontal stage picker only shows you the playbook you are in and its stages, so you can't navigate between playbooks unless you:

- Use a playbook template, or
- Create your own drop-down component with UI Builder.

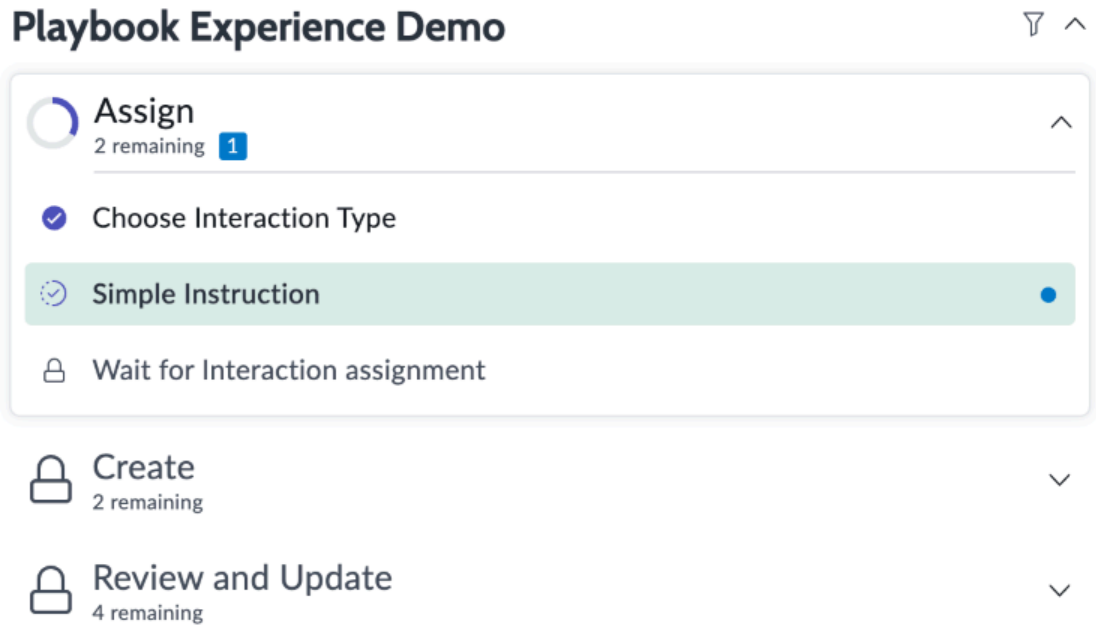
All playbook templates include a drop-down component to navigate between playbooks, and is auto-populated when you select the **Playbook Picker** preset.

Playbook activity picker

The activity picker is where you navigate between activities.

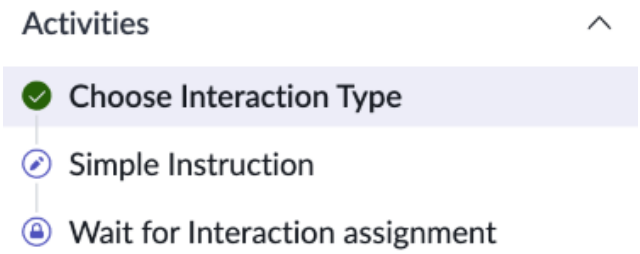
If you're using the vertical stage picker, turning on the activity picker means each stage can be expanded to shows its activities.

Activity picker when viewing stages vertically



If you're using the horizontal stage picker, turning on the activity picker allows you to expand or collapse the entire list of activities for the stage you're in.

Activity picker when viewing stages horizontally



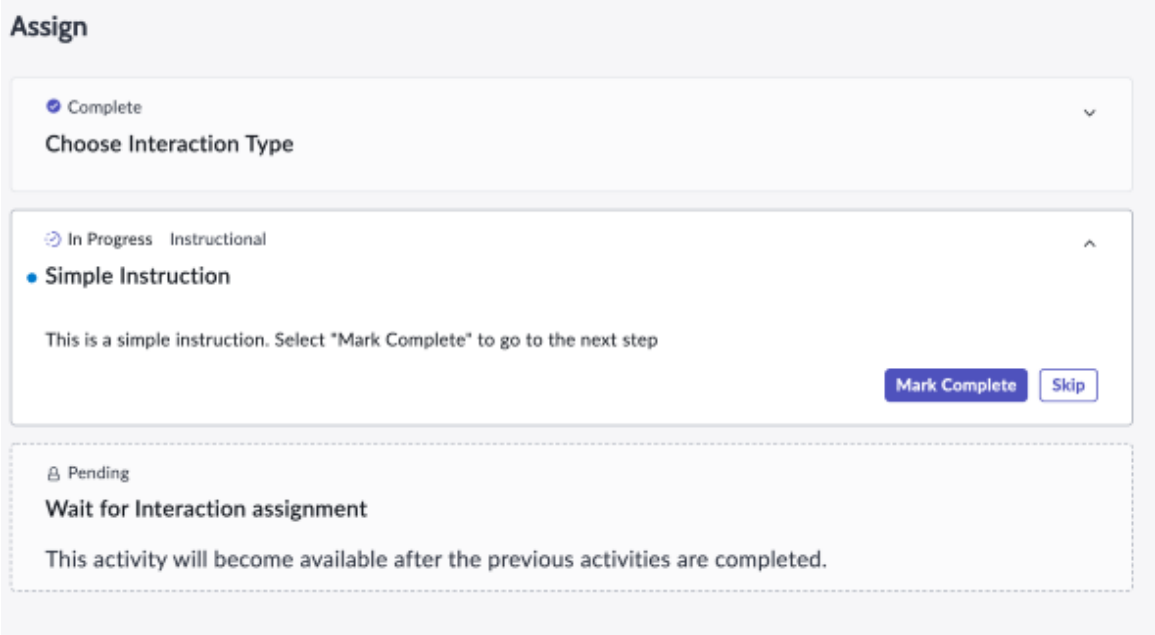
To toggle the **Show Stages flag** flag on or off, .

Playbook activity viewer

The activity viewer is where you manage a stage's activities. You can change the way activities are displayed in this space:

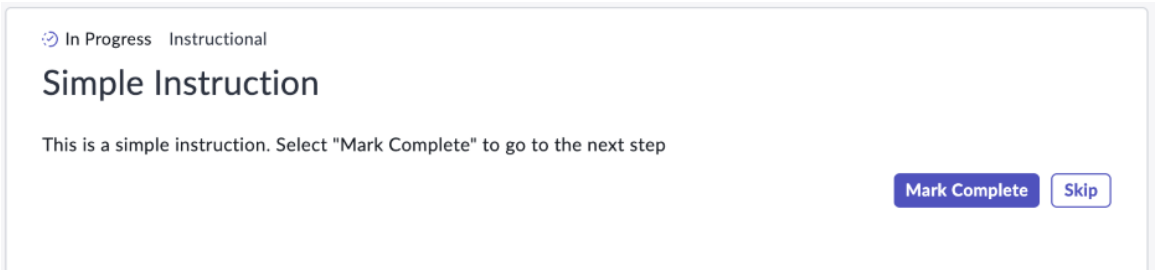
- The **Stacked** view displays all activities in a stage stacked on top of each other.

Playbook stacked view



- The **Focused** view displays a single selected activity.

Playbook focused view

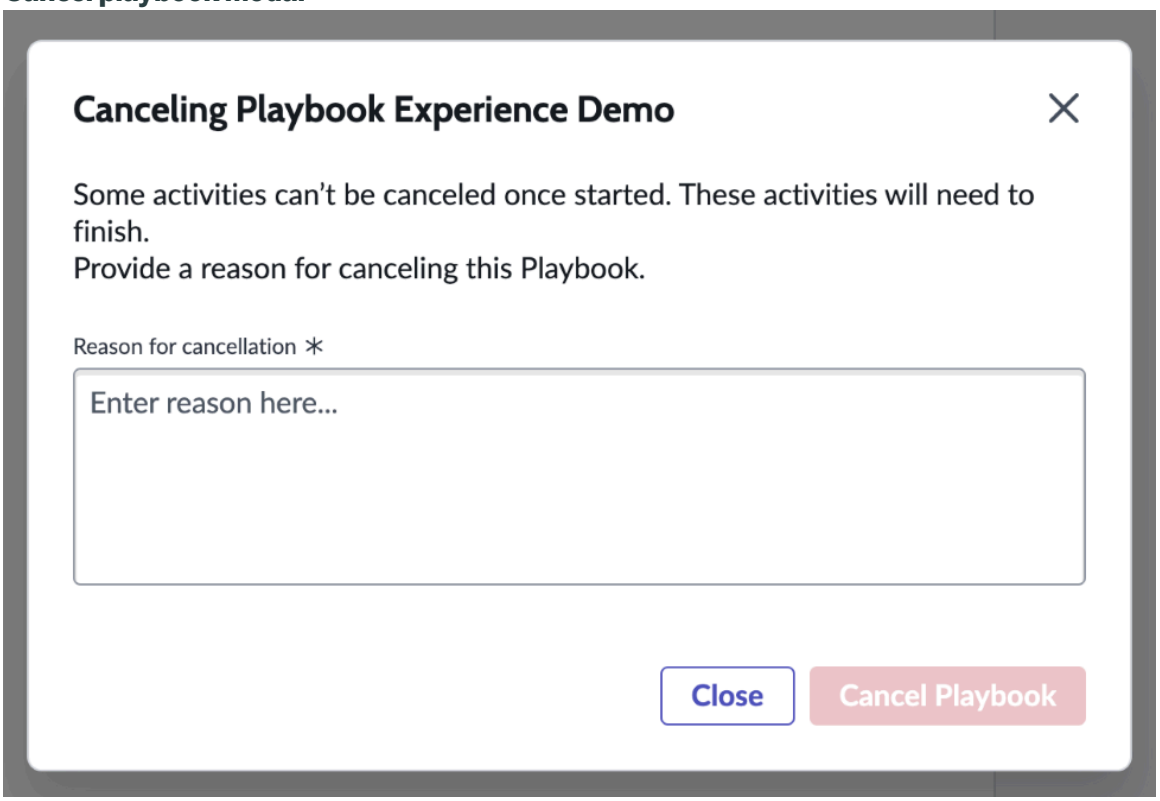


- The **Guided** view moves end users through a playbook, step-by-step. The sole focus in a Guided activity view is each activity. There are no stage pickers or filters as the end user moves through each activity of the playbook.

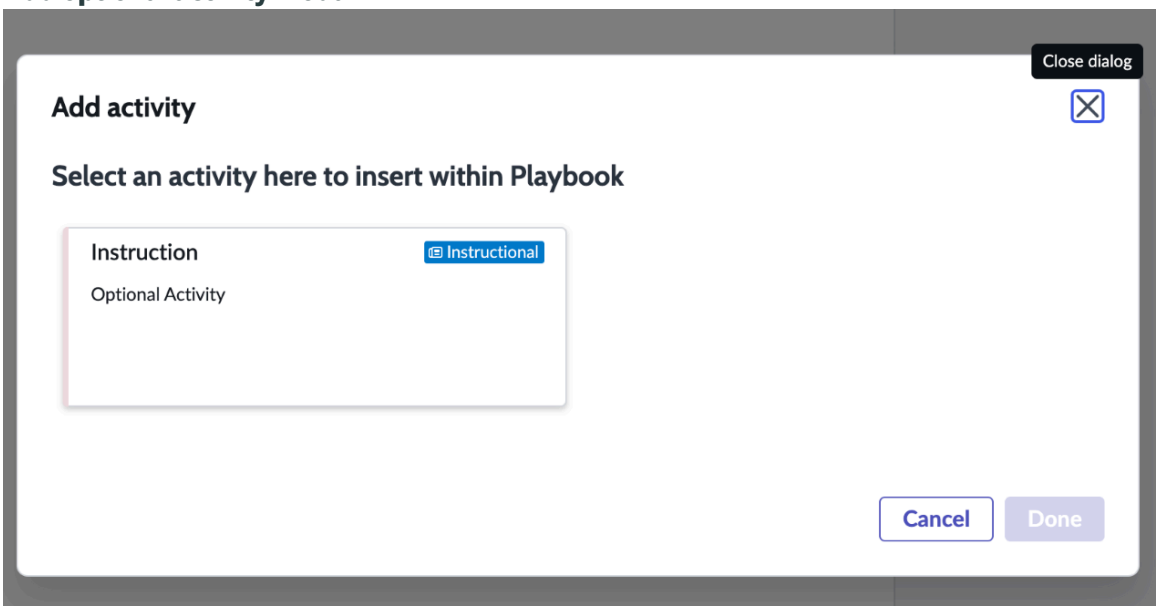
Playbook modals

If you want the ability to cancel playbooks and add optional activities, you must add the **Playbook Modals** component to your pages. If you use a template, this is already included.

Cancel playbook modal



Add optional activity modal



To start setting up components for your customized Playbook Experience, see Customize a playbook in UI Builder.

Provided UI Builder Bundles

If adding each component individually isn't ideal, you can also add an entire layout to a new or existing UI Builder template.

- Focused Vertical
- Focused Horizontal

- Stacked Vertical
- Stacked Vertical

Each layout contains the following components:

- UI controller
- Presets
- Activity viewer
- Activity picker
- Modals
- Playbook picker
- Stage picker
- Client scripts

To add a bundle to an existing or new UI Builder page, see Add a custom layout bundle to a UI Builder page. If you're creating a new experience and want to add a bundle to a standard record page, see Add a custom layout bundle to a UI Builder standard record page.

Customize a playbook in UI Builder

Use UI Builder pages and modular components for custom playbook layouts to customize the end user's Playbook Experience in a configurable workspace, Service Portal page, or mobile web page.

Before you begin

- If you don't have a playbook to apply custom layouts to, build a playbook. To learn more, see Building playbooks.
- Install the latest version of the Playbook Experience and Playbook Experience Component apps from the ServiceNow Store. See Playbook Experience apps.

Role required: ui_builder_admin, admin

About this task

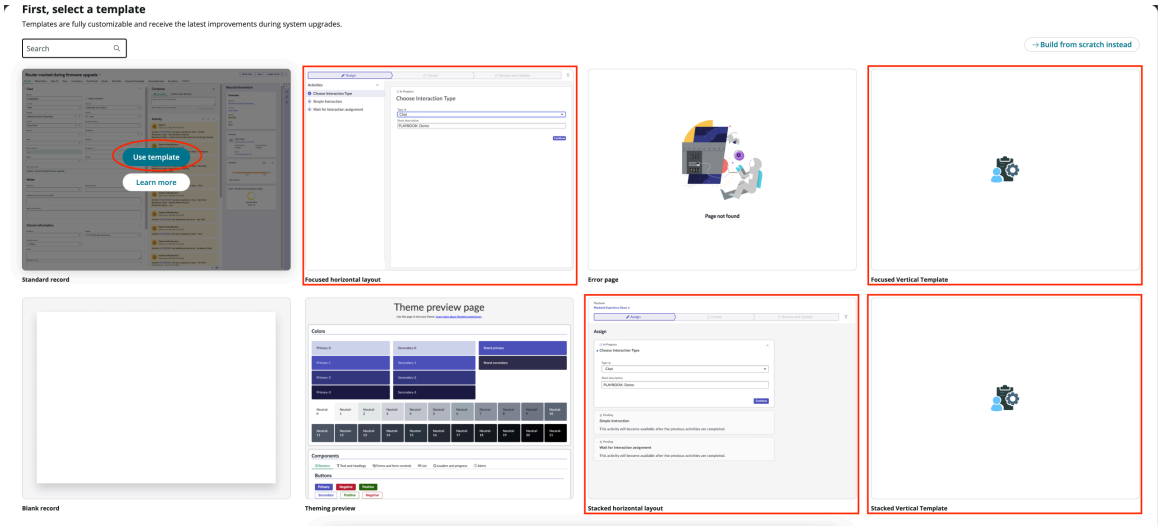
After creating a UI Builder page, you can customize Playbook Experience pages to your needs. Page templates include controllers that can be used with component presets, including the playbook templates. See Bind data to UI Builder pages using controllers (advanced feature) for more information.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. Open the **UI Builder** experience that you want to work in, or create a new experience.
   The **Playbook Experience Builder** experience was built for you to customize Playbook Experience. For more information on how to create an experience, see Configure how users interact with your applications in UI Builder.
3. In your **UI Builder** experience, you can:
   - Create a page from scratch,
   - create a page from a **Standard record** template,
   - or create a page from a Playbook Experience template.

Using a Playbook Experience template speeds up the development process, because the Playbook Custom Layout UI Controller and components are already added to every page, including record generators as of version 25.2. The controller automatically populates all the provided components with data.

Note:
You can still create and add custom components when you use a template.



4. If you're using a Playbook Experience template, **Create** your page.

   a. Set up the page details.

Name	The name of your page.
URL Path	The URL path that users navigate to, to access the page.

   b. Review the testing parameters for your page.

table	Name of the parent table for the playbook.
sysID	Sys_id of the record. For a record that doesn't exist, the value is **-1**.
experience	The experience you want to load.
selectedPlaybook	The playbook the agent or fulfiller is in.
selectedStage	The stage the agent or fulfiller is in.
selectedActivity	The activity the agent or fulfiller is in.

   c. Set up your default page variant.

Name	The name of your page variant.
Audiences	The users that can see your page variant. They can be defined by:

	▪ Role ▪ Group ▪ User ▪ Company ▪ Department ▪ Location ▪ Script
Conditions	Conditions that determine when the page variant is shown.

d. Open the default page variant you just created.

The new Playbook Experience layout already includes controllers and components populated with data, but there are a few more configurations to make. To make the remaining configurations, go to step 6.

5. If you used the **Standard record** template or created your page from scratch, add the Playbook Custom Layout UI Controller to the page.

   a. From the bottom corner of your new page, select the data icon ( )

   b. Select **+ Add**.

   c. Search for "Playbook".

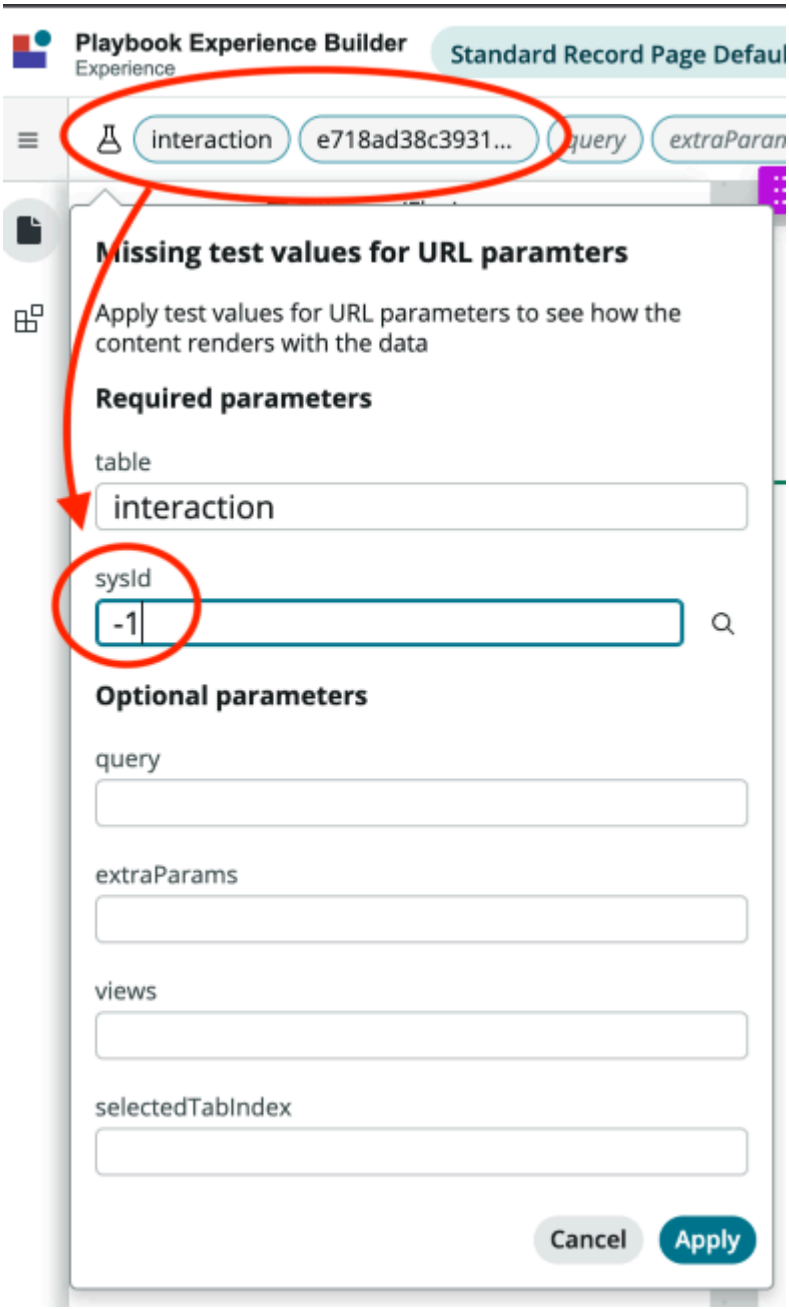
   d. Under **Data resources**, select **Playbook Custom Layout UI Controller** and click **Add**.

   e. Exit the **Data resources** panel.

6. Configure the Playbook Custom Layout UI Controller you just added.

   a. To allow users to create a new record in this customized Playbook Experience instead of the standard new record form, select a test value pill in the upper left.
   The test values popover opens.

   b. Make sure the **sysID** test value is set to **-1**, and select **Apply**.



A record generator form is available when you open the preview. You should be able to test the record generator form and confirm that new records are created.

c. You can also update the **table** name or any of the optional parameters:

d. Optional: If you want to hard-code the test values instead, navigate back to **Data** > **Data resources panel** > **Local data resource instances**, and select **UI Controller Record Page**.

e. Under the **Config** tab of the controller, add the **Parent SysID** or **Parent Table**. In the test values popover, these are the **sysID** and **table** fields.
You can enter **context.props.table** in the **Parent Table** field, and **context.props.sysId** for the **Parent SysID** field.

Note:
Make sure whatever table or record you enter has a playbook.

f. Add the **Playbook Experience** you want to use.

g. **Optional:** Enter values for the other fields, if needed.

Activity View Mode	View mode used to render Playbook activities in stacked or focused mode
Record Generator Query	Encoded Query string to optionally override the query provided for a record generator
Selected Playbook Context ID	Optional context ID of selected playbook for deep linking
Selected Stage Context ID	Optional context ID of selected stage for deep linking
Selected Activity Context ID	Optional context ID of selected activity for deep linking

h. **Optional:** In the Outputs column, you can select the values you want to hard-code.

The controller is configured and you're ready to add playbook components.

7. Add playbook components.

a. In the component tree, select the **Main Tab**.
The **Main Tab** panel opens on the right.

b. In the **Main Tab** panel, select **+ Add**.
A modal asking "How do you want to build this tab?" displays.

c. Select **Start from an empty container** and click **Next**.
A **Tab Settings** modal displays.

d. Give your tab a name and an icon, and click **Create**.
The tab is added to the component tree and a blank canvas displays.

e. In the component tree, under your new tab, select **+ Add component** and search for "resizable".

f. Select the **Resizable panes** component, give it a name, and click **Create**.

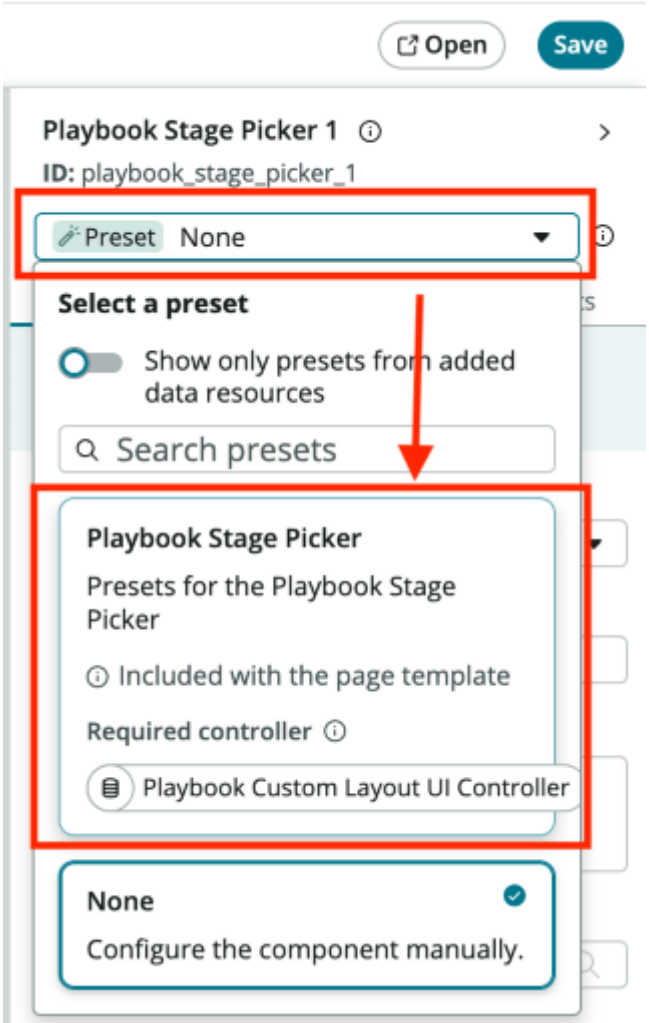
g. In the component tree, open the new **Resizable panes** component you just created.
The panel for your new **Resizable panes** opens on the right.

h. Configure as needed, and click **Save** in the upper right corner.

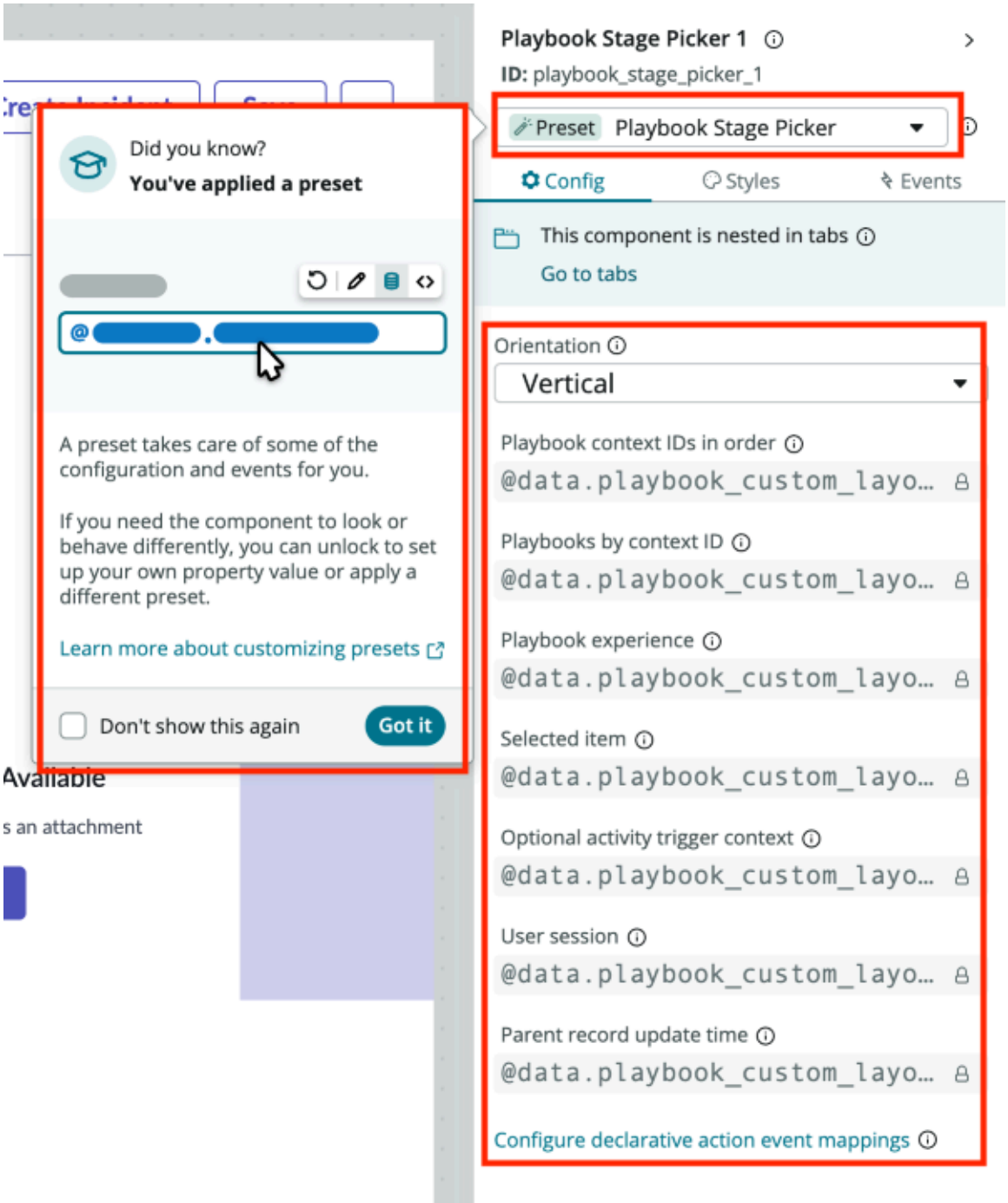
i. To add the stage picker, find your new **Resizable panes**component in the component tree, and select **+ Add component** under **left**.

j. In your component tree, select the new stage picker.
The panel for your new stage picker opens on the right.

k. Select the preset for this component from the Playbook Custom Layout UI Controller, and click **Save** in the upper right corner.



A confirmation message is displayed, and the component inputs and events under the **Events** tab in the stage picker panel is auto-populated.



l. To add the **Playbook Activity Viewer**, find your **Resizable panes**component in the component tree, and select **+ Add component** under **right**.

m. Repeat steps 7j and 7k.

n. To add the **Playbook Modals**, find the tab you created under the **Main Tab** in the component tree, and select **+ Add component**.

o. Repeat steps 7j and 7k for the **Playbook Modals**.

p. Navigate back to the **Playbook Stage Picker**, and determine the orientation of your stage picker.

q. In the upper right corner, click **Open** to preview the playbook layout.

r. Return to UI Builder.

s. Navigate back to your stage picker in the component tree, and determine the orientation of your stage picker in the panel that opens on the right.

Related topics

UI Builder

Add a custom layout bundle to a UI Builder page

Add playbook custom layout bundles to a new or existing UI Builder page that you've created from scratch.

Before you begin

Role required: admin, ui_builder_admin

About this task

Templates are page level, they are meant to be used when you create a completely new page. Bundles are like templates that you can add to a new or an existing page.

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder** and select **Playbook Experience Builder**.
   The **Playbook Experience Builder** experience was built for you to customize Playbook Experience. For more information on how to create an experience, see Configure how users interact with your applications in UI Builder.
2. Select **+** next to **Pages and variants** to create a new page.
3. Select **Create a new page**.

   The **Select a template** screen displays.
4. Select **Create from scratch instead** button at the top right.
5. Set up the page details.

Name	The name of your page.
URL Path	The URL path that users navigate to, to access the page.

6. Select **Continue**.
7. Add the test URL parameters for your page.

Note:
All playbooks require at least a table and sysId to be hardcoded on the controller or provided through a URL.

table	Name of the parent table for the playbook.

sysID	Sys_id of the record. For a record that doesn't exist, the value is **-1**.

8. Setup your default page variant.

Name	The name of your page variant.
Audiences	The users that can see your page variant. They can be defined by: ○ Role ○ Group ○ User ○ Company ○ Department ○ Location ○ Script
Conditions	Conditions that determine when the page variant is shown.

The page and default variant are created.

9. Open your new page variant.
10. In the component tree, select **+ Add component** under **Body**.
The toolbox opens.
11. Enter "bundle" in the search bar of the toolbox.
Available bundles are displayed.
12. Select the bundle you want to add to your page.

Result
You've added a bundle and all its components, presets, and controller to your new UI Builder page.

What to do next
Edit test values and controller inputs.

Add a custom layout bundle to a UI Builder standard record page

Add playbook custom layout bundles to a new or an existing UI Builder standard record page.

Before you begin
Role required: admin, ui_builder_admin

Procedure

1. Navigate to **All** > **Now Experience Framework** > **UI Builder**.
2. In the upper right, select **+Create** to create a new experience.
3. Fill in the fields for your experience.

Option	Description
Name	The name is used for tracking the experience internally. The experience name is also visi ble to users from the browser tab.
URL path	The URL path is the prefix that people will use to reach your experience. It matches your ex perience name by default. You can edit this path now or later in UI Builder. Supported characters include digits (0-9), let ters (A-Z, a-z), and the following symbols: $-_! *(), The path can't end with a forward slash (/).
App shell UI	Choose the **UXR Base Experience Shell**. The app shell is the wrapper of the page con tents, 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. To learn more about the app shell options, see Define UI experiences
Landing path	The landing path is the prefix that people will use to reach your experience homepage. It matches your experience name by default. To designate a page as the homepage, you will need to create a page that has a matching path.
Roles	Only users with these assigned roles can ac cess the experience. If you leave this field empty, the experience is open to all logged-in users by default.

4. Select **Create**.
5. In your new UI Builder experience, select **Create new page**.

   The **Select a template** screen displays.
6. Choose the **Standard record** template and select **Use template**.
7. Set up the page details.

Name	The name of your page.
URL Path	The URL path that users navigate to, to access the page.

8. Select **Continue**.
9. Add the test URL parameters for your page.

Note:
All playbooks require at least a table and sysId to be hardcoded on the controller or provided through a URL.

Required parameters

table	Name of the parent table for the playbook.
sysID	Sys_id of the record. For a record that doesn't exist, the value is **-1**.

Optional parameters

query	Encoded Query string to optionally override the query provided for a record generator.
extraParams	Pass additional parameters to a page.
views	View mode used to render Playbook activities in stacked or focused mode.
selectedTabIndex	The tab that your page opens to by default.

10. Setup your default page variant.

Name	The name of your page variant.
Audiences	The users that can see your page variant. They can be defined by: ○ Role ○ Group ○ User ○ Company ○ Department ○ Location ○ Script
Conditions	Conditions that determine when the page variant is shown.

The page and default variant are created.

11. In the success message, select **Open in editor** to open your new page in the UIB editor.
12. In the component tree, select the **Main Tab**.
The **Main Tab** panel opens on the right.
13. Under the **Tabs** section in the **Config** tab, select **+ Add**.
A modal asking "How do you want to build this tab?" displays.
14. Select **Start from an empty container** and click **Next**.
A **Tab Settings** modal displays.
15. Give your tab a name and an icon, and click **Create**.
The tab is added to the component tree and a blank canvas displays.
16. In the component tree, under your new tab, select **+ Add component**.

The toolbox opens.

17. Enter "bundle" in the search bar of the toolbox.
    Available bundles are displayed.
18. Select the bundle you want to add to your page.

Result

You've added a bundle and all its components, presets, and controller to your new UI Builder standard record page.

Trouble?

If the width of the bundle container is set to the min-width of the component, set the **min-width** of the container to **100%**.

Alternatively, you can set the **flex-direction** of your new bundle tab to **column**.

What to do next

Edit test values and controller inputs.

Enable Guided activity view

Guide end users through a playbook, step-by-step.

Before you begin

- Role required: admin or playbook.admin
- Familiarize yourself with how to begin customizing the playbook experience in UI Builder. To learn more, see Customize the Playbook Experience.

About this task

Guide end users through a playbook, one activity at a time. If you have a lot of end users encountering issues in a particular area, and you want to walk them through a series of questions that lead to a particular Instruction activity to complete work (or even KB activity for final instruction), consider using the **Guided** activity view. There are out-of-the-box playbooks that use the **Guided** activity view for employee self-service requests, such as time off or benefit requests, available through the Employee Center.

Note:

The following configurations and features are not supported for the **Guided** activity view:

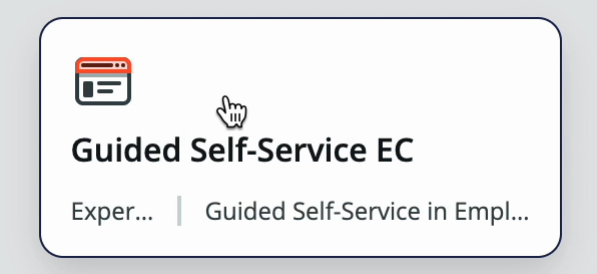
- Parallel activities
- Optional activities
- The **Pending Item Visibility** configuration in the **Guided Self-Service Experience** playbook experience record in the **Playbook Experiences** list [sys_playbook_experience.list]

  Note:
  This property is not relevant since there are no pending activities other than the current activity that an end user is in during a **Guided** self-service playbook run.

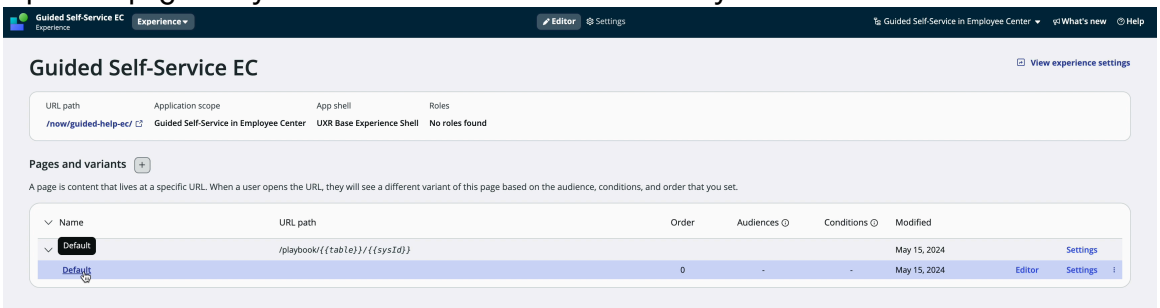
- Stage picker components, including filters

Procedure

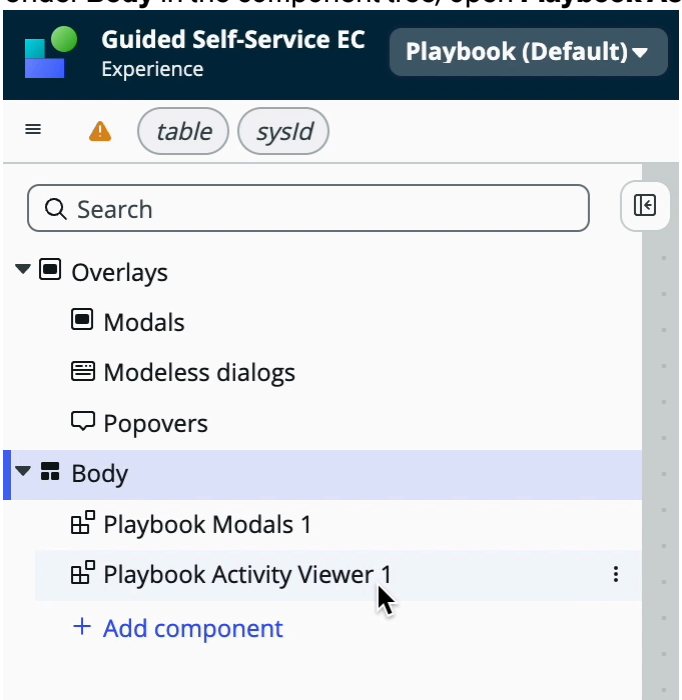
1. Navigate to **All** > **UI Builder**, and open the **Guided Self-Service in Employee Center** experience.



2. Open the page that you want to enable the Guided activity view for.

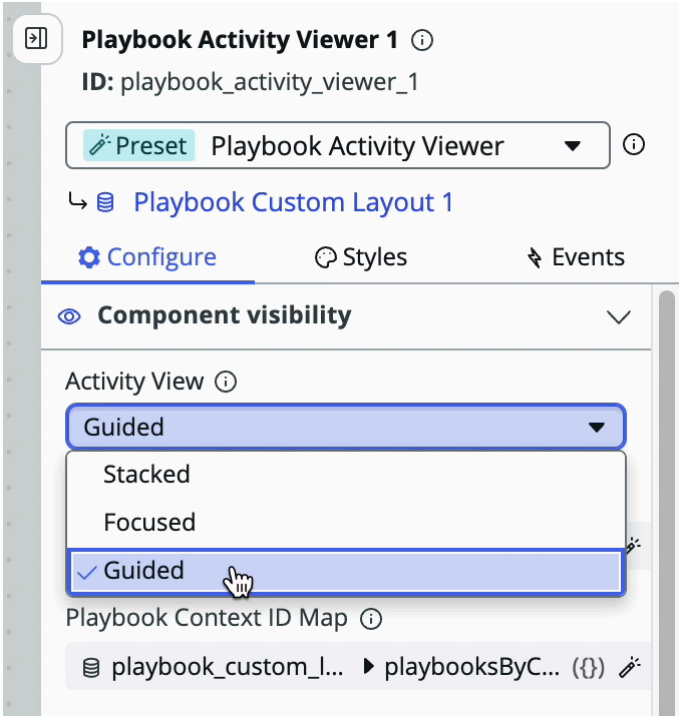


3. Under **Body** in the component tree, open **Playbook Activity Viewer**.



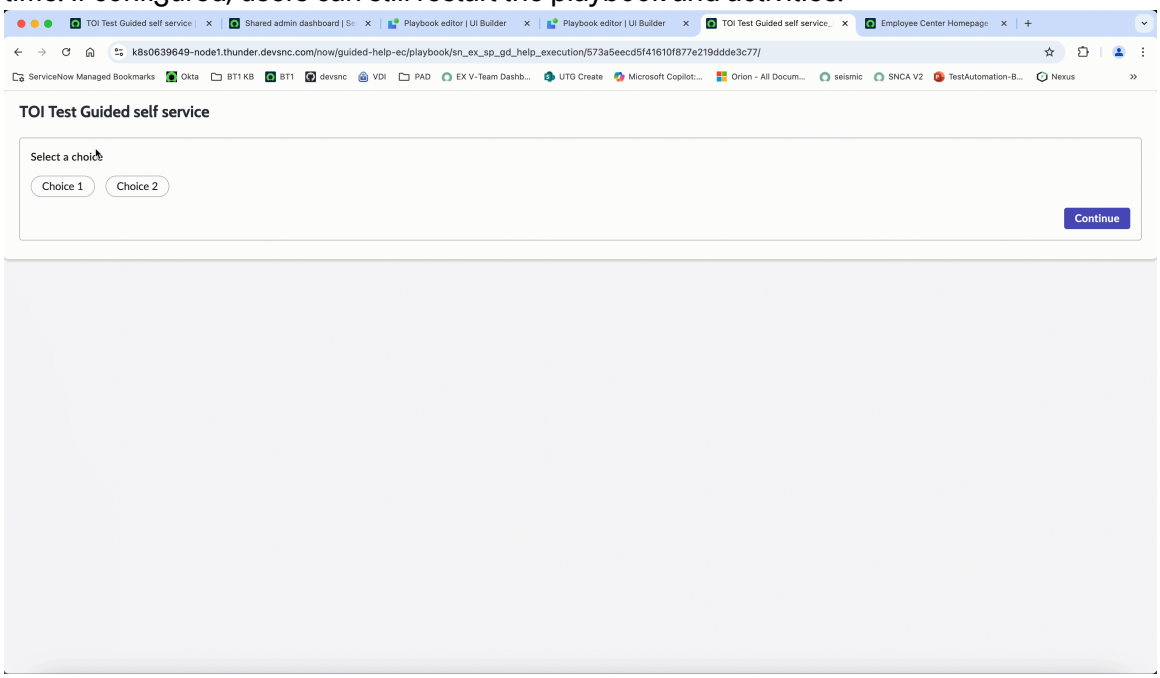
The **Playbook Activity Viewer** panel opens on the right.

4. In the **Playbook Activity Viewer** panel, under the **Configure** tab, open the **Activity View** drop-down and select **Guided**.



Result

For an end user who is running a playbook, the result is a playbook that runs one activity at a time. If configured, users can still restart the playbook and activities.



Playbooks in Service Portal

Use playbooks to guide Service Portal users through your business processes.

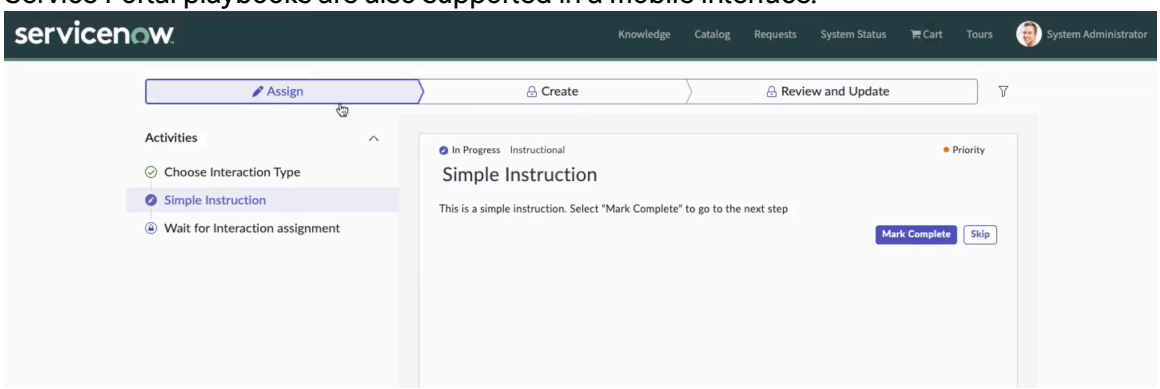
Using the widget

The Portal Playbook widget adds playbooks that your Service Portal users can run.

Provide a playbook experience to your Service Portal users for a consumer grade user experience. Requests on portals are surfaced as catalog items, including any playbooks you make available in Service Portal. Embedding a playbook in Service Portal enables:

- A rich user interface for activities during a business process
- Guidance for long business processes
- Going back and editing any previous steps of a business process
- Saving long running playbooks for a business process and resuming at a later point
- Seamless transition for agents and teams to enter a business process
- Visibility into where you are in a business process
- Visibility on a specific user's tasks
- Single admin experience for a whole business process.

Service Portal playbooks are also supported in a mobile interface.



Use cases

Business account onboarding

Onboarding a new or existing business customer with one or more contacts along with onboarding the services requested such as wire transfers, treasury services. Promote compliance and deliver a faster time to value.

Loan or mortgage application

Intake of a new credit card application or loan application for both consumers and corporate programs. Customer needs the ability to save the application and come back at a loter time to resume application.

Insurance claims

Policy holder submitting a new personal auto or home claim for an auto accident or home damage via portal.

Request quote for a new policy

Business customer requesting a quote for small business insurance policy that includes liability and commercial property insurance The customer can start filling in the request and the agent needs the abilith to fill in the remaining on behalf of the customer.

License request

Constituents requesting a new license such as a business license, or personal license (fishing, driving or hunting) from a government agency via a government portal. These applications typically vary by service and state and constituents need to be guided through request.

Customer cases

Cases logged by users who need guidance to select the correct issue and enter all the necessary details, such as instances, steps to reproduce, best time to contact, etc. Enable faster case resolution.

Life cycle

The general life cycle of a Service Portal playbook can be summarized as:

- A business process administrator builds the playbook in Workflow Studio. To learn more about building a playbook in Workflow Studio, see Building playbooks.
- An administrator adds the playbook to a Service Portal page and customizes the playbook runtime experience at the same time through the ServiceNow AI Platform®. To learn more about embedding a playbook in Service Portal and further customizations in the ServiceNow AI Platform®, see Embed the playbook in Service Portal.
- Service Portal administrators and developers customize the playbook portal page. To learn more about using the Service Portal Designer to customize the playbook portal page, see Customize the playbook in Service Portal.
- Service Portal end users, also called requesters, launch and run through the playbook. To learn more about launching and running a playbook in Service Portal, see Run a playbook in Service Portal.

Set up mappings between themes

Make sure that playbooks run as expected in your Service Portal by setting up mappings between your Service Portal theme and the UXF theme used in UI Builder.

Before you begin

Role required: admin, or portal admin with either snc_internal or snc_external

About this task

The Portal Playbook widget uses a system property **ux_portal_theme_to_uib_theme_mapping** to handle mappings between a Portal page theme and a UXF theme.

Procedure

1. Navigate to **All** > **System Properties** > **All Properties**.
2. Select **New** in the upper right corner.
3. Enter values for the following fields.
4. In the form context menu, select **Save**.

Result

The mapping between your Service Portal theme and the UXF theme is saved. Launch the playbook in Service Portal to see your changes.

Note:
To turn a mapping off, you must delete it.

What to do next

Embed a playbook in a Service Portal page and customize the Service Portal playbook runtime experience in the ServiceNow AI Platform®.

To learn more about customizing the playbook runtime experience in Service Portal, see Embed the playbook in Service Portal.

Embed the playbook in Service Portal

Add a playbook that Service Portal users can launch and run by creating a Playbook Content Item.

Before you begin

- Role required: admin, or portal admin
- If you don't have a playbook that you want to customize for Service Portal yet, build one in Workflow Studio. To learn more about building a playbook in Workflow Studio, see Building playbooks.
- Create a record generator, if there isn't already one you want to use. To learn more about creating a playbook record generator, see Playbook record generator.
- You may need to set up mappings between your Portal theme and the UXF theme, if they are very different. To set up mappings between your Portal theme and the UXF theme, see Set up mappings between themes.

About this task

In a Playbook content item, the admin specifies the URL that points to the portal page.

Procedure

Navigate to **All** > **Playbook Experience** > **Playbook Content Items**.

Playbook content item form view

Field	Description
Name	Enter a name for the playbook.
Active	Uncheck this box if you don't want users to be able to run the playbook in Service Portal.
Catalogs	Enter the catalog that you want this playbook to be added to. There is no default, but the playbook is not searchable if you don't add it to a catalog.
Category	Enter the catalog category that you want this playbook to be in. There is no default, but the playbook is not searchable if you don't pick a category for the playbook.
Icon	Upload a small image that appears next to the name when the item is displayed in the service catalog. Supported file types are jpg, png, bmp, gif, and jpeg.
Application	Choose an application scope that you want your Service Portal playbook to run in. Selecting **Global** lets your playbook run in any application scope. For more information, see Application scope.
Picture	Upload an image showing the item. Supported file types are jpg, png, bmp, gif, and jpeg.
Short description	Enter some descriptive details about your playbook.

Field	Description
Table	Enter the parent table that the playbook opens on. If not specified, defaults to the Incident table.
Record ID	Enter the parent Sys ID of the record you want to open in the record generator table. If you want to use the record generator, leave it as -1. If not specified, defaults to -1.
Playbook Experience	Choose the playbook experience record with the playbook configurations that you want for your Service Portal users. If not specified, defaults to the Global Playbook Experience.
Playbook Experience Record Generator	Choose the record generator with the playbook that you want to show.
Portal Page	Enter a Service Portal page that you want to open the playbook in. If not specified, the playbook is launched in the out-of-the-box Playbook Portal page.
Title	Add a title for any components that use the title property in the Portal Playbook widget's UI Builder page.
Meta	Add tags to the playbook item for Service Portal home screen searches.

Trouble?

For issues that seem like process execution related, check the execution logs on the sys_pd_context that is related to the process that was run. You can find these in the "today's executions" module within the navigator.

For issues that are flow designer related, debug using the flow debugger options. To learn more about flow debugger options, see [an article linked to debugging flows].

For other issues:

- Check the instance logs around the execution time of the sys_pd_context records and see if you can find any errors that seem related (record watchers, flow exceptions, or any other exceptions seeming relevant).
- Make sure versions of playbook are as follows:
  - now-playbook-experience version 25.1.4 or greater
  - playbook-experience version 25.1.2 or greater
- Verify that you are using the out-of-the-box Portal Playbook widget.

What to do next

Customize the runtime playbook experience in the Service Portal Designer.

Portal Playbook Widget

Explore the Portal Playbook Widget.

About the widget

The Playbook widget is an iFrame that enables administrators to specify where a playbook event should be directed. The iFrame URL is the UI Builder Playbooks Portal page. The widget handles events that playbook is listening for via the Session Storage API so that it knows when to perform an action in portal, like opening a record or list in a modal within Service Portal.

The out-of-the-box components are built for any Playbook Experience you may need for your Service Portal users. We do not recommend directly editing the out-of-the-box UI Builder Playbook Portal Page, the Service Portal Playbook widget, or the Playbook Content Item. Changing the out-of-the-box components can result in technical problems.

If, for example, you need a UXF Client Action to work for your instance of the playbook page, we recommend cloning the playbook widget instead.

Note:
To learn more about how to clone or create your own widget instead, see Developing custom widgets.

Cloning a Playbook Service Portal Widget

Navigate to **All** > **Service Portal** > **Widgets** and find the **Playbook** widget.

Note:
If you clone the **Playbook** widget, make sure that all out-of-the-box actions and configuration properties are copied over to your cloned widget.

Form Fields

Field	Description
Name	Enter a name for the cloned widget.
ID	The widget ID is created automatically based on the widget name by default, but you can change it to whatever you want.
Description	Add an optional description that provides details for the widget does.
Application	Choose an application scope that you want your widget to run in. Selecting **Global** lets your playbook run in any application scope. For more information, see Application scope.
Public	Select if your widget is public. If unchecked, your widget is private and only authenticated users with the snc_internal or snc_external role can see the widget.
Roles	Restrict access to the widget to certain roles.
Body HTML template	Leverage the Angular JS two-way binding to bind your controller variables to your markup. **Danger:** Only make changes to HTML templates if you have advanced coding knowledge and a firm understanding of AngularJS and the platform API.

Form Fields (continued)

Field	Description
	The iFrame URL is the URL of the UI Builder page. To learn more about the HTML template field, see . and . **Warning:** Make sure the iFrame URL of your cloned widget is different from the iFrame URL for the out-of-the-box widget.
CSS	Configure the widget CSS. Configuring CSS in an actual widget affects all instances of that widget. To learn more about the CSS field, see . and .
Server script	Script the server-side logic. This is helpful primarily with interacting with the Glide platform through server-side APIs. **Danger:** A server script requires knowledge of the ServiceNow API to work with record data. To learn more about the Server script field, see . and .
Client controller	In Angular, HTML templates contain Angular-specific elements and attributes. Angular combines the template with information from the model and client controller to render the dynamic view that a user sees in the browser. Identifier name for a reference to the controller in the directive's scope **Danger:** A client script requires knowledge of both the ServiceNow API and AngularJS to create a client controller.
controllerAs	The HTML template uses the controllerAs syntax for basic binding.
Link	Use a link function to directly manipulate the DOM. **Danger:** The link function requires knowledge of AngularJS. To learn more about the Link field, see . and .
Has preview	Select the checkbox to enable a preview of the widget in the Widget Editor.

Form Fields (continued)

Field	Description
Demo data	Provide data when previewing the widget in the widget editor.
Data table	Select a table to use as a data source.
Fields	Select fields to display as instance options.
Option schema	Allows a Service Portal admin to configure a widget. To learn more about the Option schema field, see . and Widget option schema .
Docs	Select a Service Portal documentation link.

Check for errors in the UI Builder Playbook Experience Portal page config and events properties, e.g. disable the Open Record event as a maint user.

Customize the playbook in Service Portal

Use the Service Portal Designer to customize the playbook runtime experience for Service Portal users.

Before you begin

- Role required: admin, or portal admin with either snc_internal or snc_external
- Create a record generator if you don't want to use the default record generator or any of the other existing record generators.

> **Note:**
> A key difference in Service Portal is that the playbook record generator is used instead of the record producer. The Service Portal requester enters the information for the record generator and uses the Next or Continue declarative action to move to the next activity.

To learn more about playbook record generators, see .

Procedure

1. To edit your playbook widget in the Service Portal Designer, navigate to **All** > **Service Portal** > **Service Portal Configuration**.
2. Select **Page Editor**.
3. Search for and select your playbook.
4. Select **Edit [Playbook] (playbook) page in Designer**.
5. Select the widget, and select the edit icon.

Data parameters

Field	Description
Playbook UIB Page URL	Enter a URL for the UI Builder page. If not specified, this points to the default **Playbook Experience Portal** page.

Field	Description
Open Record Widget	The Sys ID for the widget to use when a user opens a record. If not specified, the form widget is used.
Open Record is Pop Up	If enabled, shows the reference picker inside the form.
Open List Widget	The Sys ID for the widget to use when a user opens a list. If not specified, the Data Table from the Instance Definition is used.
Extra Params	Pass arbitrary data from the widget to its UI Builder page.

Presentation parameters

Field	Description
Height	Enter the height of the playbook. If not specified, height is set to 850px.
Open Record Modal Title	Add a title to the modal that a user sees whenever they open a record. If not specified, there is no title.
Open Record Modal Message	Add a message to the modal that a user sees whenever they open a record. If not specified, there is no message.
Open Record Modal Buttons	Add the buttons to the modal that a user sees whenever they open a record. If not specified, there are no buttons. Enter in JSON format.
Open Record Modal View	The form view that you want to show for the open record.
Open Record Modal Size	Choose the size of the modal that a user sees whenever they open a record. Choice are: ○ sm - small ○ md - medium ○ lg - large If not specified, defaults to large.
Open List Modal Title	Add a title to the modal that a user sees whenever they open a list. If not specified, there is no title.
Open List Modal Message	Add a message to the modal that a user sees whenever they open a list. If not specified, there is no message.
Open List Modal Buttons	Add the buttons to the modal that a user sees whenever they open a list. If not specified, there are no buttons. Enter in JSON format.
Open List Modal View	The form view that you want to show for the open list.

Field	Description
Open List Modal Size	Choose the size of the modal that a user sees whenever they open a list. Choice are: ○ sm - small ○ md - medium ○ lg - large If not specified, defaults to large.
Activity View	Change the activity view between stacked or focused.
Playbook Experience ID	Enter the playbook experience record ID with the playbook configurations that you want for your Service Portal users. When nothing is set in the URL during the playbook run, this is the default playbook experience. If nothing is specified here, defaults to the Global Playbook Experience.
Open Record Modal Prevent URL Update on Submit	Prevents the URL of a running playbook in Service Portal from updating whenever a new record is submitted from a form in a record modal.

Trouble?

Verify that you are using the official Playbook Widget for the Portal Page.

Run a playbook in Service Portal

Launch a playbook as a Service Portal requestor.

Before you begin

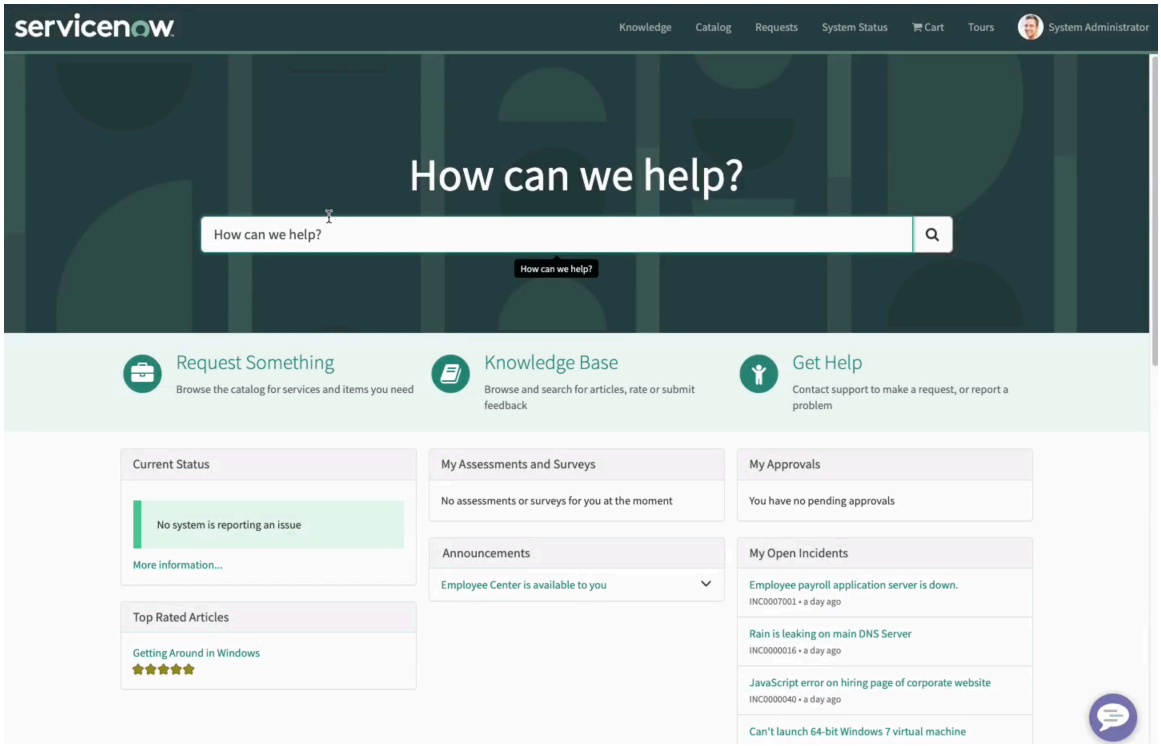
Role required: snc_internal or snc_external

About this task

Keep track of where you're at in an overall process more easily, and pause and resume the playbook as needed.

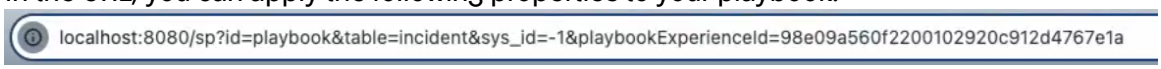
Procedure

1. Navigate to your Service Portal instance.
2. In the home screen, search for the playbook and select it.



The playbook launches.

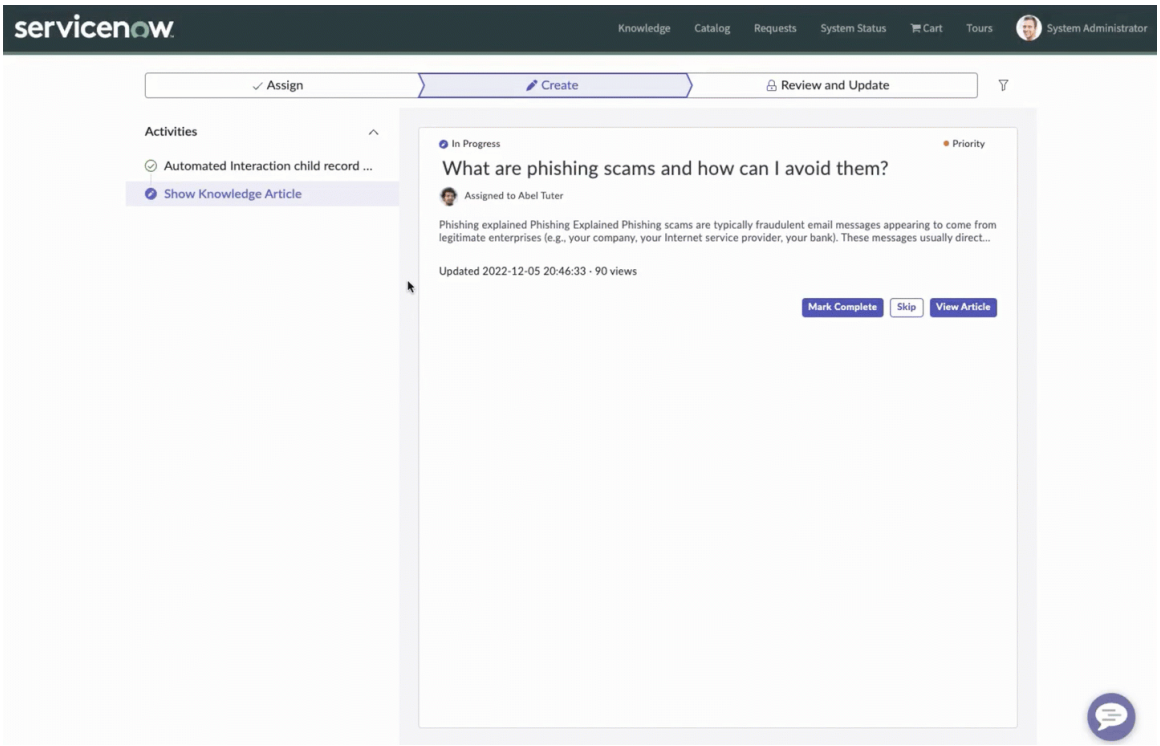
3. **Optional:**
   In the URL, you can apply the following properties to your playbook.



Property	Description	Example
layoutType	Set the playbook to a custom layout. If not specified, defaults to the default (standalone) layout.	layoutType=custom
table	Enter the parent table that the playbook opens on. If not specified, defaults to the Incident table.	table=incident
sys_id	Enter the parent Sys ID of the record you want to open in the record generator table. If you want to use the record generator, leave it as -1. If you leave it as -1, the value changes once the record is generated.	sys_id=-1
playbookExperienceid	Enter the playbook experience record ID with the playbook configurations that you want for your Service Portal users. If not specified, defaults to the Global Playbook Experience,	playbookExperienceid=9809a560f220010292

Property	Description	Example
	which has a record ID of 9809a560f2200102920c912d4767e1a.	

4. As you go through the playbook, you can open records and lists in a modal without needing to leave Service Portal, such as when you view a Knowledge Base article or a user record via the information icon.



Trouble?

Errors

Error	Description
404: The page you are looking for could not be found	The table or sysId is missing from the URL or the URL is incorrect.
Configuration Error	The process cannot be found by the playbook. For example the table is interaction, but the Sys ID does not exists for this table.
There are not playbooks available	There is a valid record for the table, but the record does not satisfy the conditions to show the playbook. For example an interaction record might need to be of type Chat and start with the short description of PLAYBOOK: to kick off a playbook.
No Activities Available	Here are not activities to view for this particular stage of playbook. Activities can be hidden based on user role or Run Conditions from the process.

Errors (continued)

Error	Description
No filtered results	When using the playbook filter, no results can be shown for a particular stage.

For client-side issues, check local browser console.logs which might show a runtime issue using the browser's inspector.

Configure a playbook for ServiceNow® mobile

Configuring a playbook for ServiceNow® mobile is exactly the same as in a configurable workspace, but with an additional step for embedding the playbook.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **Workspace Experience** > **Actions & Components** > **Related Items** or **Contextual Side Panel**.
2. Click **New**.
3. On the form, fill in the fields.

Action Assignment form

Field	Description
Action label	Label of the Related Item or Contextual Side panel tab.
Action name	Unique name for your item. This name can be overridden.
Implemented as	Select **UI Component**.
Specify UI component	UI component associated with the action. Enter `now-playbook-experience`.
Icon	Icon that is displayed in the Contextual side panel to differentiate from other components. **Note:** This field is available for the Contextual Side Panel only.
Application	Application for the action assignment.
Workspace	Option to limit a Related Item or Contextual Side Panel to a specific Workspace. For example, Agent Workspace or HR Workspace.
Table	Select the table you want to show the Related Item or Contextual Side Panel on.
View	Field to only display a Playbook when this Form View is selected on the parent record.
Active	Option to activate the action assignment.

Field	Description
Order	Integer that determines the precedence of this action in relation to matching actions with the same name. The lower the number, the more likely it is to be selected against other actions. The typical practice is to use numbers that are in the hundreds. For example, 100, 200, 300, or 400.
Tooltip	Message that's displayed when your mouse points to the Related Item tab or Contextual Side Panel icon.
Description	Description for the action assignment. This description is displayed in the Action Assignment list and provides context on the form.

4. Click the **Advanced View** related link.
5. Click the **Component Attributes** tab.
6. On the form, fill in the fields.

Component Attributes form

Attribute name	Description
playbookExperienceId	Associated Playbook experience ID. Copy and paste the sys_id of a Playbook Experience record. **Note:** If no Playbook experience ID is provided, the global Playbook experience is used by default.
parentSysId	Associated parent sys_id. Enter `{{sysId}}` to automatically take the parentSysId of the record that you're viewing.
parentTable	Associated parent table. Enter `{{table}}` to automatically take the parentSysId of the record that you're viewing.
compactMode	Option to display a Playbook in compact mode. Typically set to true for Contextual Side Panel and false for Related Item.
recordGeneratorQuery	Not currently supported in Agent Workspace.
isNewParentRecord	Set to `{{isNewRecord}}`.

Note:
Select a different **UI Component** if the attributes are missing from your form. Change the **UI Component** back to **now-playbook-experience** and the attributes will appear.

7. Select **Update**.

8. Click the **Conditions** tab.
9. On the form, fill in the fields.

Conditions form

Field	Description
Script Condition	Script condition for the action assignment. Enter `sn_playbook.PlaybookExperience.parentRecor` The script condition enables you to show a Playbook only when the record has triggered a process execution.
Client Conditions	Choose conditions to limit collisions based on your use case.
Record Conditions	Choose conditions to limit collisions based on your use case.
Required Roles	Roles to limit Playbook access.
Requires create access	Option to require create access.
Requires read access	Option to require read access.
Requires write access	Option to require write access.
Requires delete access	Option to require delete access.

10. Select **Update**.

Result
The playbook is configured for ServiceNow® mobile.

What to do next
Embed your playbook in ServiceNow® mobile. To learn more, see

Embed a playbook in ServiceNow® mobile

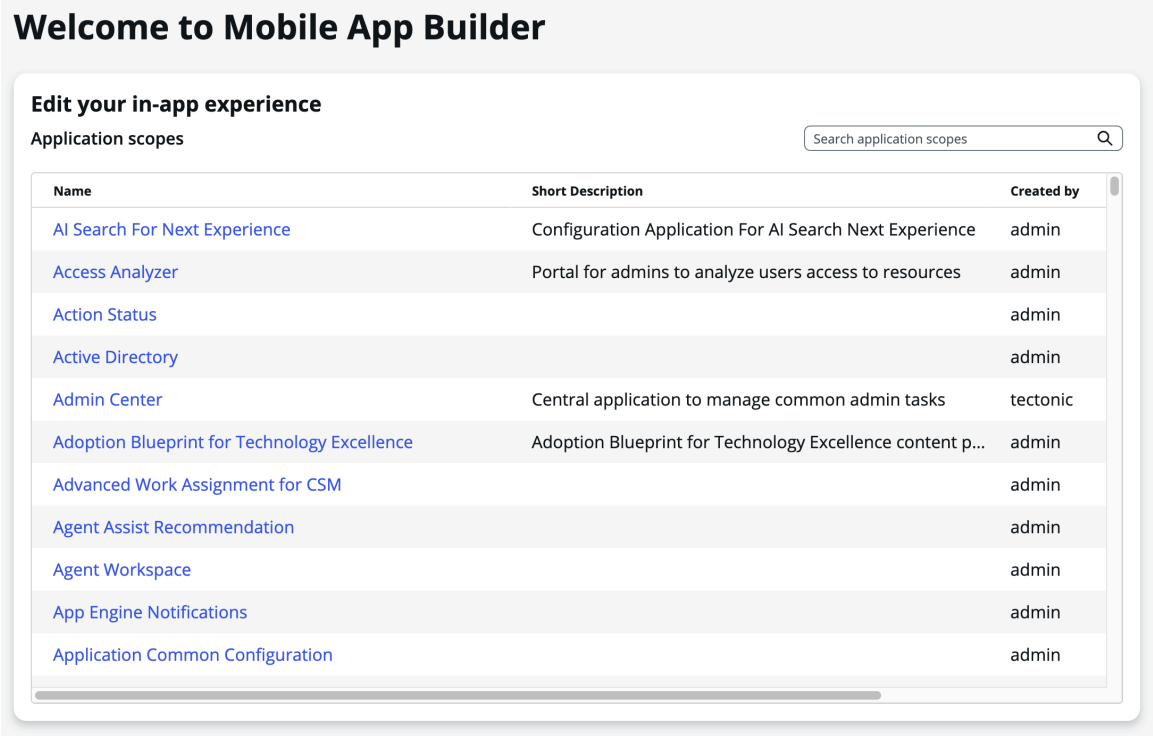
Embed a playbook in ServiceNow® mobile by creating a screen in Mobile App Builder.

Before you begin
Role required: admin

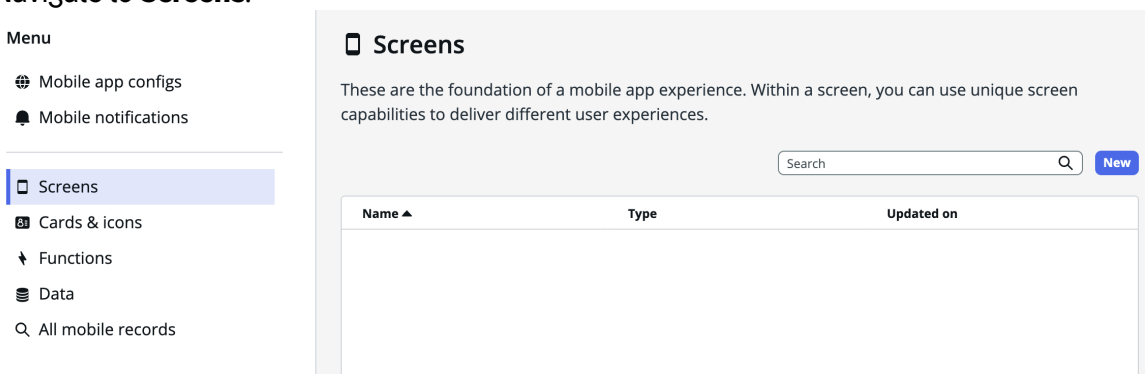
If you haven't configured your playbook for ServiceNow® mobile in UI Builder yet, see Configure a playbook for ServiceNow mobile.

Procedure
Create a playbook screen

1. Navigate to **All** > **System Mobile** > **Mobile App Builder**.
2. Search for and select the scope of the application that you want to create a screen for.

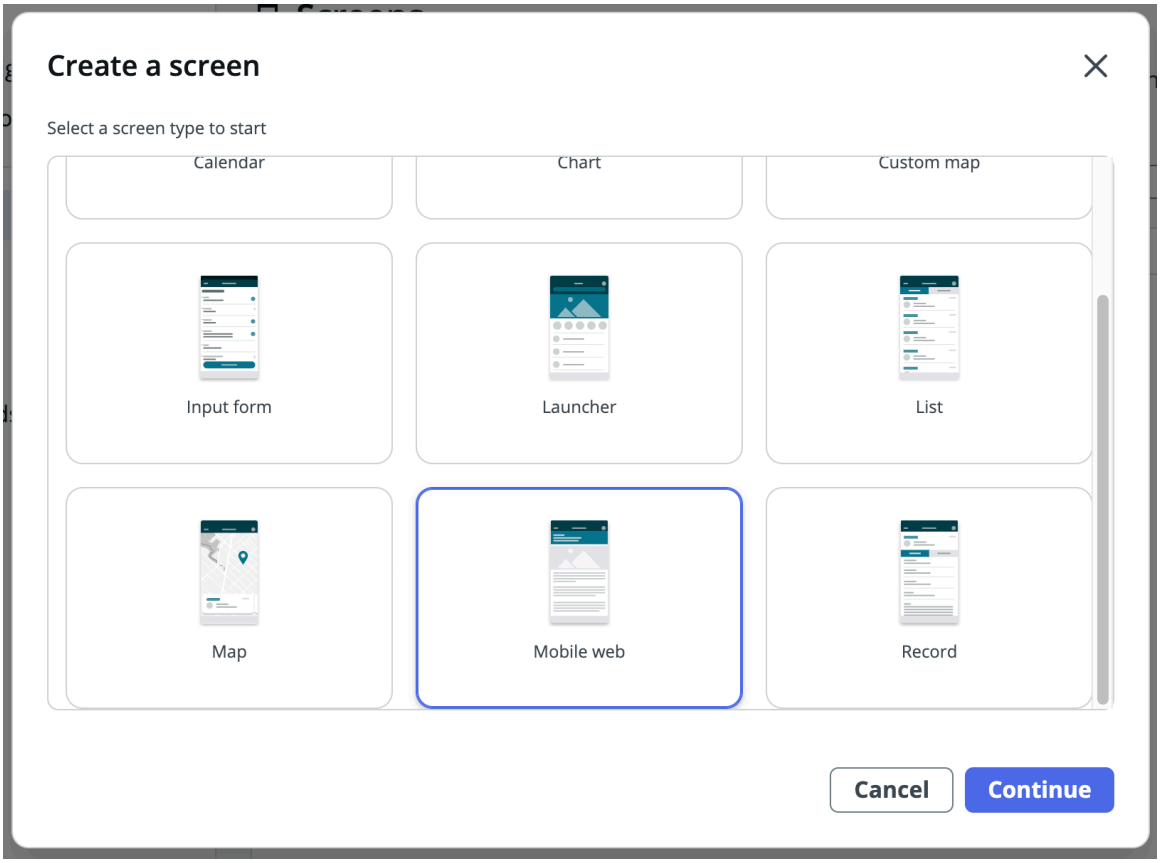


3. Navigate to **Screens**.

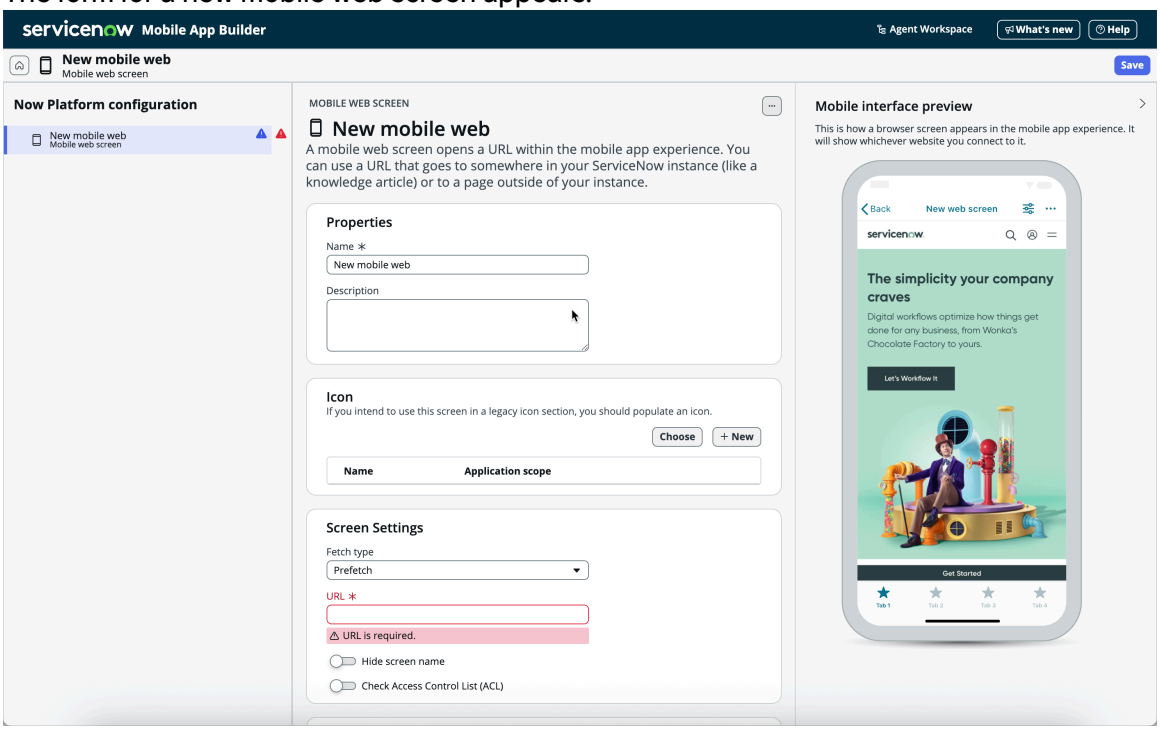


4. To create new screen, select **New**.

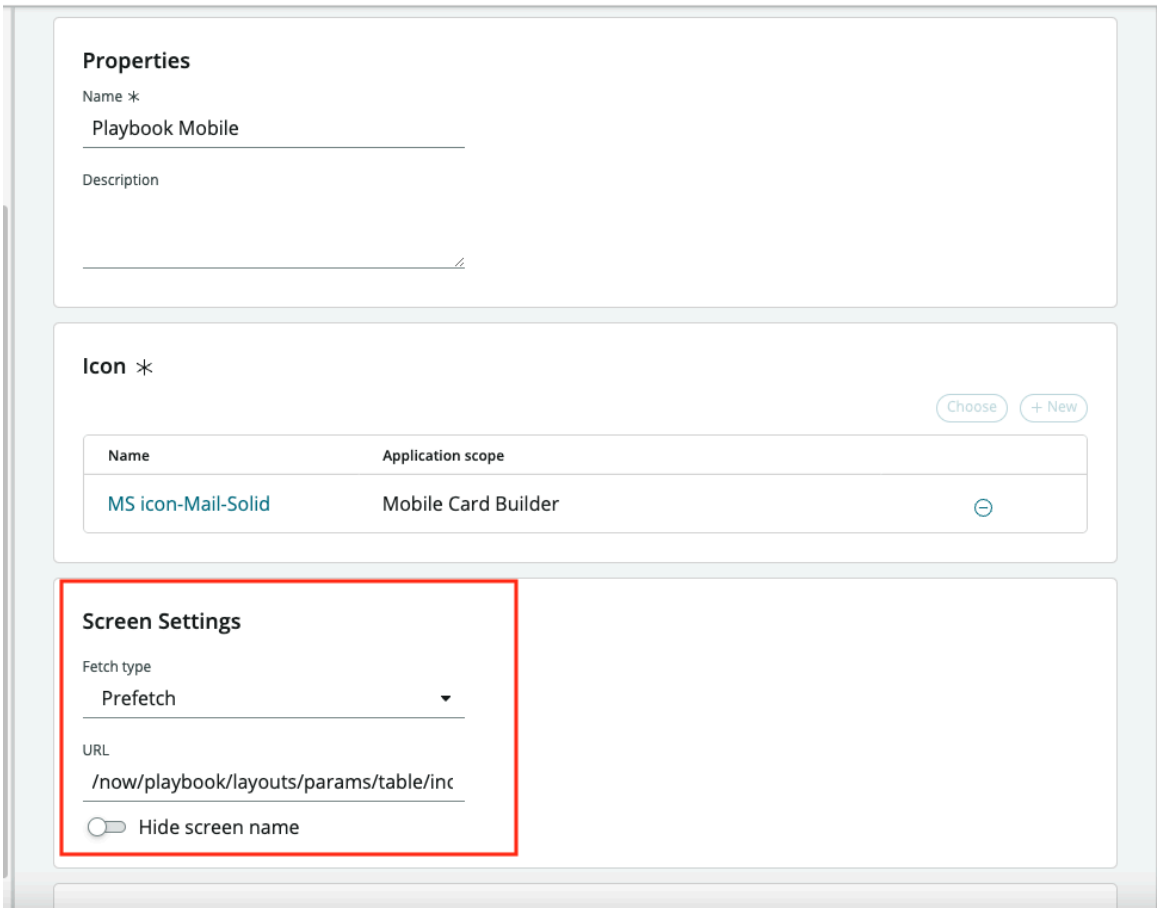
5. In the **Create a screen** modal, select **Mobile Web**.



The form for a new mobile web screen appears.



6. Give your mobile web screen a name and description.
7. Choose an icon for your mobile web screen.
8. Under **Screen Settings**, enter the URL of your mobile playbook.



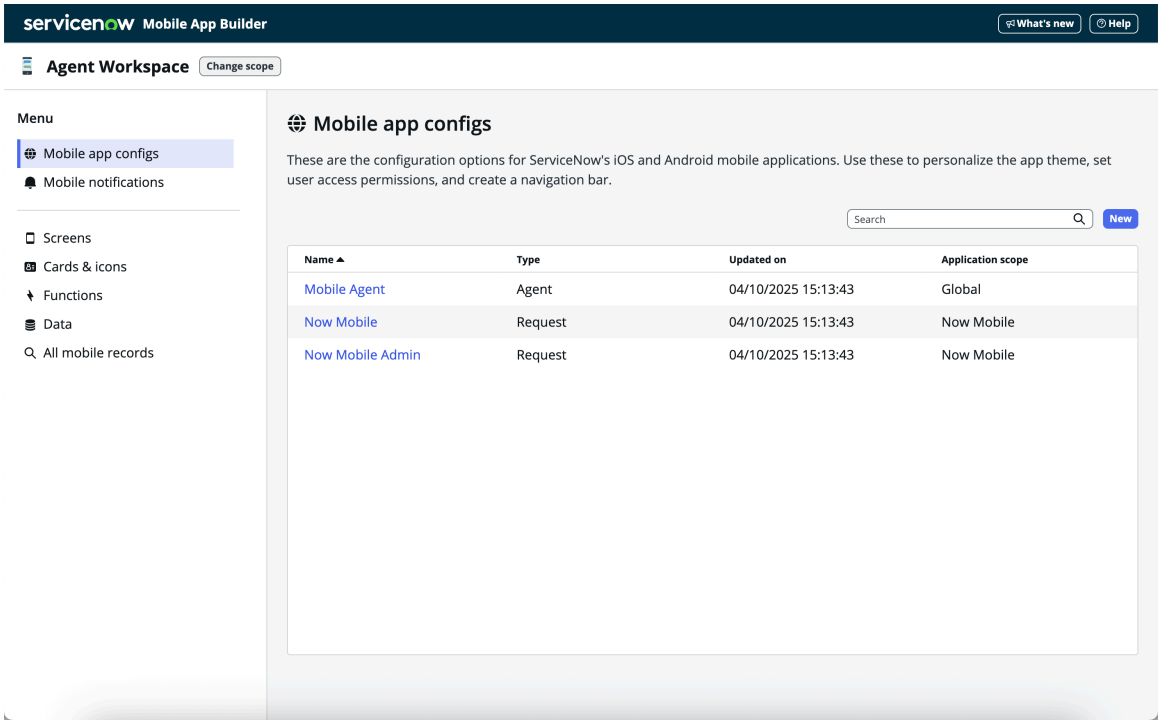
Important:
The **web_controller_spinner=on** parameter is important to include for proper loading behavior on mobile.

Example
/now/playbook-mobile/playbook/interaction/-1/params/view/stages?web_controller_spinner=on

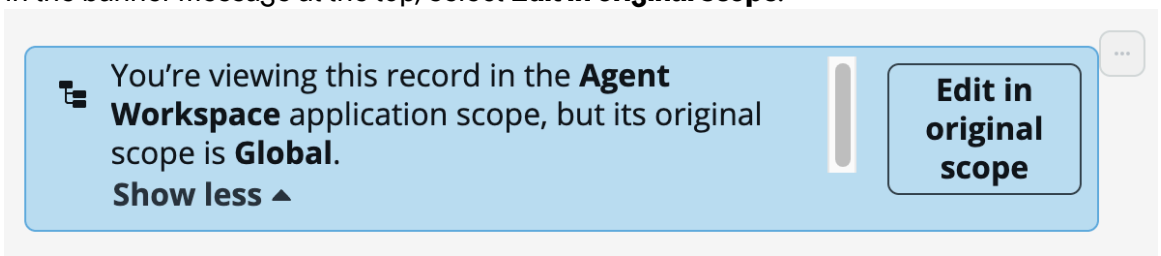
9. **Optional:** If you have any roles you want to limit this screen to, or any other configurations you would like to learn more about, see Configure a mobile web screen.
10. **Save** the screen.
    Your mobile web screen is configured for your playbook.

Configure the Mobile Agent

11. Navigate back to the scope level by selecting the home icon ( ).
12. Select **Mobile app configs**.



13. Select **Mobile Agent**.
14. In the banner message at the top, select **Edit in original scope**.



15. In the component tree, select the **Launcher screen** component.
16.

Result
Your playbook is now embedded in ServiceNow® mobile.

Reflow for playbook components

Apply reflow to out-of-the-box standalone and custom layout Playbook Experience components so that the UI adjusts when you resize your window or zoom.

Reflow, is a configurable workspace feature that enables pages and content to be zoomed up to 400% through browser settings. In a playbook with reflow enabled, the playbook changes to compact mode when zooming in to 200% browser zoom and leaves compact mode when zooming back out.

If you want Reflow to work for your Playbook Experience components, see Apply Reflow to playbook components.

For more information about Reflow, see Reflow for Configurable Workspace.

Apply Reflow to playbook components

Apply reflow to out-of-the-box standalone and custom layout Playbook Experience components so that the UI adjusts when you resize your window o r zoom.

Before you begin

Role required: admin

Procedure

1. Create the sys_ux_auto_reflow_rule record.
   a. Open the **All** menu.
   b. In the filter bar, type and enter **sys_ux_auto_reflow_rule.list**.

   The **UX Auto Reflow Rules** list is created and displays.

   There is a rule for each of the playbook standalone and custom layout components. When under a certain UI Builder page size, these rules use the default Reflow engine to override macroponent properties in the **UX Macroponent Definitions** list (**sys_ux_macroponent.list**) and certain CSS values. The rules provided in the Playbook Experience store apps use a 640px page width to toggle the components' **compactMode** property, as well as a 100vh height to ensure the components resize to fit the space.

   If you're using the standalone component, you are done. If you are working with custom layout components, proceed to the next step.
2. Navigate to **All** > **UI Builder**.
3. Open the layout you want to apply Reflow to in Playbook Experience Builder.
4. In the bottom left corner, select the **Data** icon and open the **Playbook Custom Layout** UI controller.

5. In the **Activity View Mode** field, update the value for the **stagePickerVisible** output property to **true**.
6. Select the component that you want the Reflow rule to apply to.
7. Under the **Events** tab, add the **Compact Mode Changed** event handler.
   This automatically turns Compact Mode on and off according to the Reflow rule, by changing the value of the **compactMode** output property to true or false. This is then applied to the other components of your playbook so that everything automatically resizes as well.
8. **Optional:** Select **Resizable panes**.

   a. Update the **Panes position** to show in the **left and right** orientation, or as **top and bottom** panes.

   b. Under the **Config** tab, open the scripted property value the **Default displayed pane** field.

   c. Update the value for **if(!api.data.playbook_custom_layout.compactMode) return** to show only the **"left"**/**"top"** pane, only the **"right"**/**"bottom"** pane, or **"both"** panes when not in Compact Mode.

   d. For Compact Mode, update the first value for **return (api.data.playbook_custom_layout.selectedItem || ()).stageContextId ?** to show the **"left"**/**"top"** or **"right"**/**"bottom"** activity pane when a stage is selected.
   The second value indicates which pane to show when a stage is not selected.

Playbook record generator

Use the playbook record generator to guide a user through the record creation process using a playbook experience.

Playbooks requires a record to be created or updated before a process can start. However, you can use the playbook record generator to allow users to create a new record using the playbook experience. You can then configure your workspace or UI Builder page to display the record generator playbook experience component in place of the standard new record form when a user opens a new record tab.

Playbook record generator inserts a record generator activity as the first step within a specified process definition created with Playbooks. This record generator activity contains a new record form. Once a user submits the form, the user is redirected to the newly created record, which now contains a running process. The running process guides the user through the rest of the record creation. If no process definition is running after the new record form is submitted, then the playbook will manually trigger whichever process definition was shown to user before record creation. The user stays within the playbook experience before and after the record is created for a seamless and guided record creation experience.

Admins can specify the name of the record generator activity, the form view, and the process definition shown to the user before the record is created. Admins can also optionally configure the declarative action used to submit the form.

Running playbooks

Using Playbooks as an agent or fulfiller.

About Playbook Experience

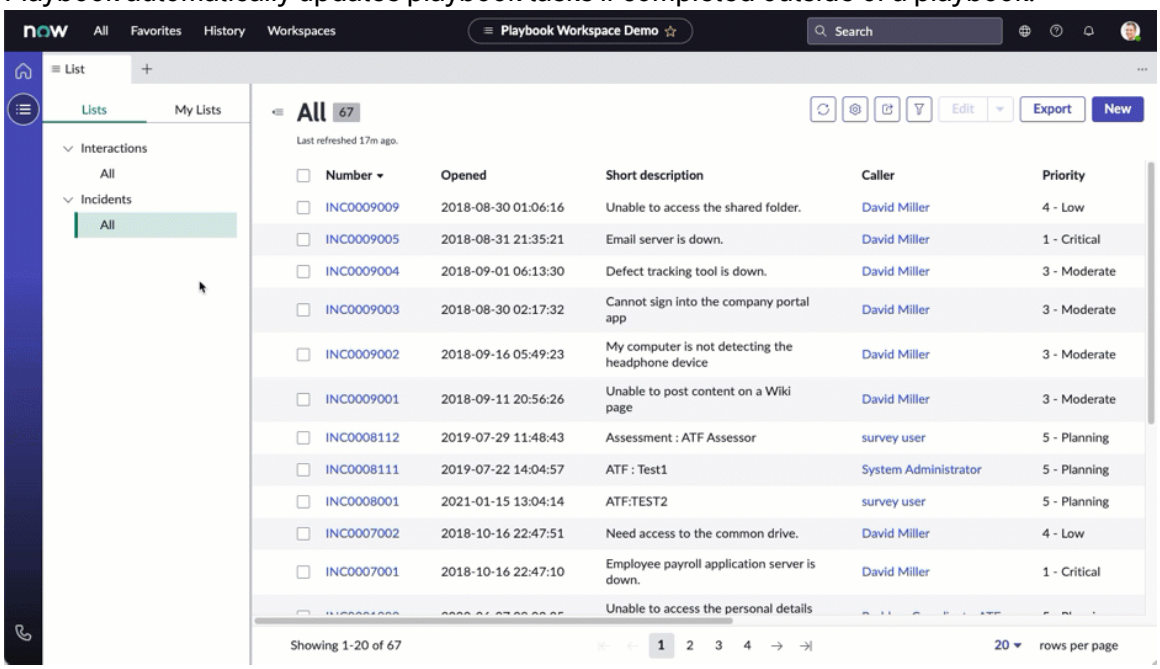
Interact with a business workflow in real time from within Workspace. Agents can use Playbook Experience to update records, upload attachments, and complete tasks across multiple workflow activities.

Playbook Experience overview

Playbooks are built by admins in Workflow Studio, customized by admins in UI Builder and more, and embedded in Workspace, Configurable Workspace, Service Portal and more. The Playbook Experience for fulfillers during runtime provides visibility into cross-business workflows and the actionable tasks used to complete these business workflows.

- Playbooks may appear in the side panel or in the related items of records configured with playbook.
- Activities that you must perform to complete the business workflow are displayed. You're able to see what you've done and what you must do to complete the playbook. You can collapse activities to display relevant activities, and expand them again.
- Activities are typically performed sequentially. You can go back to those activities to complete them later. Complete an activity and go to the next activity, or complete the playbook.

Playbook automatically updates playbook tasks if completed outside of a playbook.

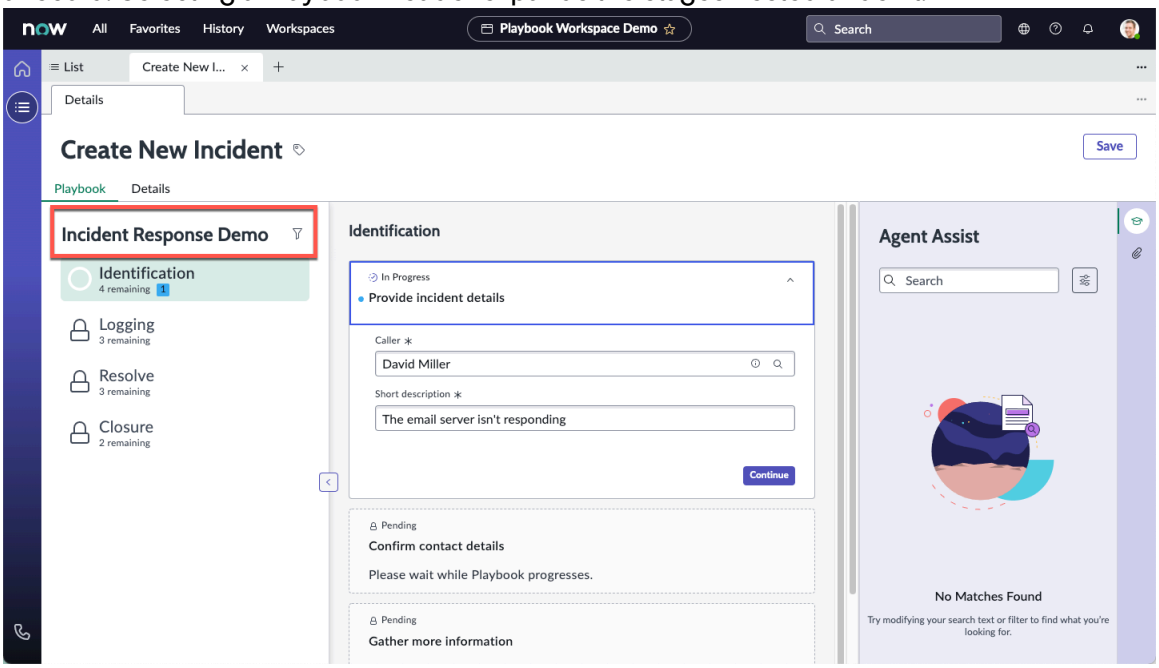


Playbook UI

Playbooks contain helpful UI features.

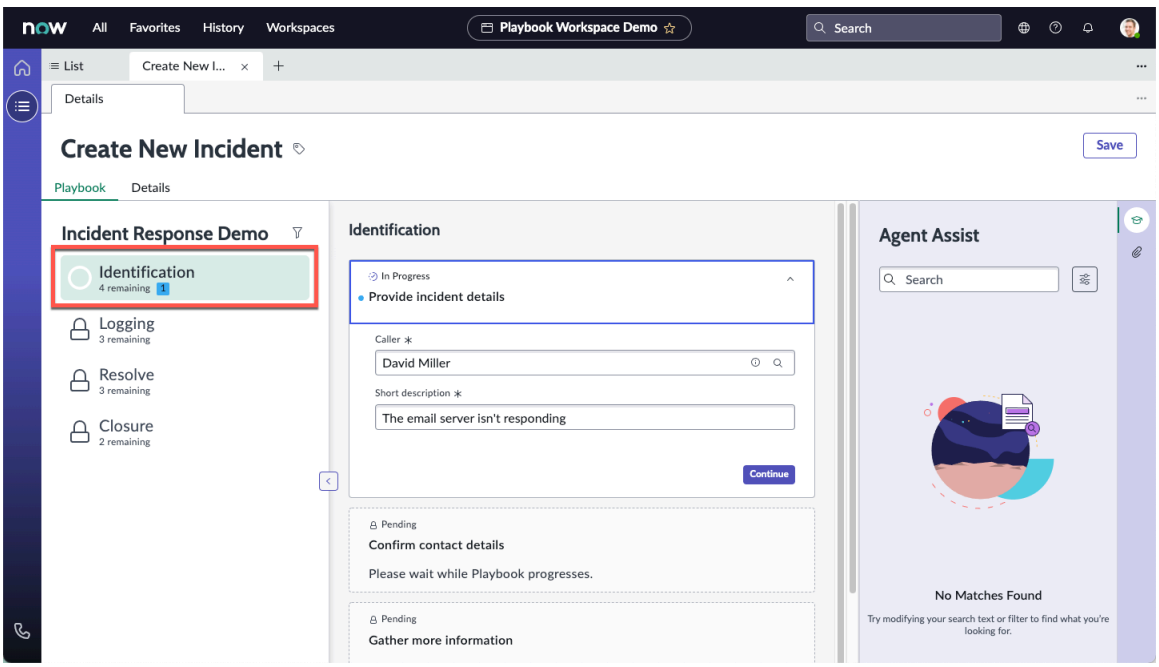
Header

Shows the title of a Playbook. A header exists for each playbook attached to a record. Selecting a Playbook header expands the stages nested under it.

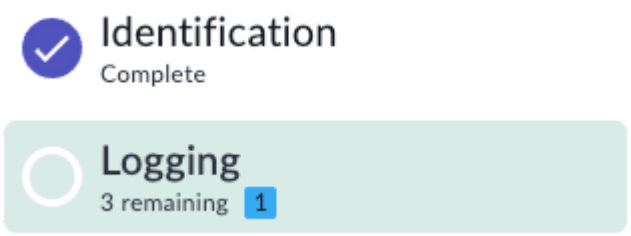


Stages

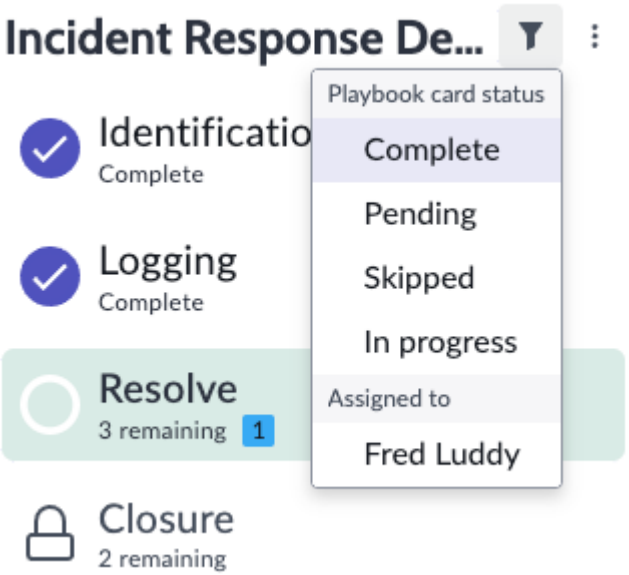
Click a stage title to view its activities. By default, all activity cards are collapsed except for the first card in a stage.



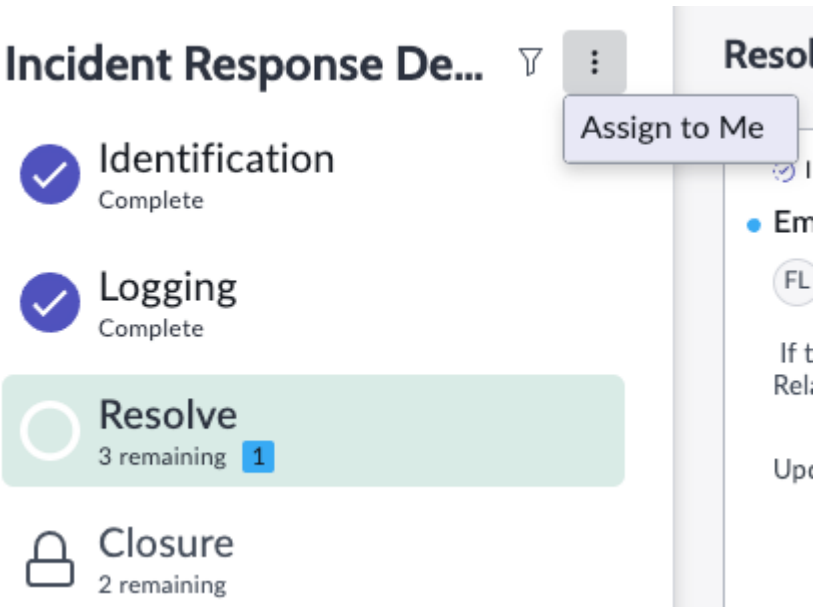
The stage progress updates as activities are completed. A checkmark inside the playbook header indicates that the stage is complete.



Use the stage filter ( ) to filter a playbook.



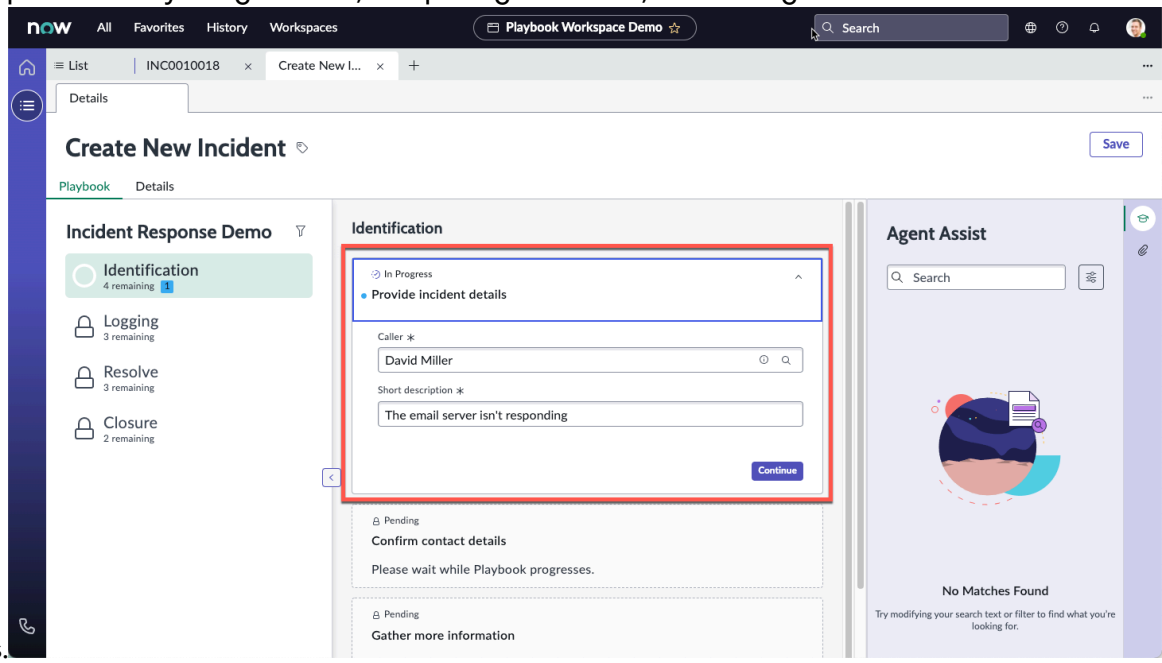
Use the ellipses action menu icon ( ) to perform select actions at the playbook and stage level.



Activity cards

Playbook activity cards display details about an activity, which may include the status, SLA timer, form data, and attachments. Use playbook activity

cards to complete tasks by filling in forms, completing checklists, and adding attachments.



Add an activity to a playbook

Add pre-selected optional activities to a playbook if available.

Before you begin

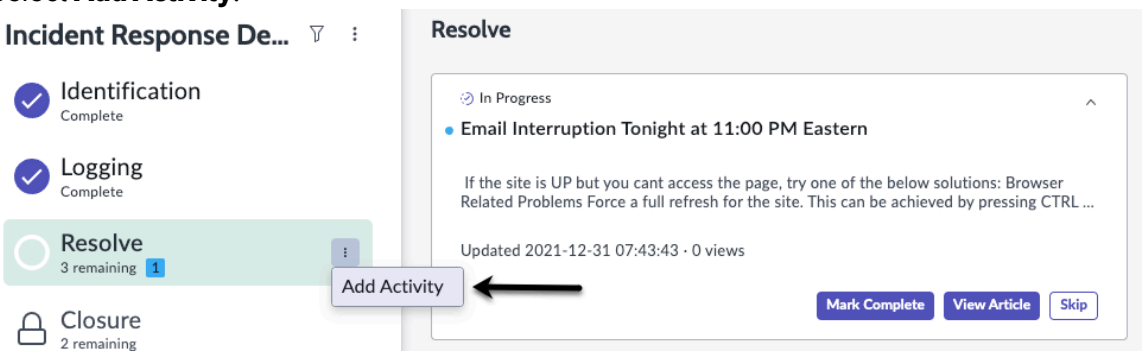
Role required: agent

About this task

Add and complete optional playbook activities in a running playbook. This example shows how an optional activity can be added to a playbook. Playbook admins must pre-configure optional activities. You cannot insert an optional activity as the first activity in a stage or between two finished activities. Some optional activities can be added to any stage while others can only be added to a single stage. If optional activities are not configured for a playbook, no optional activities will be available.

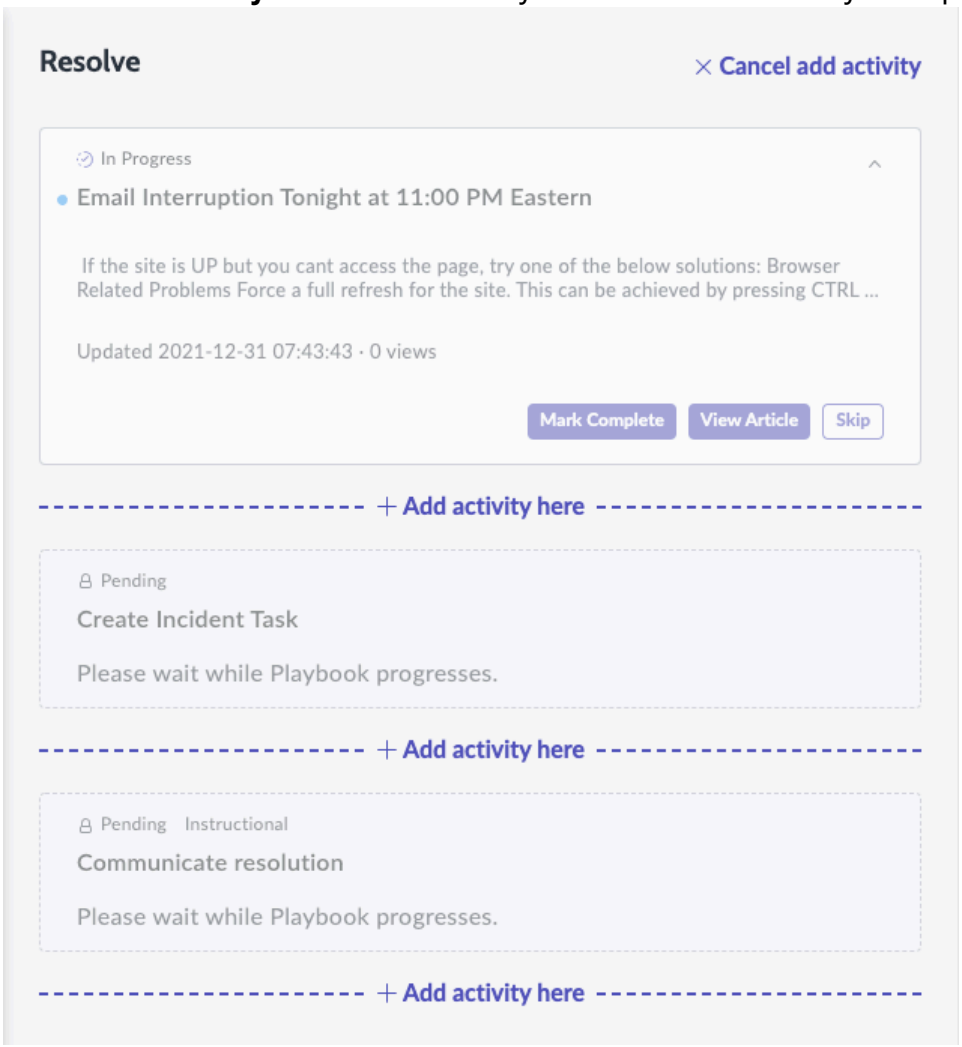
Procedure

1. Open a playbook in workspace.
2. Select the action menu icon (⋮).
3. Select **Add Activity**.

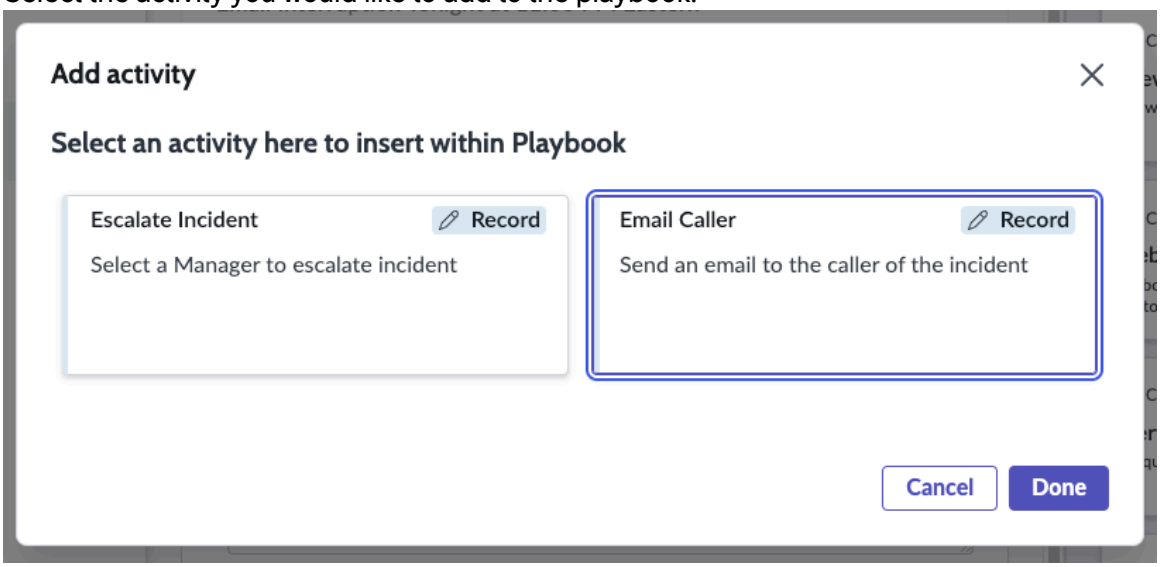


Note:
Add Activity will not display if no optional activities available in this stage.

4. Select **+ Add activity here** in the location you want to add the activity to the playbook.

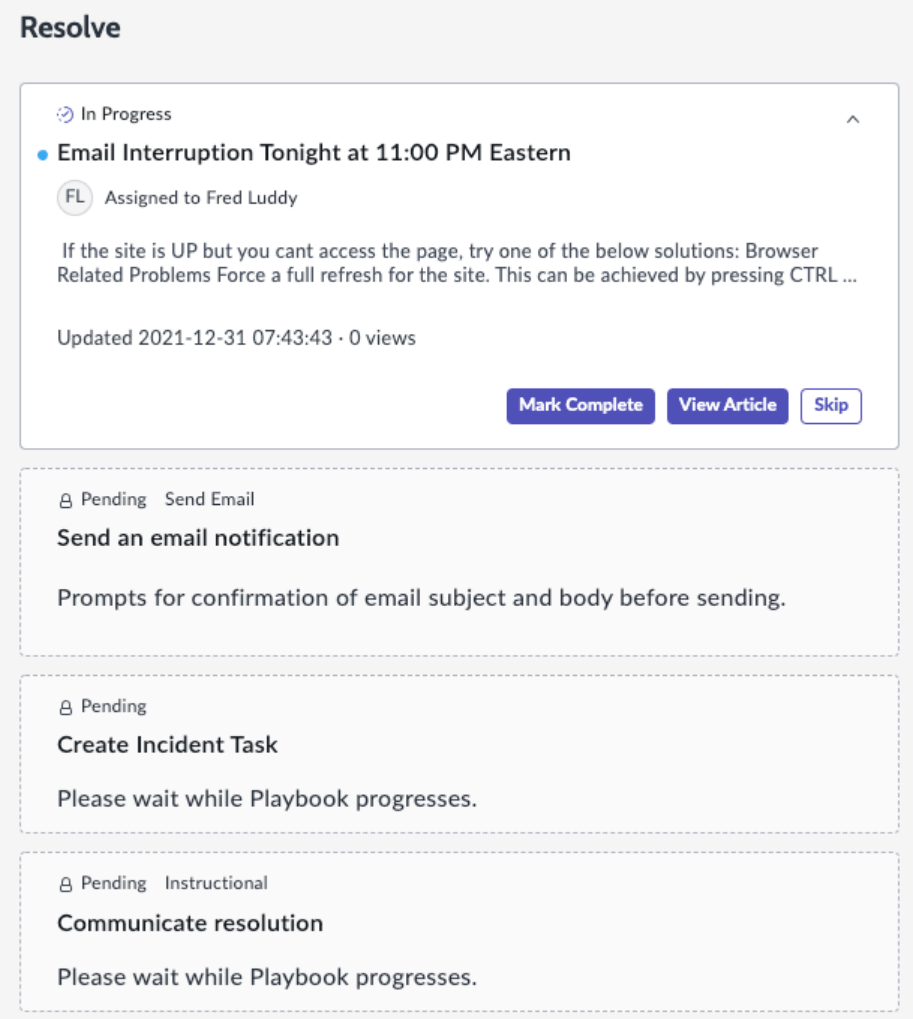


5. Select the activity you would like to add to the playbook.



6. Select **Done**.
   The selected optional activity appears in your playbook.

Result
The selected optional activity appears in your playbook.



Restart a playbook

Restart a playbook from the beginning, an activity, or a stage.

Before you begin

Role required: agent or pd_restarter

If your playbook admin has enabled restart in your playbook, you can find the restart action in the action menus of the playbook, stages, or activities. If you are an admin and would like to enable and configure restart for a playbook in Workflow Studio, see Enable and Configure Restart for Playbooks.

If a stage or activity is not configured to be restartable, you will not see all of the buttons in this procedure.

Note:
Certain activities within a playbook cannot be restarted once they have started. Playbooks in a Cancelled, Complete, or Error state also cannot be restarted. However, the opposite is true for activities and stages. Activities and stages must be complete before they can be restarted.

Procedure

Restart playbook

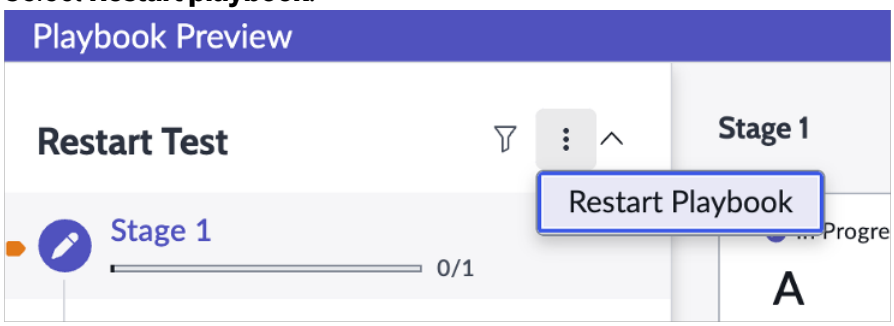
1. **Note:**
   Playbooks in a **Complete**, **Error**, or **Cancelled** state cannot be restarted.

   Open your playbook.

2. To restart the whole playbook:

   a. Hover over the playbook name in the contextual side panel to open the context menu ⋮.

   b. Select **Restart playbook**.



Restart stage

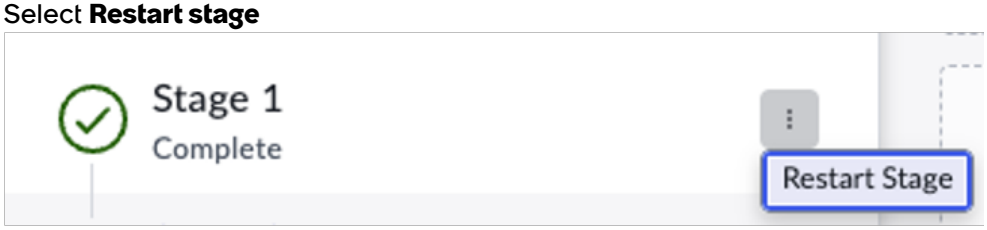
3. Open your playbook.

4. **Note:**
   Stages and activities must be completed before they can be restarted. When a stage or activity is restarted, you may need to:
   - Complete all restartable stages and activities after it again.
   - Re-enter information to complete restartable stages and activities again.

   To restart a stage:

   a. Hover over the name of the stage in the contextual side panel to open the context menu ⋮.

   b. Select **Restart stage**



.

Restart activity

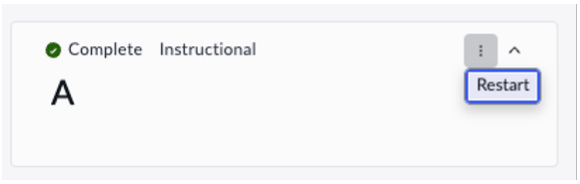
5. Open your playbook.

6. **Note:**
   Stages and activities must be completed before they can be restarted. When a stage or activity is restarted, you may need to:
   - Complete all restartable stages and activities after it again.
   - Re-enter information to complete restartable stages and activities again.

   To restart an activity:

a. In the activity card, open the context menu.

b. Select **Restart**.



Cancel a playbook

Cancel a playbook to stop a business workflow when no longer valid.

Before you begin

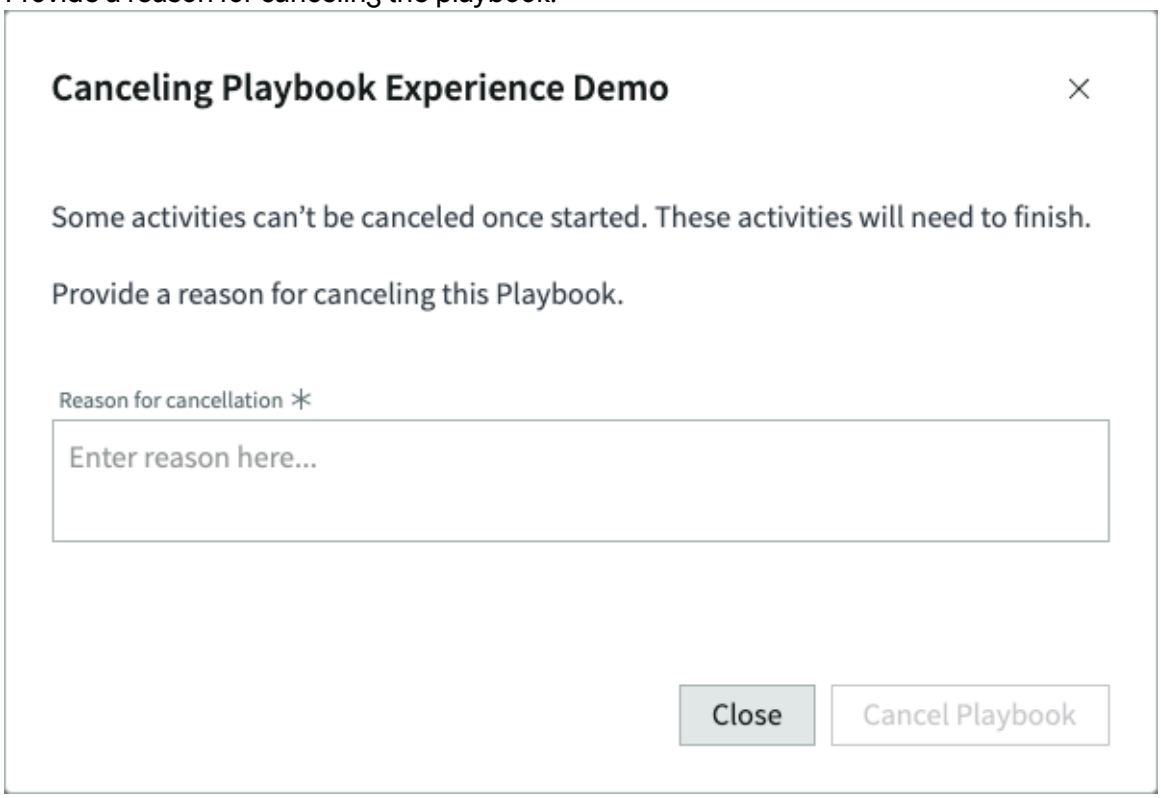
Role required: agent or pd_cancel

If your playbook admin has added a cancel action to your playbook experience, you can typically find the cancel action under the playbook actions menu.

Note:
Certain activities within a playbook cannot be canceled once they have started.

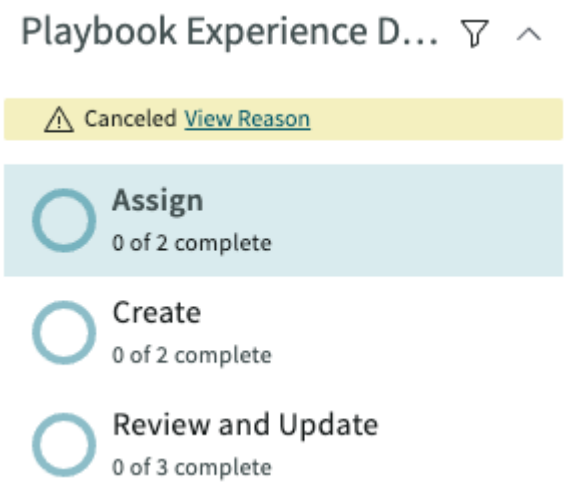
Procedure

1. Open a playbook in workspace.
2. Click the ellipses action menu icon (⋮) in the playbook header.
3. Click **Cancel Playbook**.
4. Provide a reason for canceling the playbook.



5. Click **Cancel Playbook**.

A canceled banner appears below the playbook header confirming that the playbook has been canceled.

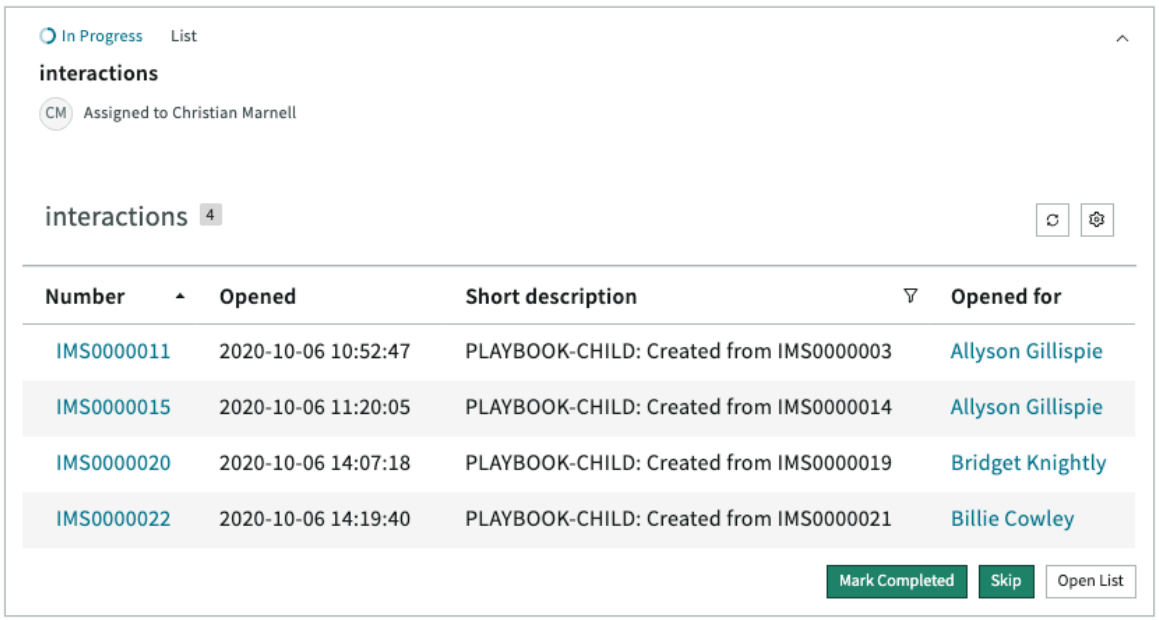


Open full lists within playbook

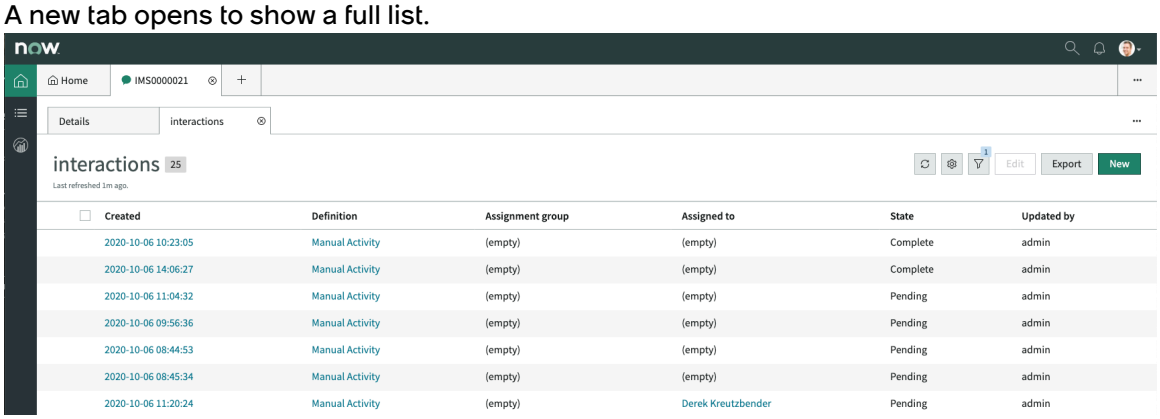
Open a full list within playbook cards to view and update list items.

To open a full list within playbook, click the **Open List** icon (Open List).

Card view



A new tab opens to show a full list.



Using activity stream within playbook

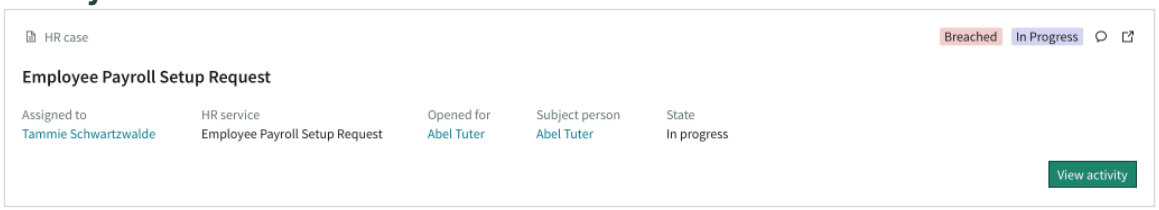
Use activity stream within playbook to add comments or notes, and view communication and task history for the parent or associated record.

Activities may contain a button, icon, or dropdown item to view a record's activity stream.

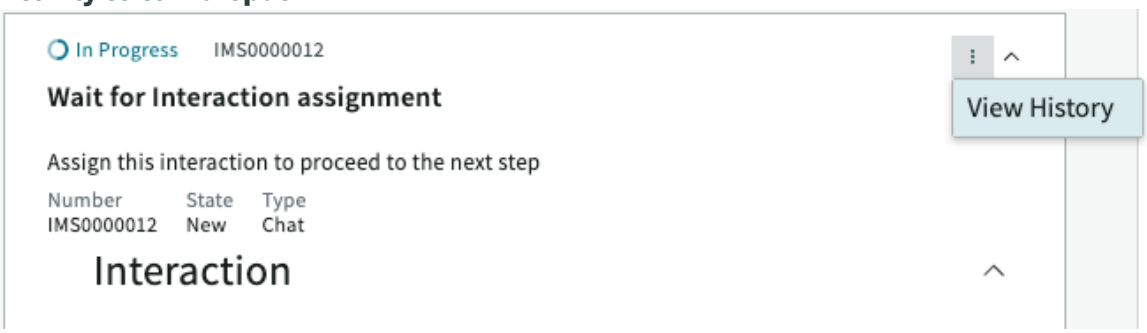
Playbook activity stream examples

To open the activity stream within playbook, click the **View activity** button or select **View History** from the dropdown list.

Activity stream button

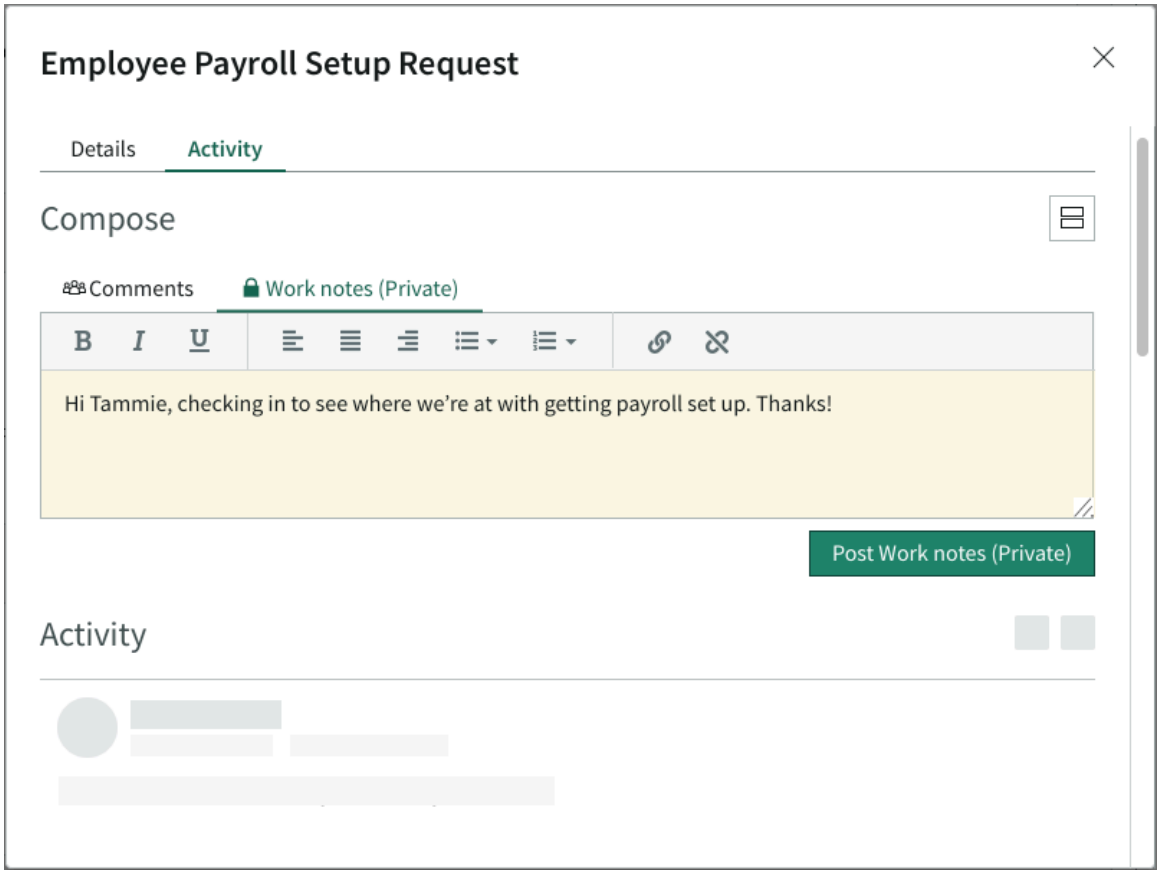


Activity stream dropdown



Activity stream modal

You can see tabs for comments and work notes related to a parent or associated record. You can use the **Compose** text box to post new comments or work notes.



You can view previous activity by scrolling down to the **Activity** section.

Using decision tables

Using decision tables in Workflow Studio, you can create, modify, test, and update decision tables for use in your workflows and applications. You can also create decision tables in Workflow Studio and then modify them in Excel.

Create decision tables in Workflow Studio

Create decision tables in Workflow Studio to embed business logic into a series of if-then decision rules. Use decision tables when business logic is complex or may be reused in multiple places.

Before you begin

Role required: admin, decision_table_admin, or delegated developer permissions. For more information, see Delegate developers using App Engine Studio.

About this task

You can create decision tables for use in flows, subflows, playbooks, and scripts. Create your decision table and then reference it from one of those objects, or anywhere else on the platform that you write code. As of the Xanadu release, you can also create the structure of a decision table directly in a flow or playbook, and populate the table afterwards.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. Select **New**.
3. Select **Decision table**.
4. On the form, fill in the fields.

Properties form

Field	Description
Decision table name	Name for the decision table.
Description	Description of the decision table.
Application	Application scope for the decision table.
Accessible from	Scopes that can use the decision table. Available values are **Application scope only**or **All application scopes**.
Enable draft authoring	Option to author decision tables in draft mode before publishing to make them available for use. This function also enables you to change a decision table in draft mode after it has been published and republish with changes.

5. Select **Build decision table**.
6. Select **Add an input**.

> **Note:**
> Note that adding more then 30 inputs to your table can adversely affect the performance of the application while you create the table.

7. On the form, fill in the fields.

Input definition

Field	Description
Label	Header for the input.
Type	Type of data used for the input. When the input type is **Reference**, a new column titled **Reference** appears in the input section and displays the reference table. This input type enables you to add multiple condition columns to a decision table. The available input types are: ○ Choice ○ Currency ○ Date ○ Date/Time ○ Decimal ○ Due Date ○ Integer ○ Long

Field	Description
	○ Reference ○ String ○ String (Full UTF-8) ○ True/False
Mandatory	Option for making an input field required when using the table.
Add reference filter	Option to filter the list of reference records in a linked condition column where the input type is **Reference** and the Data to evaluate is **Reference record**. Use this field to build a filter condition statement. For more information, see Filter reference inputs and results in a decision table.

Note:
For the Choice input type, the ability to select an existing choice list depends on the choice list living on a table in the same application scope as the decision table. For example, if you create a decision table in the Global scope, you can use any existing choice list on tables in the Global scope. But, if you create a decision table in a custom scope, you can only use existing choice lists from tables in the same custom scope.

8. Select **Add condition column**.
9. On the form, fill in the fields.

New condition column form

Field	Description
Condition column label	Label for the condition column.
Description	Brief overview of the condition column.
Input	Input linked to the condition column. To evaluate multiple fields, you can add multiple conditions with the Reference input type.
Table	If the data type is Reference, the name of the reference table is displayed.
Data to evaluate	For condition columns with the input type of Reference, specifies whether the condition column evaluates the reference record or a field on the reference table.
Condition type	Data type selected for the condition column.
Default operator	How every row in the condition column evaluates a user-specified value. A default operator is required for all input data types except for True or False.

Field	Description
	For more information about operators, see Operators available for filters and queries.

10. Select **Done**.
11. **Optional:** Add more condition columns.
    a. Navigate to the last condition column, select the plus icon (+), and select **Add condition column**.
    b. Point to a condition column and select the plus icon (+).
    c. Select the **Add condition column** button to the right of an input.
12. Select **Add result column**.
13. On the form, fill in the fields.

New result column form

Field	Description
Result column label	Label for the result column.
Description	Brief overview of the result column.
Result type	Type of data used for the result column. The available result types are: ○ Choice ○ Currency ○ Date ○ Date/Time ○ Decimal ○ Due Date ○ Duration ○ Integer ○ Long ○ Reference ○ String ○ String (Full UTF-8) ○ True/False
Add reference filter	Option to filter the list of reference records in the result column when the result type is Reference. Use this field to build a filter condition statement. For more information, see Filter reference inputs and results in a decision table.

Note:
The Currency and True/False result types have several important exceptions.

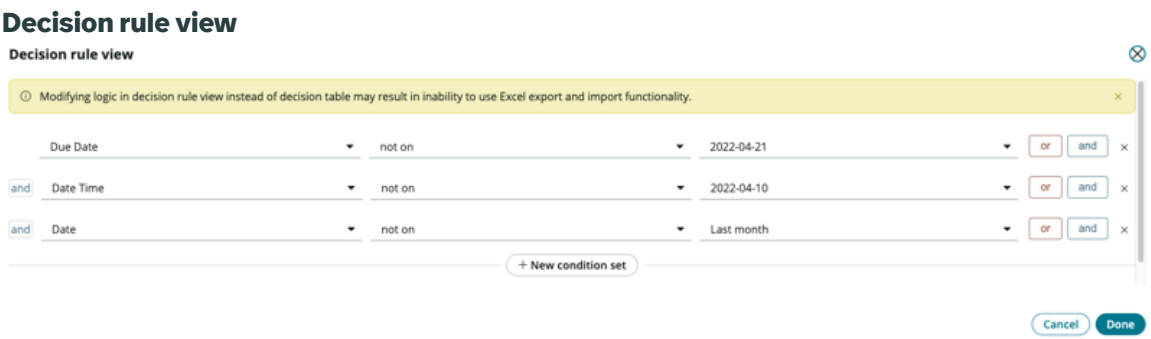
- If no alternate result is specified (0.00 for currency, false for true/false), cells in currency and true/false type result columns return a default value.
- For instances using multi-currency mode, you can specify currency results using any available instance currency. However, result values are always converted to the session currency on saving.
- For instances using single-currency mode, you can only specify currency results using the single instance currency.

14. Select **Done**.
15. **Optional:** Enable multiple results in a decision table by adding more result columns using one of the following methods.
    - Navigate to the first result column, select the plus icon (+), and select **Add result column**.
    - Navigate to the last result column and select the plus icon (+).
    - Point to a result column and select the plus icon (+).
16. For each condition, click into the empty box in the condition column to select an operator and enter a value.

For more information about operators, see Operators available for filters and queries.

17. **Optional:** To modify a decision table condition in Decision rule view, select the Decision rule row menu options icon (⋮) to the left of the row number, and then select **Open in Decision rule view**.

    (Optional) Decision rule view supports complex conditions that may not fit into the table structure.

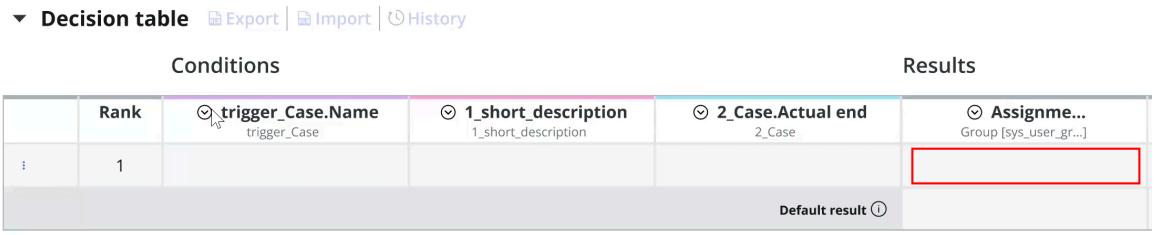


Note:
Using Decision rule view can result in creating complex conditions. Tables with advanced rows can be edited in Excel but advanced rows are read-only. For more information, see Manage decision tables in Excel. Where possible, split complex decisions into multiple simplified decision rule rows.

a. **Optional:** Edit the condition as needed.

b. **Optional:** Select **Done** to commit your changes.

18. For each result, click into the empty box in the result column and enter a result value.



19. **Optional:** Add more decision rules by selecting **Add new decision row** and entering conditions and desired results.
20. Select **Save**.
21. Select **Publish**.
A modal appears asking if you're sure you want to publish. If you want to make edits to this table after it's published, you must create a draft of the table. For more information about editing a published decision table, see Edit decision tables using draft authoring.
22. Select **Publish**.

Duplicate a decision table

Save time creating similar decision tables by duplicating existing tables in Workflow Studio.

Before you begin
Role required: admin, decision_table_admin, or delegated developer

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Open the decision table you want to duplicate.
4. Select the Decision table menu options icon (⋯) and select **Duplicate Decision Table**.
5. On the form, fill in the fields.

Duplicate this table form

Field	Description
New decision table name	Name for the decision table. This field pre-fills with `Duplicate of <original table name>`.
Application	Application scope for the decision table.
Version to duplicate	Version of the table you want to duplicate. This field is only available when draft

Field	Description
	authoring is enabled on the original table and there is a draft and a published version. You can select **Draft** or **Published**.
Include decision rows	Option to include the decision rows as well as the structure of the original table. This option is limited to tables with 300 rows or fewer. If your table has more than 300 rows, this feature is not available, and you can only duplicate the structure, not the rows.

6. Select **Duplicate**.

Edit decision tables using draft authoring

Edit published decision tables in Workflow Studio when draft authoring is enabled. Without draft authoring enabled, decision tables are active and available to use as soon as they're created.

Before you begin

Role required: admin, decision_table_admin, decision_rule_author, or decision_result_editor

About this task

To modify a published decision table, you must create a draft, make any required changes, and publish the draft.

Important:
Older versions of the decision table aren't retained, so if you publish a draft, you can't go back and access the previously published version again. This new published version is the live version.

Draft authoring is enabled by default on all new decision tables. For existing tables, draft authoring is available but not enabled by default.

To enable draft authoring for existing tables, ensure the decision table has no unsaved changes, open the decision table properties, and select **Enable draft authoring**. After you make this change, the decision table is moved to a published state, and you must create a new draft to make further changes.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a published decision table.
4. Select **Create Draft**.
   This action creates a draft version of the table that you can edit to your liking and republish. The published version is available and is the version scripts, playbooks, and flows use when they execute decisions.

   You can view both the draft and published versions of the table by switching the **View** toggle.
5. Change your decision table.
6. Select **Save**.

7. If you want this version to become the new published version and the original published version irretrievably replaced, select **Publish**.
8. Select **Publish**.

Modify decision table structure in Workflow Studio

Evolve with your business logic by modifying the inputs or columns of your decision table in Workflow Studio.

Modify an input for a decision table

Modify an input by changing the label, modifying the type and associated properties, and changing the mandatory toggle.

Before you begin

Role required: admin, decision_table_admin, or delegated developer permissions. For more information, see Delegate developers using App Engine Studio.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. Choose the input that you want to modify.
5. Modify the input.
   - To modify the label, select the label, provide a new name, and press **Enter**.
   - To change the input type, choose a different input type from the Type list.

     Note:
     The input type can be modified only if the input is not linked to a condition column.

   - To modify the input type properties, select the property field and update the value.
6. Select **Save**.

Filter reference inputs and results in a decision table

Narrow down the list of results to only information that you need to see for Reference type input and result fields in decision tables in Workflow Studio.

Before you begin

Role required: admin or decision_table_admin

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. Create a filter for an input or result.

5. Select **Add reference filter**.
6. Build a filter by adding conditions that contain a field, operator, and value.

You can build a more complex filter by selecting **New condition set** and adding more conditions with **or** or **and** to string them together.

7. Select **Done**.
8. **Optional:** Add a linked reference condition column with the filters you've created applied by selecting **Add condition column** in the Reference input row.
9. **Optional:** In the **Data to evaluate** section, select **Reference record**.
   A filtered input filters the list of records in linked condition column cells where the data to evaluate is **Reference record**. A filtered result column filters the list of records in the corresponding result column cells.
10. Select **Save**.

Update an existing reference filter

Make changes to an existing filter on Reference type inputs or results in decision tables in Workflow Studio.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. Select an input or result that you want to modify.
5. Point to the result column header, and select the Edit column properties icon ( ).
6. Select **Modify reference filter**.
7. Make your changes and select **Done**.
8. Select **Save**.

Delete an input from a decision table

Delete inputs that are no longer necessary from your decision table in Workflow Studio. Deleting an input also deletes any related condition columns.

Before you begin

Deleting an input from a decision table can break a flow, playbook, or script passing data through the input. Because of this possibility, you should know where the decision table is used before deleting anything from it.

Role required: admin, decision_table_admin, or delegated developer permissions. For more information, see Delegate developers using App Engine Studio .

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. At the end of the input row, select the Input menu options icon ( ), and select **Delete**.
5. Select **Save**.

Modify a decision table condition column

Modify a condition column by renaming the label, editing the description, or updating the input type or default operator of your decision table in Workflow Studio.

Before you begin

Role required: admin, decision_table_admin, or delegated developer permissions. For more information, see Delegate developers using App Engine Studio.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. Point to the condition column header and select the Edit column properties icon ( ).
5. Make your changes and select **Done**.

> **Note:**
> If you have an input of the type *Choice*, and you switch the input type to any other type, that action deletes all values from the condition column.

6. Select **Save**.

Modify a decision table result column

Change the data that a decision evaluates by changing the source table for that decision table in Workflow Studio.

Before you begin

Role required: admin, decision_table_admin, delegated developer permissions. For more information, see Delegate developers using App Engine Studio.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. Point to the results column header and select the Edit column properties icon ( ).
5. Replace the existing result table with a new source table.

> **Note:**
> Changing the result table deletes the current decision table results values.

6. Select **Done**.
7. Select **Save**.

Modify decision table rules in Workflow Studio

Evolve with your business logic by modifying the rows, default result values, or content of your decision table in Workflow Studio.

Define default result values

Account for changing business logic by defining default result values in your decision tables in Workflow Studio. Defining a default result accounts for scenarios when no decision rules are met by the input data.

Before you begin

Role required: admin

About this task

If no values are defined in the Default result row or you have cleared values from the row, no default results are returned.

> **Note:**
> You can set default results for decision tables created prior to Decision Builder version 4, but no values are set automatically. The Default result row is compatible with any decision tables created outside of Decision Builder with a default answer specified. If you have the decision_result_editor role, you need an admin or user with higher level permissions to set the initial default result for any decision tables created in an earlier version of Decision Builder.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. Define at least one Result column in the decision table.
   With a result column defined, the Default result row appears at the bottom of the table.
5. Define a default result for any columns that need one.

   > **Note:**
   > If any column has a default result value, and there are other result columns with the currency or true/false result types, those result columns automatically specify a default result. You can clear all default result values using the ellipsis menu on the row. However, if you add a default value in another column, currency and true/false result type columns auto-populate with a default value.

6. Select **Save**.

Modify a decision rule in a decision table

Modify conditions and results to update decision rules in a decision table in Workflow Studio.

Before you begin

Role required: admin, decision_table_admin, or delegated developer permissions. For more information, see Delegate developers using App Engine Studio.

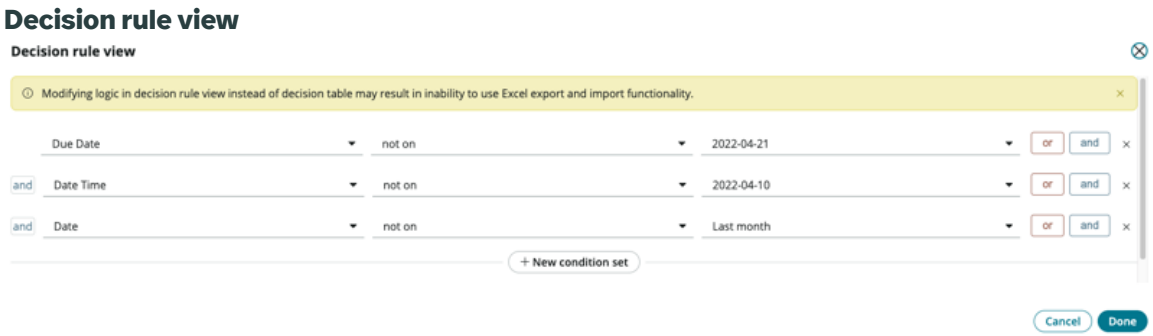
Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. Select the cell that you want to edit.

- Modify a condition in one of the following ways.
  - Select the operator and replace it with another operator.
  - Select the condition value and provide a new value.
  - Update both the operator and the condition value.
- To modify a result, select the result value and provide a new value.

5. **Optional:** To modify a decision table condition in Decision rule view, select the Decision rule row menu options icon (⋮) to the left of the row number, and then select **Open in Decision rule view**.

   (Optional) Decision rule view supports complex conditions that may not fit into the table structure.



> **Note:**
> Using Decision rule view can result in creating complex conditions. Tables with advanced rows can be edited in Excel but advanced rows are read-only. For more information, see Manage decision tables in Excel. Where possible, split complex decisions into multiple simplified decision rule rows.

   a. **Optional:** Edit the condition as needed.

   b. **Optional:** Select **Done** to commit your changes.

6. Select **Save**.

Duplicate rows in a decision table

Save time and effort creating decision rules by strategically duplicating rows in a decision table in Workflow Studio.

Before you begin
Role required: admin, decision_table_admin, or decision_rule_author

About this task
Plan the best way to duplicate rows to maximize efficiency and save time creating your decision table. For example, if you have five rows that have the same first condition, fill in the condition in the first row, and then duplicate the rows. Rows can be duplicated at any time and any point in their completeness.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.

3. Open a decision table.
4. On the far left side of the row you want to duplicate, select the Duplicate decision row below this row icon ( ).

Reorder decision rows in a decision table

Determine or change the sequence in which your decision rules are evaluated by reordering the rows in a decision table in Workflow Studio. Enter a different number for your row in the **Rank** column to quickly reorder rows in large tables.

Before you begin

Role required: admin, decision_table_admin, or delegated developer permissions. For more information, see Delegate developers using App Engine Studio .

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. Reorder the rows using either of the following methods.

Option	Description
Reorder by changing the Rank number	**a.** In the **Rank** column, double-click a number to edit it. **b.** Enter a new value for the row and press **Enter**. **Note:** If you change the number but don't press **Enter**, the rank does not change. This method is most effective for reordering rows in large decision tables.
Reorder by dragging and dropping a row	**a.** Point to a decision row and select the Row drag and drop gripper icon ( ) in the far left. **b.** Reorder the rows by dragging the row to a new location. This method is most effective for reordering rows in smaller decision tables, as you can on ly see 20 rows at a time.

5. Select **Save**.

Use decision tables

Decision tables built in Workflow Studio are executed in flows with flow logic or in scripts with API calls.

Although decision tables are built in Workflow Studio, the actual execution of the decision takes place in other applications. Keeping decision actions separate from these applications allows the decisions to be reused and called when needed.

Decision tables in flows

In Workflow Studio, decisions are executed as part of the **Make a decision** flow logic. For more information, see Make a decision flow logic.

Decision tables in APIs

In scripts, decision tables are executed using API calls to the decision table with the GetDecision() or GetDecisions() methods from the DecisionTableAPI. For more information, see DecisionTableAPI - Scoped, Global.

Manage decision tables in Excel

If you have large decision tables to build or want to enhance decision tables outside of the ServiceNow AI Platform, manage them in Excel. Export a decision table to an Excel file, edit the downloaded file to add and edit rows, and then import the file back into Workflow Studio.

Exporting a decision table to Excel

After creating and saving inputs, condition columns, and result columns, export a decision table to Excel in `.xlsx` format by selecting **Export**. The exported file contains two worksheets: One with the decision table and the other with instructions on how to build decision rows and prevent import errors.

Editing your decision tables in Excel can help you build large tables more quickly. It also gives you the ability to assign the structural setup to a developer, and then delegate rule authoring to someone else, who could fill it out in Excel.

Note:
If you have any advanced rows in your decision table when you export to Excel, these rows are read-only and cannot be edited in Excel. You can edit the rest of the table in Excel and import it back into Workflow Studio.

Modifying the decision table in Excel

After you export a decision table to Excel, you can add, remove, edit, or reorder decision rules. Follow these directions to build your decision table.

- Build conditions using the operator and value columns.
  - In each **Operator** cell, choose the relevant operator from the drop-down list. The operator list is specific to the condition column field type.
  - In each **Value** cell, enter or select the condition value, following the format guidance from the instructions sheet.
  - When building date, due date, date/time, reference, or choice conditions, you can select a value from the drop-down list.
- Specify result values.
  - In each **Result** cell, enter the result value.
  - For reference or choice results, select a result value from the drop-down list.

Consider the following limitations when modifying a decision table in Excel.

- Use Excel to add, remove, or edit decision rules, but do not use Excel to add or modify condition columns or result columns.
- Edit only cells in the **Condition** and **Result** columns. During the import, any data entered to the right of these columns are ignored.
- Retain the headers in the exported file, and do not modify them in any way.
- Modify only the original exported file in Excel. Do not copy and paste the contents of an exported file into a new Excel file. However, the original file can be renamed.
- Ensure that there are no empty rows. Any entry after five consecutive empty rows is ignored.

Importing an Excel file into Workflow Studio

Note:
The import option is only available after you export the decision table to Excel.

During the import process, all the modifications in the Excel file are validated before your changes are imported to Workflow Studio. The following outcomes are possible while attempting to import.

- **Successful import**: The Excel file import is successful. The Import window closes automatically, and your decision table is updated with the imported data. Save your changes before continuing.
- **Failed import**: The Excel file import has failed. Download the `Error.xlsx` file that contains the detailed description about the errors and how to fix them. After fixing the errors, follow the import process and import the corrected error file.

View export and import history

View the export and import history of a decision table in the History sidebar. Each entry displays the name of the user who imported or exported the file and the timestamp.

You can download the relevant Excel file based on the history type: The exported file (Export Successful), the imported file (Import Successful), or the error file (Import Failed). History records are not created when you select **Cancel** in the import window.

Note:
The `glide.ui.export.choice_list_max_characters` property sets the maximum number of characters that will be included from a condition field type when exporting to Microsoft Excel, as well as the maximum number of characters displayed in the list view condition builder in Core UI. This property is of type integer and has a default value of 80. To use this property, add it to the System Property [sys_properties] table. It's important to note that this property does not affect the length of condition field values in tables.

Limitations

The following are scenarios when you cannot modify a decision table in Excel.

- Decision table does not contain a conditions column.
- Decision table has unsaved changes.
- Decision table has condition columns with unsupported field types.
- Decision table has a condition column that is related to an inactive input.

Note:
The Localization Framework is integrated in decision tables. However, because the Edit in Excel feature doesn't support localization, you cannot use this feature in any instance that doesn't use English.

View related objects in a decision table

See and open objects that are related to your decision table, such as flows, subflows, and playbooks.

Before you begin
Security constraints determine the related objects that you see. If you expect to see related objects but don't, contact your administrator for assistance.

Role required: none

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Open a decision table.
4. Select the Decision table menu options icon () and select **See Related Objects**.
   A list of related objects opens, showing you where your decision table is used in flows, playbooks, and subflows. Depending on your role, you may see an empty (or not as full) list with a number next to **Number of objects hidden by security constraints**.
5. **Optional:** Select a related object to open it in Workflow Studio.

Test a decision table in Workflow Studio

Test your decision table in Workflow Studio before publishing to make sure the rules provide the desired outcome for a given set of input data.

Before you begin
You can only test saved decision tables. You can either create a table with at least one input and result and then save and test it, or you can test an existing table that has been saved.

Role required: admin, decision_table_admin, decision_table_reader, Change manager, or delegated developer

Note:
Test decision table inputs within the specified character limits. For example, if the limit is 40 characters, ensure inputs do not exceed this length to avoid incorrect results.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Select a decision table.
4. Select **Test**.
5. If the table has draft authoring enabled and it has already been published, select whether to test the draft or published version from the **What to test** drop-down list.
6. From the **How to execute** drop-down list, select whether you want to return only the first decision where your input matches the conditions or all decisions where your input matches the conditions.

7. Enter input data to test.
8. Select **Test**.

Result

The results of the test are displayed, showing either no results or one or all of the decisions where your input data matches the conditions. You can run additional tests on the table by changing the test parameters and inputs, and selecting **Test** again.

Create decision table code snippets

Create code snippets of published decision tables in Workflow Studio to execute the decision table from any script and make it easier to replace hard-coded logic with decisions.

Before you begin

If you have draft authoring enabled, you must publish your decision table to be able to copy a code snippet of the table. If draft authoring is not enabled, save your decision table to access this feature.

The ability to copy code snippets for a decision table enables you to quickly insert the table into your script without having to author the code using API documentation. For more information about the Decision table API, see DecisionTableAPI - Scoped, Global.

Role required: none

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Open a published decision table.
4. Select the Decision table menu options icon () and select **Create Code Snippet**.
   Two code snippets are generated. The first code snippet calls the GetDecision() API, which returns only the first result that matches the inputs and specifications in the decision table. The second code snippet calls the GetDecisions() API, which returns a list of all the results that match your inputs.
5. Select either **Return first decision that matches** or **Return all decisions that match** depending on which code snippet you need.
6. Select **Copy Code**.
7. **Optional:** If you want to copy the other code snippet, select **OK** to close the confirmation message, and copy the code from the other option.
8. Insert the code snippet into your script where you want to call the decision table and define the data to use for each input variable.

Modify Decision Tables in the Classic UI

Workflow Studio supports decision tables that were created in the classic environment. However, some data types and conditions from the classic environment aren't fully supported in Workflow Studio. For unsupported field types and decisions with complex conditional logic, use the decision rule view to modify these condition expressions.

The first time you open a decision table that was created in the classic environment, Workflow Studio automatically creates condition columns for all unique fields evaluated in the condition criteria.

- If the table contains table-compatible decision rules, one condition per cell is displayed.
- If a decision rule contains unsupported field types or complex logic, all the conditions in the condition columns are merged into a single condition expression. This expression is a hyperlink to Condition Builder that you can use to modify the conditions for the decision rule. For more information, see Condition builder.
- If you remove the aspects of the condition expression that define the decision as complex, the decision is displayed in the supported format.

The following sections provide summaries of complex types of conditional logic.

Decisions with complex logic

Some decision rules may not be supported in Workflow Studio. These decisions are:

- Decisions with an OR condition in the condition expression
- Decisions with multiple criteria evaluating the same non-reference input
- Decisions with multiple criteria evaluating the same reference field or record
- Decisions that evaluate unsupported input data types
- Decisions that evaluate inactive inputs

Decision tables with unsupported input data types

If an input has an unsupported data type, you can change only the input label and the **Mandatory** toggle switch. The input data type is read only, and you cannot add new condition columns.

Decision tables with unsupported conditions

If a condition column has an unsupported data type, you can still change the label and description. The input and default operator are read-only.

- In an unsupported condition column, cells without values are inactive. Users must use the decision rule view to enter a value.
- When a value is entered for an unsupported condition column, the resulting decision row will be an unsupported decision.

Delete a decision table

Delete decision tables that you no longer need from the list of decision tables in Workflow Studio.

Before you begin
Role required: admin

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. On the homepage, select **Decision tables**.
3. Point to the far left side of the row your decision table is in, select the check box, and select **Delete**.
4. Select **Delete**.

Building spokes using Spoke Generator

Automate integration tasks by creating a sequence of reusable actions to develop custom integrations called spokes.

Request apps on the Store

Visit the ServiceNow Store website to view all the available apps, for information about submitting requests to the store, and to download and install Spoke Generator (sn_spoke_builder). For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

App dependencies

If you're having trouble installing the app, ensure that the dependent plugin, ServiceNow IntegrationHub Professional Pack Installer (com.glide.hub.integrations.professional) is installed.

Note:
Some of these plugins are licensable features and require appropriate licenses, if used outside the spoke implementation.

Capabilities

An Integration Hub custom spoke provides both inbound and outbound integration with third-party applications. These integrations, referred to as spokes, are easy to configure and enable you to quickly add powerful actions without the need to write a script. This feature requires an Integration Hub subscription. For more information, see Legal schedules - IntegrationHub overview.

You can build custom spokes for the required third-party application by importing an OpenAPI Specification or Postman collection, documentation snippets from third-party API documentation, or creating a spoke manually.

Required user roles

- Users with the admin role can:
  - View the **Spokes** tab.
  - Create a new spoke.
  - Import an OpenAPI Specification.
  - Create connection alias.
  - Publish actions and save actions as drafts.
  - See spoke activity log.
- Users with the action_designer and fd_read_actions roles can:
  - View the **Spokes** tab.
  - See actions in a spoke.
- Users with the action_designer role can add actions manually.

  Note:
  Users with just the fd_read_actions role can only see actions.

- Users with the flow_operator or flow_designer can view flows and subflows.

OpenAPI Specification size limit

By default, the system can import OpenAPI Specifications up to 10 MB. To increase the import size, update the *glide.rest.openapi.max_request_size* system property. The maximum value is 100 MB.

Limitations of OpenAPI Specification

Spoke actions created by importing OpenAPI specification, contain the OpenAPI step. Ensure that you are aware of these OpenAPI step limitations.

Request body media types

The request body only supports JSON media types.

> **Note:**
> A string type output object is created when the OpenAPI schema has additionalProperties or no properties.

OpenAPI 3.0 components

OpenAPI 3.0 adds new components to Swagger 2.0 to describe an API in further detail. OpenAPI support in the OpenAPI step supports some, but not all of these components. The OpenAPI step does not currently support these components.

- Schema Object: oneOf, anyOf, additionalProperties properties
- Discriminator Object
- Info object: termsOfService, contact, license fields
- Example Object
- Link Object
- Callback Object
- Security Scheme Object
- Security Requirements Object
- Tag Object
- External Documentation Object
- Server Object
- Specification extensions
- Recursive references

More information on these components is available in the OpenAPI documentation. See OpenAPI Specification.

Maximum number of operations supported

The number of API operations is limited to 500 by default. However, using the system property *glide.rest.openapi.max_operation_limit*, you can configure the number of operations from 1 through 1000.

App version

Spoke Generator v4.1.4 is the latest version.

Create spoke and build actions by importing an OpenAPI Specification

Automate an integration and generate reusable actions by importing an OpenAPI Specification.

Before you begin

- Install the Spoke Generator app from the ServiceNow Store.
- Role required: admin

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. Click **Create new** > **Spoke**.
3. On the Spoke Info screen, specify if you want to create the spoke in a new scope or an existing scope.

   a. If you choose to create the spoke in a new scope, select an image as the logo for your integration and fill in the fields.

   Here, we will go through an integration with Zoom as an example.

Field	Description
Spoke name	Name to identify the custom spoke.
Description	Description about the custom spoke.

SPOKE INFO

Let's get started on your new spoke

Add a name and description that define your spoke. You can also add a thumbnail image.

Choose how you want to build your spoke

- (•) Create a spoke in new scope
- ( ) Create a spoke in existing scope

Spoke name * ⓘ

Zoom spoke

App scope name ⓘ

x_snc_zoom_spoke

Description * ⓘ

ServiceNow integration with Zoom.

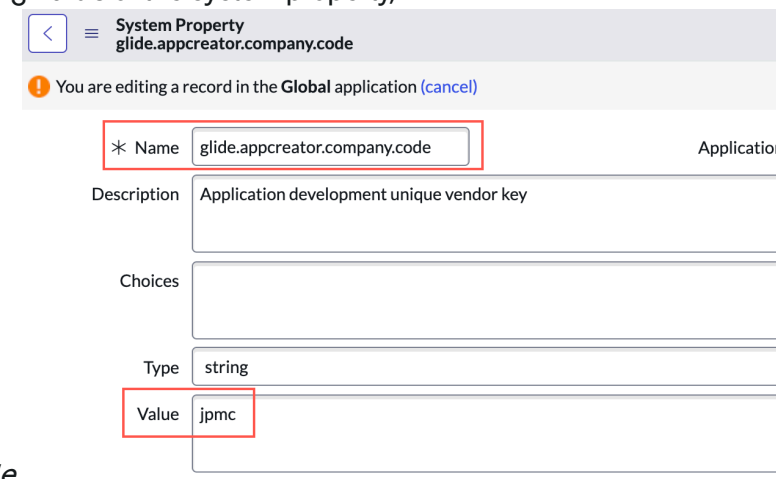
⊗ Remove image

Cancel

Continue

Note:

The value of **App scope name** is the format: `x_<company-code>_<spoke-name>_<spoke>`. By default, the <company-code> is, `snc`. You can configure the company code by configuring **Value** of the system property,



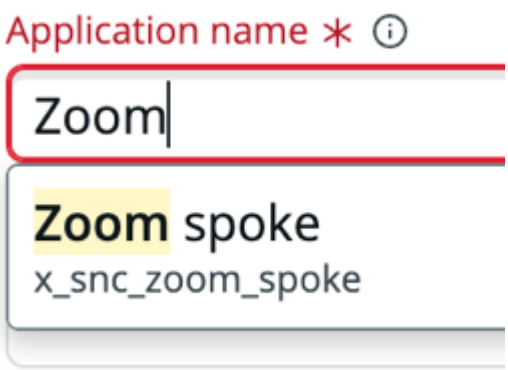
glide.appcreator.company.code.

This configured value is used when the value **App scope name** is

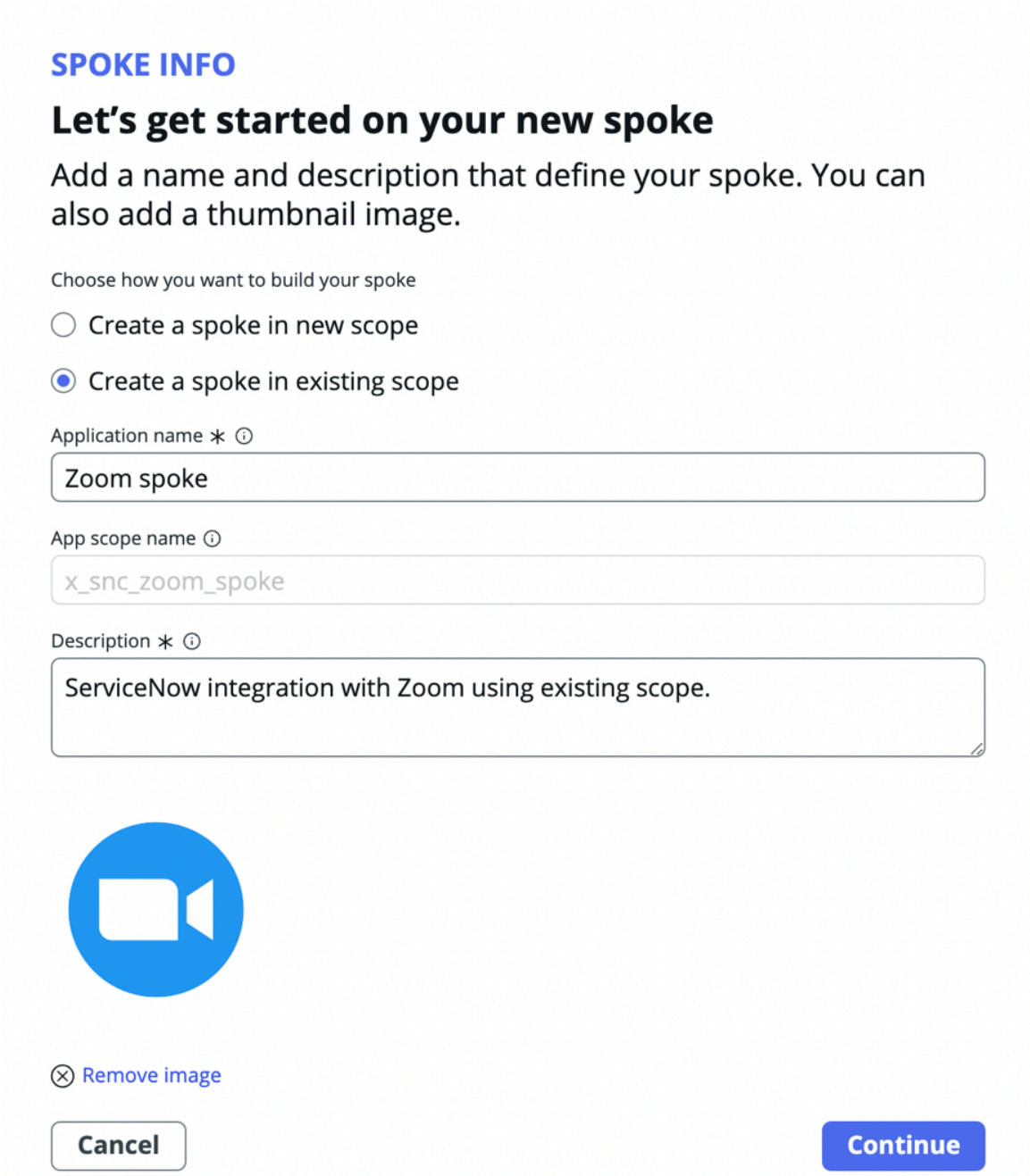
generated.

b. If you choose to create the spoke in an existing scope, select an image as the logo for your integration and fill in the fields.

Here, we will go through an integration with AWS as an example.

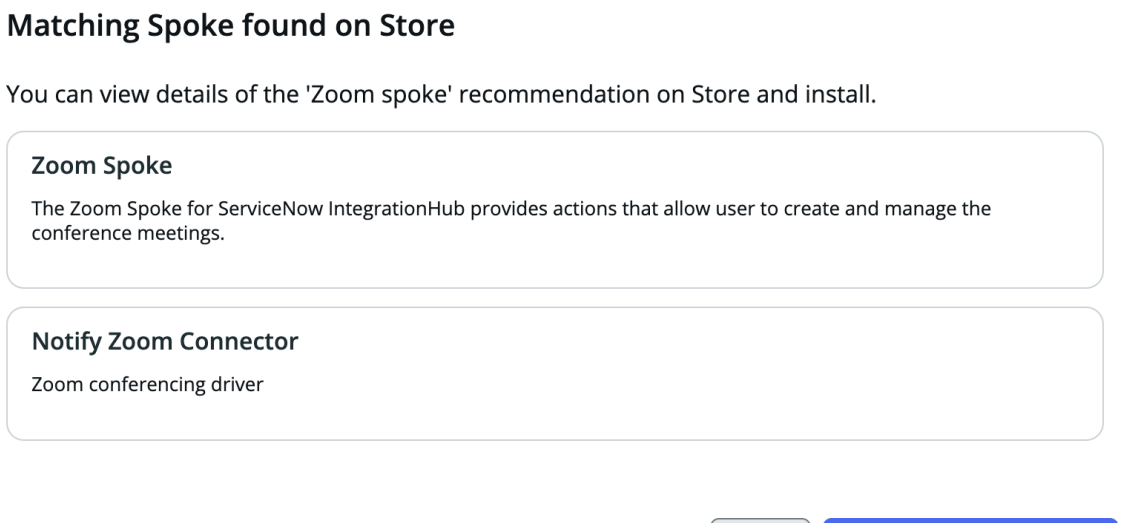
Field	Description
Application name	An existing application name or scope. 
App scope name	Scope name that is auto-populated based in the selected **Application name**.

Field	Description
Description	Description about the custom spoke.



4. Click **Continue**.

Based on the provided name and description, if there are any matching spokes on Store, the spoke details are displayed.



a. Click **View details on Store** to see the details of the matching spokes.
Details of the matching spokes are displayed in a new browser tab.

b. Install the spoke from the Store.
For more details, see Install a ServiceNow Store application.

c. After installing the spoke, navigate to the Workflow Studio tab.
The system displays the message `Have you installed a spoke from the Store?`.

d. Select one of these options and click **Continue**.

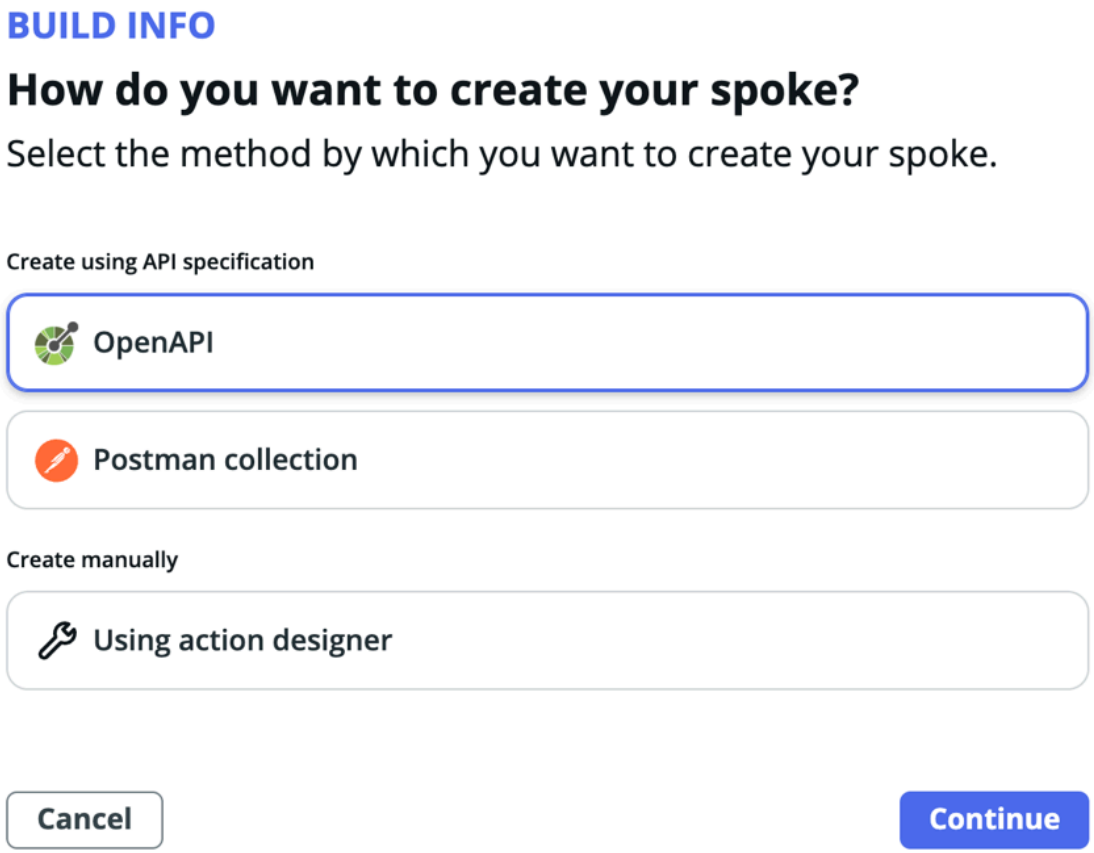
Option	Description
Yes, view the installed spoke.	Option to redirect to the **Spokes** dashboard under **Integrations**.
No, I will build custom spoke.	Option to continue with spoke creation.
No, I want to exit spoke creation.	Option to close the current tab.

5. Click **Skip** if you want to build the custom spoke.

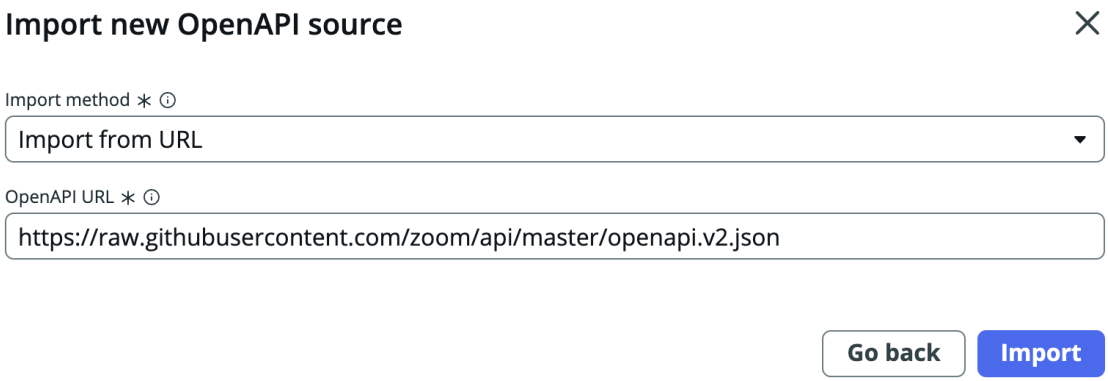
> **Note:**
> These following steps are also applicable if you have selected the **No, I will build custom spoke.** option.

The spoke is created and a confirmation message is displayed.

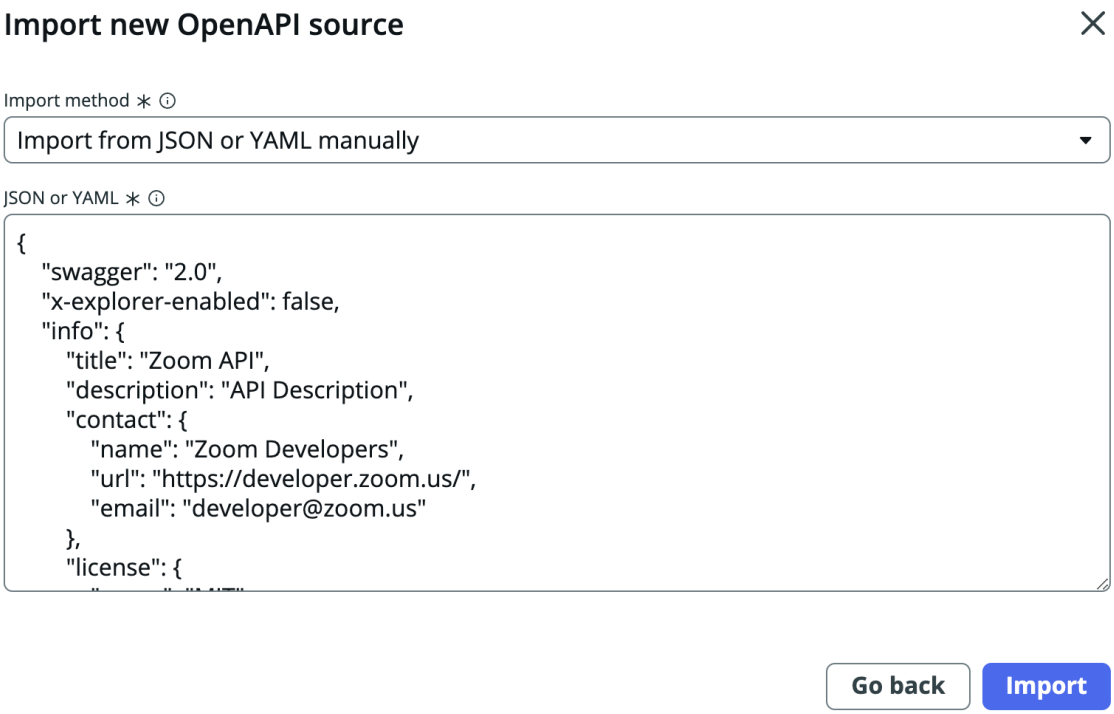
6. On the Build Info screen, select the method using which you want to build your spoke.
You can choose to build your spoke using OpenAPI specification or Postman collection.
7. Select **OpenAPI Specification** and click **Continue** to import an OpenAPI specification.



8. For **OpenAPI source**, click **Import new**.
9. On the Import new OpenAPI source screen, perform one of these two steps.
   a. If you want to import using an URL, select **Import from URL** for **Import method** and specify the URL in **OpenAPI URL**.

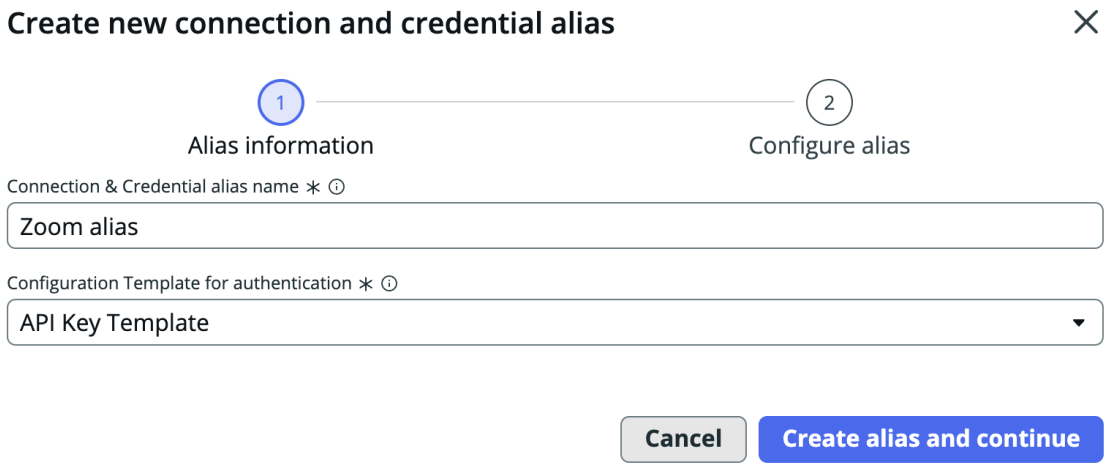


b. If you want to import manually using JSON or YAML code, select **Import from JSON or YAML manually** for **Import method** and provide the code in **JSON or YAML**.



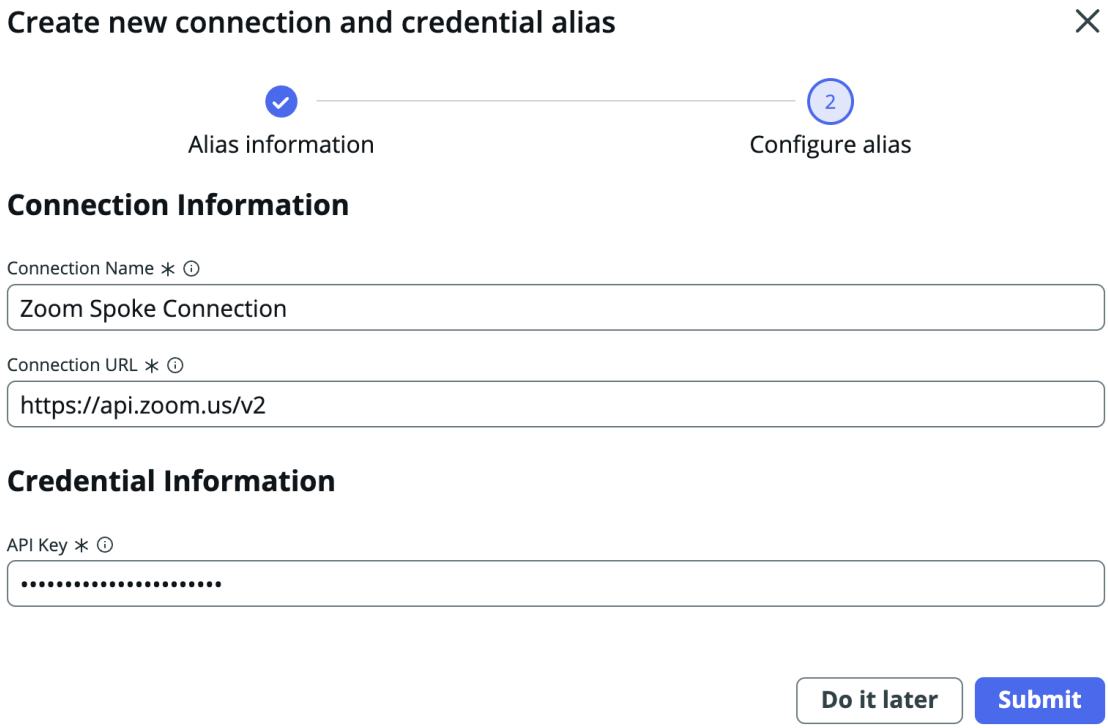
10. Click **Import**.
    The OpenAPI source is imported.
11. For **Connection and credential alias**, click **Create new**.
12. On the Create new connection and credential alias screen, fill in the fields and provide the alias information.

Field	Description
Connection & Credential name	Name to identify the connection and credential alias record.
Configuration Template for authentication	Required authentication mechanism for this integration. Ensure that the authentication mechanism is compatible with the OpenAPI source. **Note:** The configuration template is auto-populated based on the OpenAPI specification you had provided. You can continue with the default option or change it as per your requirement.



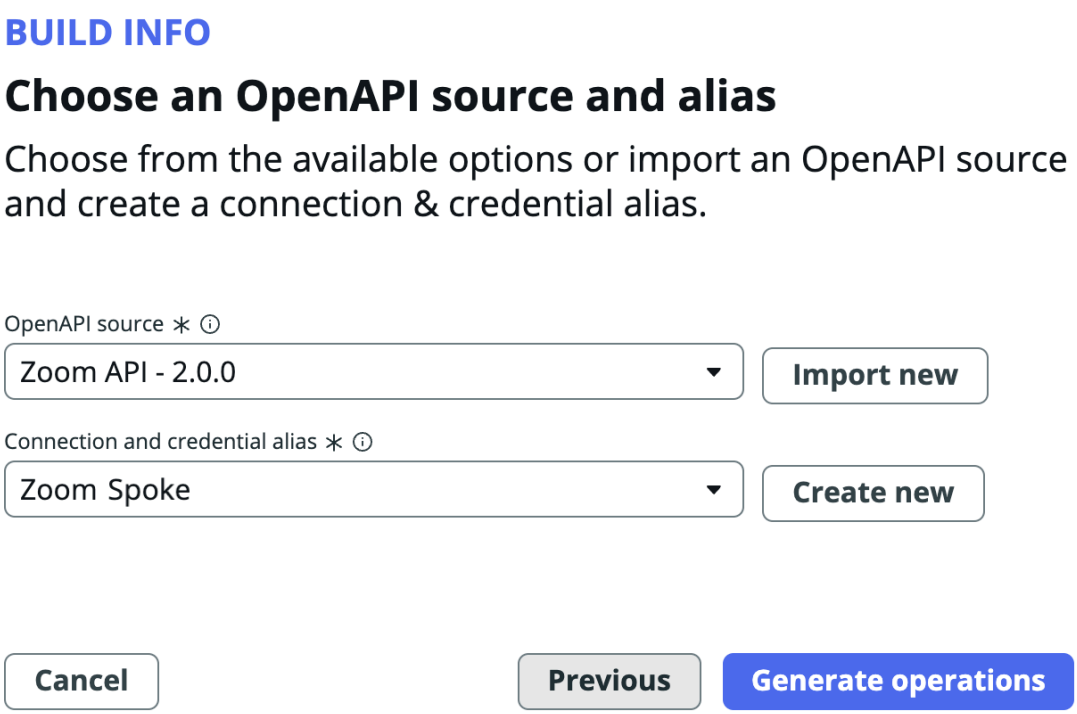
13. Click **Create alias and continue**.
14. On the same screen, fill in the fields to configure the alias.

Field	Description
Connection Information	
Connection Name	Name to identify the connection record.
Connection URL	Base URL to connect to the third-party instance or server. This URL is auto-populated based on the OpenAPI specification you had provided.
Credential Information	
Based on the configuration template you had selected, the relevant credential fields are displayed. Provide the required values to configure the credential record.	



If you want configure the alias record later, click **Do it later**.

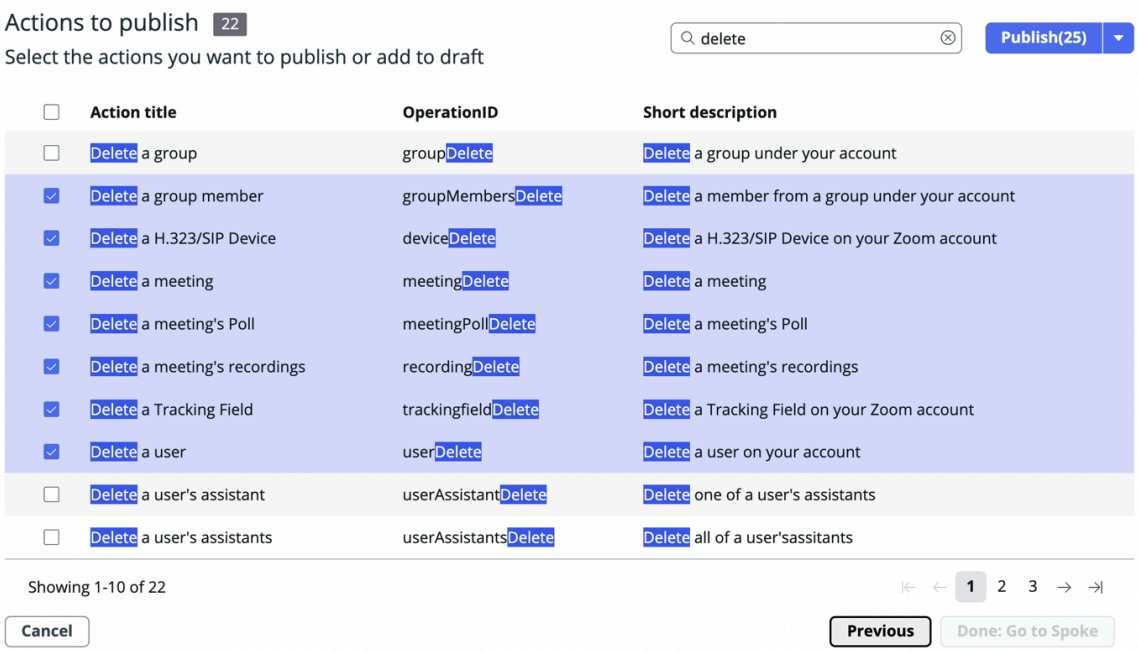
15. Click **Submit**.
    The connection alias record is created.
16. Click **Generate operations**.



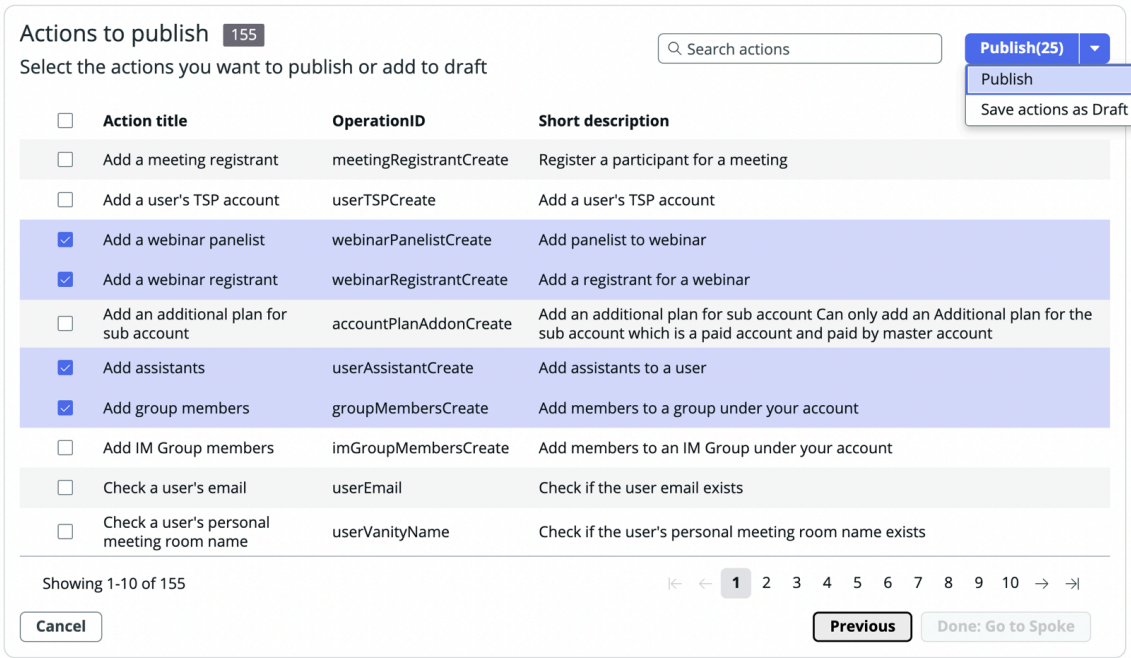
All the operations that can be performed using the OpenAPI Specification are listed.

17. Select the required operations.

You can search for the required actions by entering the required term in the search bar. The actions that match the specified search term are displayed.



18. Click **Publish**.
    Alternatively, you can also select **Save actions as Draft** to save the actions as draft, modify them as per your requirement, and publish them later.



19. Click **Done: Go to spoke** to go the Spokes page and view the publish status.
    - Actions with the OpenAPI step are created. For information about the OpenAPI step, see OpenAPI support in the REST step.
    - Action inputs and outputs are mapped.
    - Actions are published and listed in the spoke details page under **Actions** > **Published**.

    You can start using these published actions to create flows and subflows as per your requirement.

- If you have saved actions as draft, you can access these draft actions in the spoke details page under **Actions** > **Draft**.
- To view run-time information about the spoke activities, click **Spoke activity log** in the spoke details page. Every time a spoke activity is performed, the system generates its information as a spoke activity log. Click the required **Number** to view the activity log. Every operation in the spoke activity log has one of these status values:

Status	Description
new	An event for the operation is created and this operation will be executed soon.
error	The operation has failed to execute.
processing	The operation execution is in progress.
success	The operation has been executed successfully.

You can create flows and subflows in the spoke details page and use them in your integration. For more information, see Building flows and Building subflows.

Along with **Spoke activity log**, you can also view details of the available flows, subflows, and actions in the spoke details page.

Create spoke and build actions by importing a Postman collection

Automate an integration and generate reusable actions by importing a Postman collection.

Before you begin

- Install the Spoke Generator app from the ServiceNow Store.
- Role required: admin

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. Click **Create new** > **Spoke**.
3. On the Spoke Info screen, specify if you want to create the spoke in a new scope or an existing scope.

   a. If you choose to create the spoke in a new scope, select an image as the logo for your integration and fill in the fields.

   Here, we will go through an integration with Notion as an example.

Field	Description
Spoke name	Name to identify the custom spoke.
Description	Description about the custom spoke.

SPOKE INFO

Let's get started on your new spoke

Add a name and description that define your spoke. You can also add a thumbnail image.

Choose how you want to build your spoke

- (•) Create a spoke in new scope
- ( ) Create a spoke in existing scope

Spoke name * ⓘ

Notion Spoke

App scope name ⓘ

x_snc_notion_spoke

Description * ⓘ

ServiceNow integration with Notion.

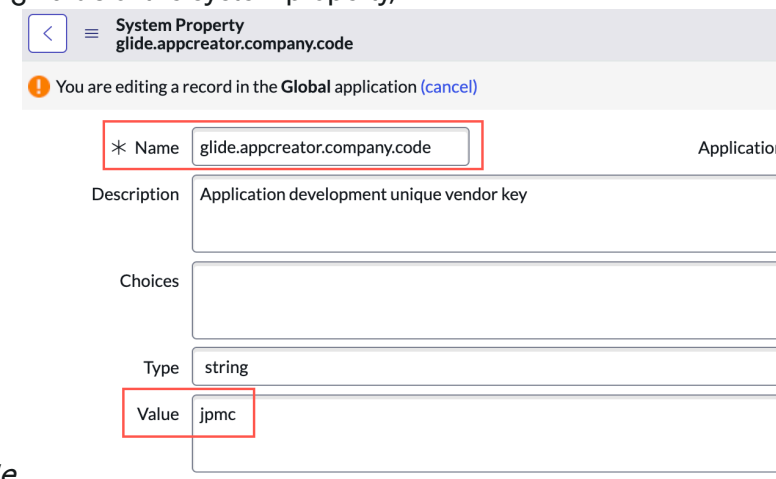
⊗ Remove image

Cancel

Continue

Note:

The value of **App scope name** is the format: `x_<company-code>_<spoke-name>_<spoke>`. By default, the <company-code> is, `snc`. You can configure the company code by configuring **Value** of the system property,



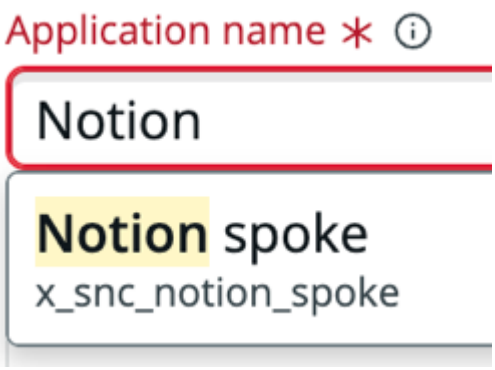
glide.appcreator.company.code.

This configured value is used when the value **App scope name** is

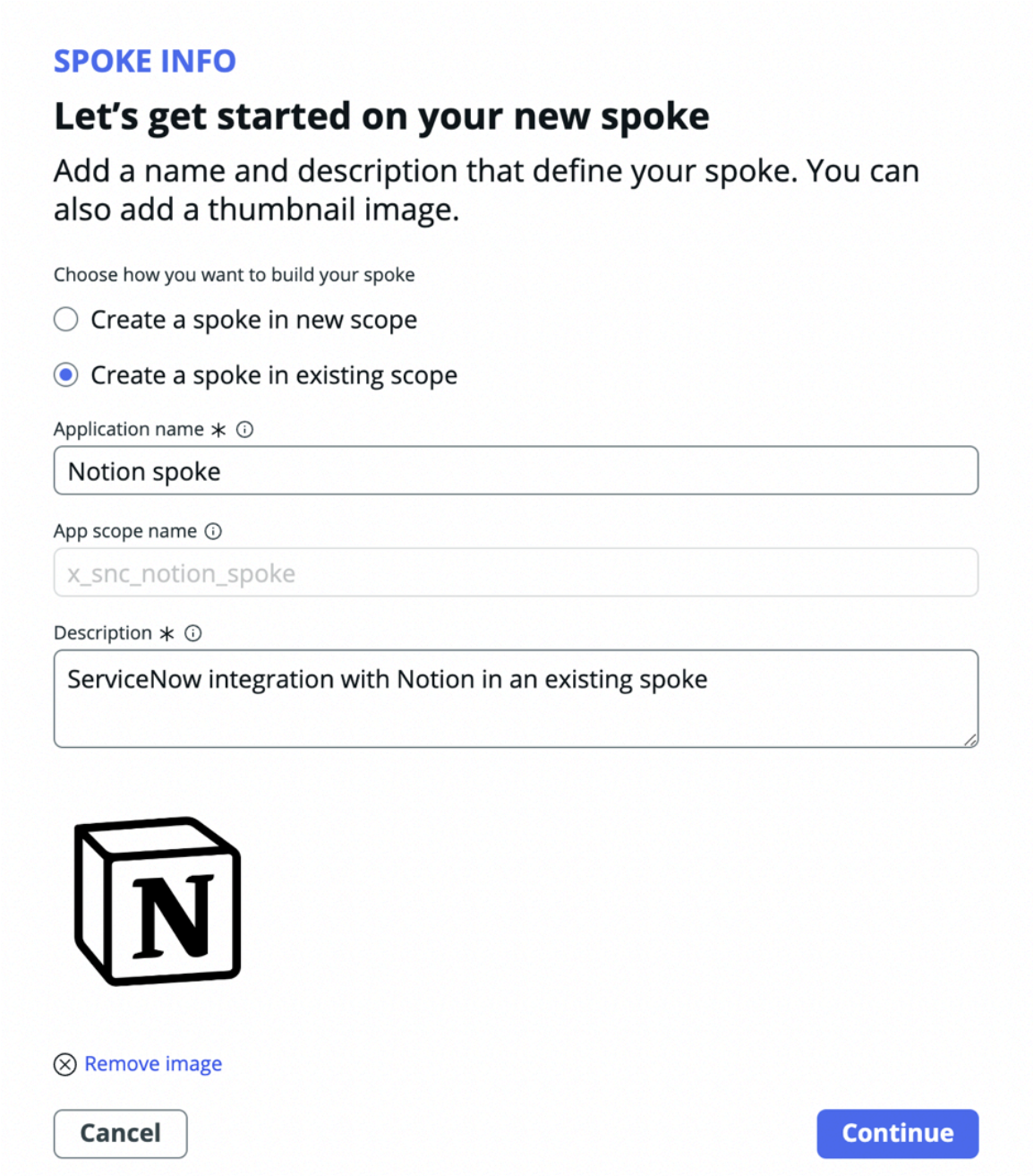
generated.

b. If you choose to create the spoke in an existing scope, select an image as the logo for your integration and fill in the fields.

Here, we will go through an integration with AWS as an example.

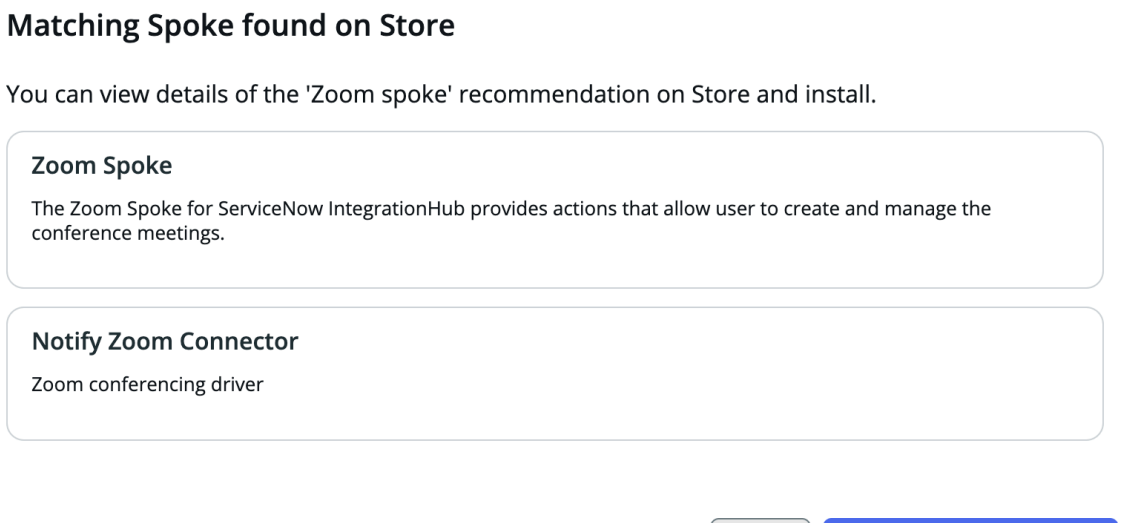
Field	Description
Application name	An existing application name or scope. 
App scope name	Scope name that is auto-populated based in the selected **Application name**.

Field	Description
Description	Description about the custom spoke.



4. Click **Continue**.

Based on the provided name and description, if there are any matching spokes on Store, the spoke details are displayed.



a. Click **View details on Store** to see the details of the matching spokes.
Details of the matching spokes are displayed in a new browser tab.

b. Install the spoke from the Store.
For more details, see Install a ServiceNow Store application.

c. After installing the spoke, navigate to the Workflow Studio tab.
The system displays the message `Have you installed a spoke from the Store?`.

d. Select one of these options and click **Continue**.

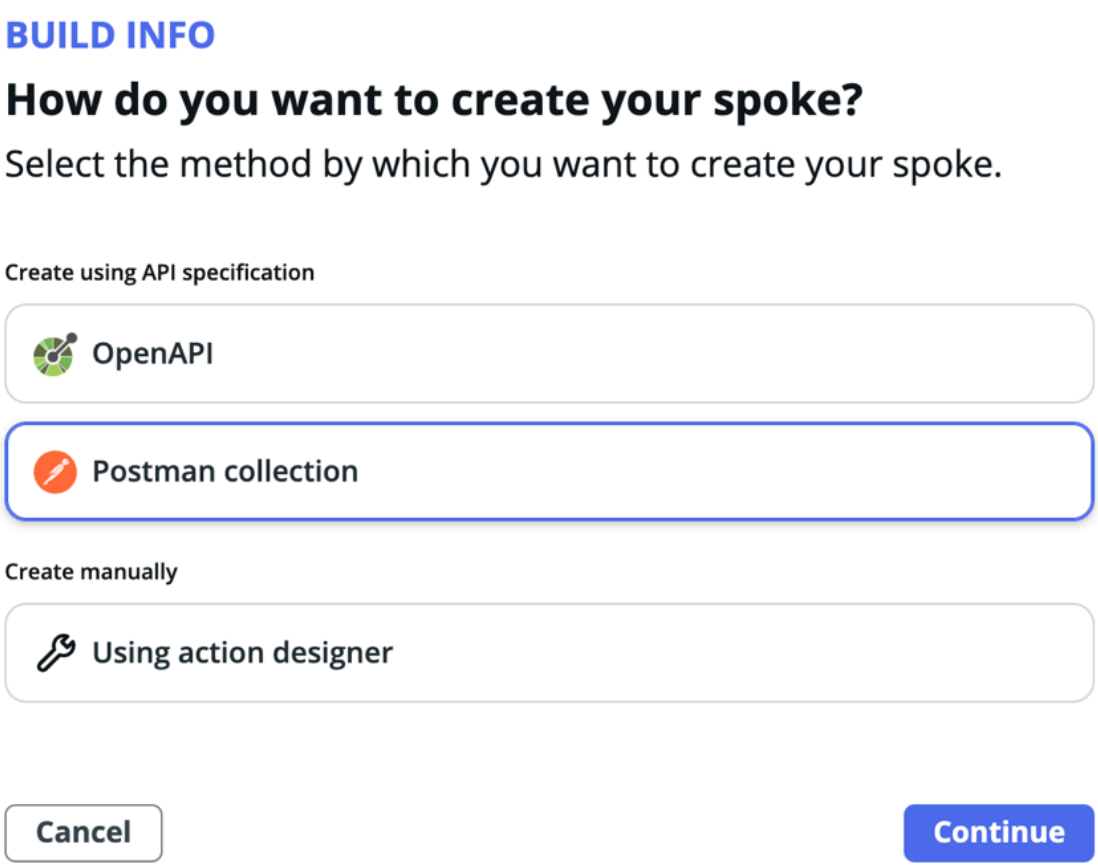
Option	Description
Yes, view the installed spoke.	Option to redirect to the **Spokes** dashboard under **Integrations**.
No, I will build custom spoke.	Option to continue with spoke creation.
No, I want to exit spoke creation.	Option to close the current tab.

5. Click **Skip** if you want to build the custom spoke.

Note:
These following steps are also applicable if you have selected the **No, I will build custom spoke.** option.

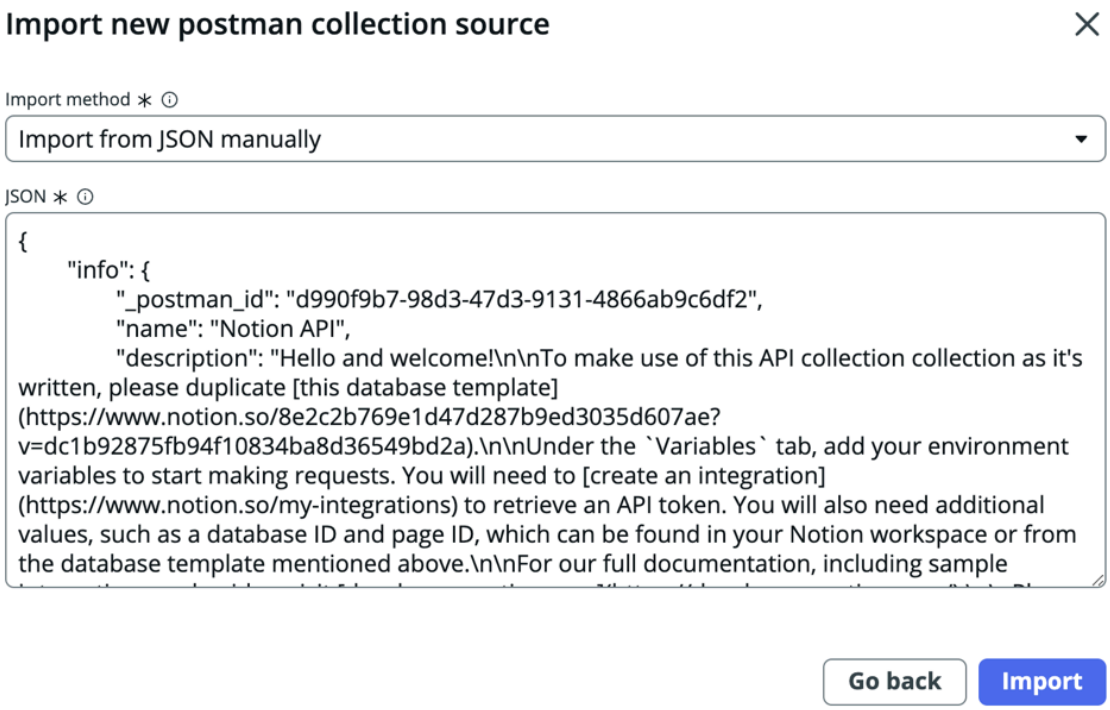
The spoke is created and a confirmation message is displayed.

6. On the Build Info screen, select the method using which you want to build your spoke.
You can choose to build your spoke using OpenAPI specification or Postman collection.
7. Select **Postman collection** and click **Continue** to import a Postman collection.



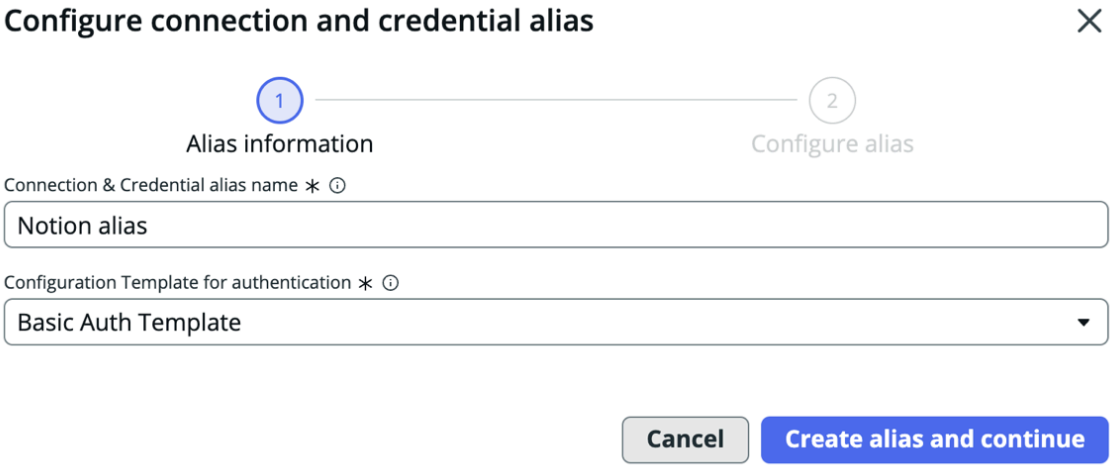
8. For **Postman collection source**, click **Import new**.
9. On the Import new postman collection source screen, perform one of these two steps.
   a. If you want to import using an URL, select **Import from URL** for **Import method** and specify the URL in **Postman collection URL**.

b. If you want to import manually using JSON code, select **Import from JSON manually** for **Import method** and provide the code in **JSON**.



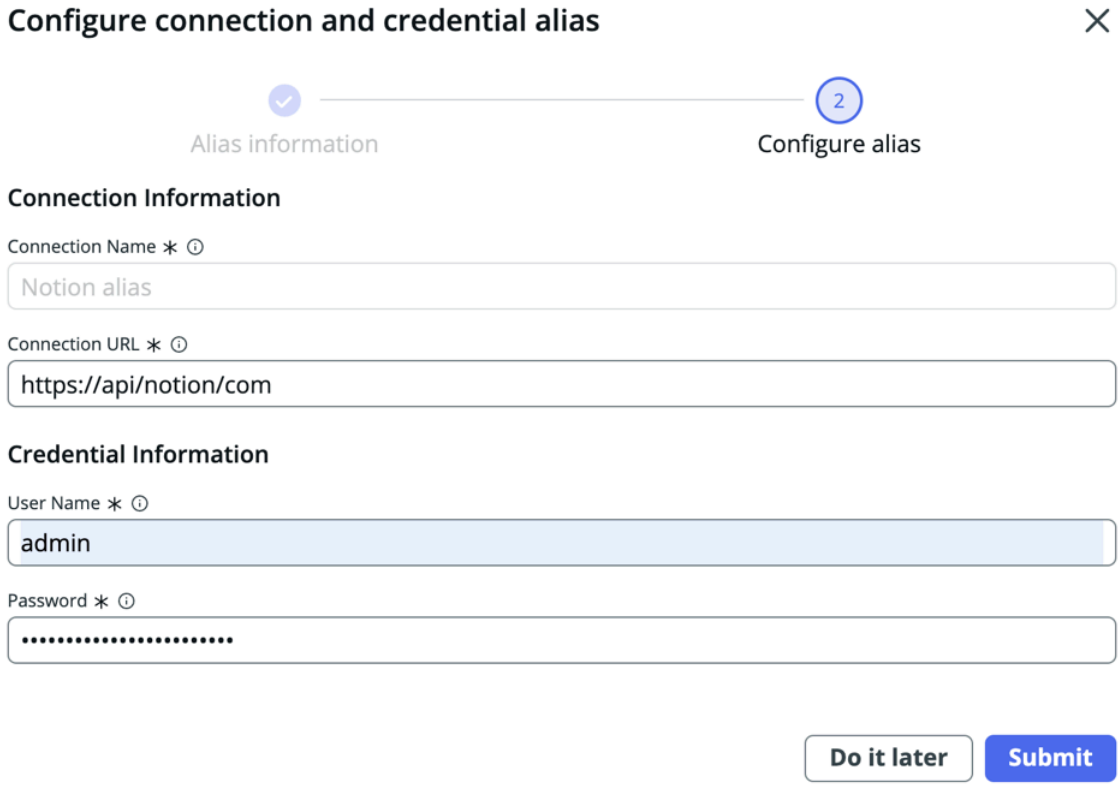
10. Click **Import**.
    The Postman collection source is imported.
11. For **Connection and credential alias**, click **Create new**.
12. On the Create new connection and credential alias screen, fill in the fields and provide the alias information.

Field	Description
Connection & Credential name	Name to identify the connection and credential alias record.
Configuration Template for authentication	Required authentication mechanism for this integration. Ensure that the authentication mechanism is compatible with the OpenAPI source. **Note:** The configuration template is auto-populated based on the OpenAPI specification you had provided. You can continue with the default option or change it as per your requirement.



13. Click **Create alias and continue**.
14. On the same screen, fill in the fields to configure the alias.

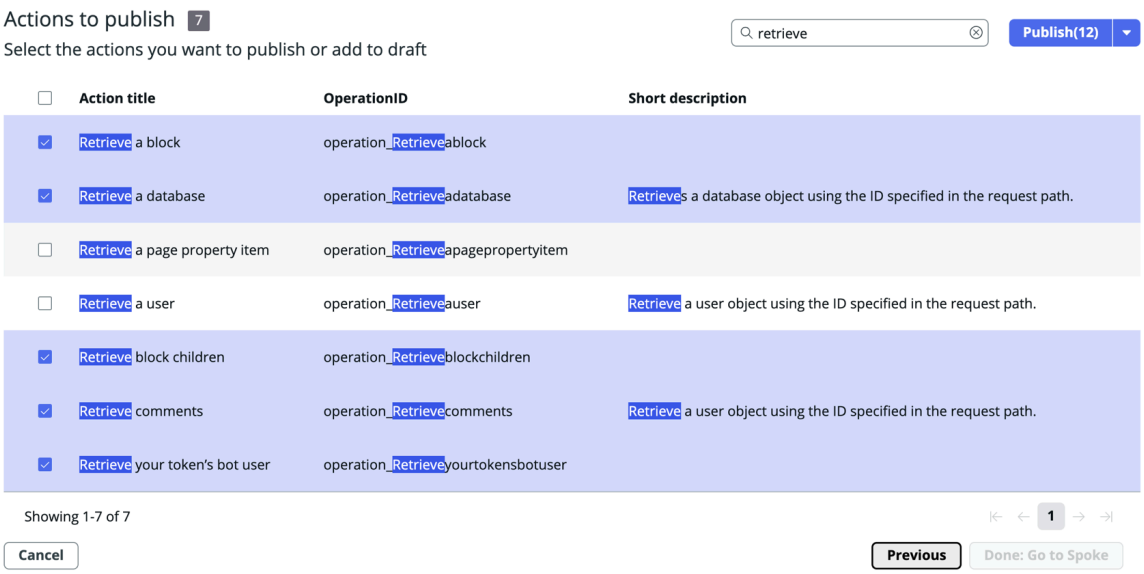
Field	Description
Connection Information	
Connection Name	Name to identify the connection record.
Connection URL	Base URL to connect to the third-party instance or server. This URL is auto-populated based on the Postman collection source you had provided.
Credential Information	
Based on the configuration template you had selected, the relevant credential fields are displayed. Provide the required values to configure the credential record.	



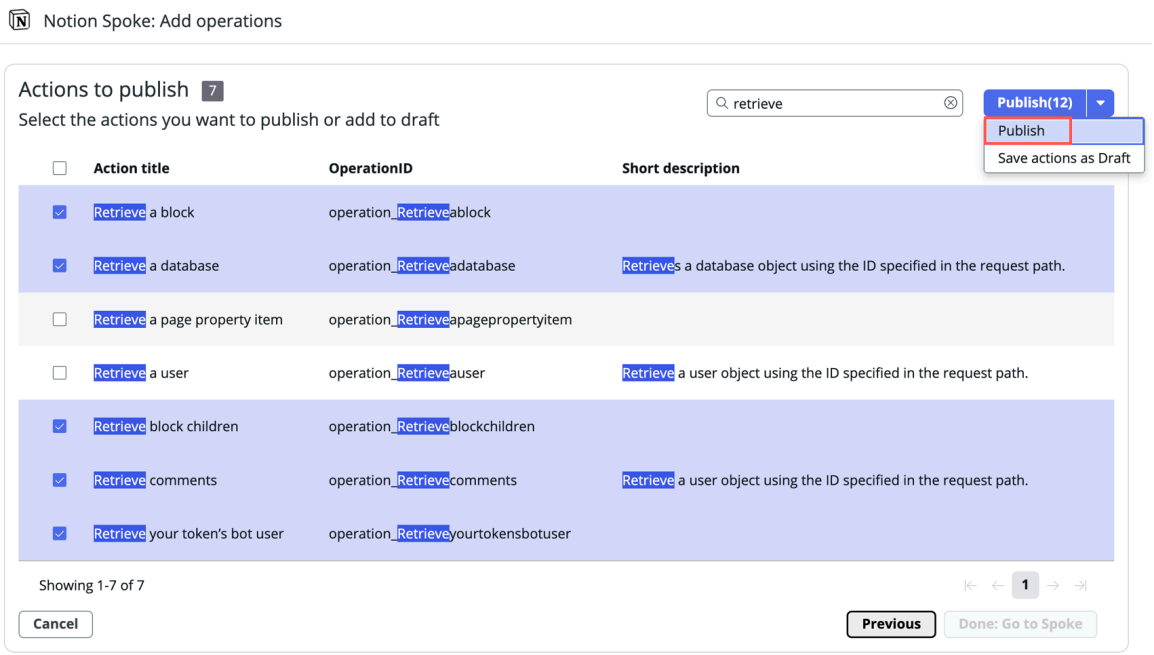
If you want configure the alias record later, click **Do it later**.

15. Click **Submit**.
    The connection alias record is created.
16. Click **Generate operations**.
    All the operations that can be performed using the OpenAPI Specification are listed.
17. Select the required operations.

    You can search for the required actions by entering the required term in the search bar. The actions that match the specified search term are displayed.



18. Click **Publish**.
    Alternatively, you can also select **Save actions as Draft** to save the actions as draft, modify them as per your requirement, and publish them later.



19. Click **Done: Go to spoke** to go the Spokes page and view the publish status.
    - Actions with the OpenAPI step are created. For information about the OpenAPI step, see OpenAPI support in the REST step.
    - Action inputs and outputs are mapped.
    - Actions are published and listed in the spoke details page under **Actions** > **Published**.

    You can start using these published actions to create flows and subflows as per your requirement.

    - If you have saved actions as draft, you can access these draft actions in the spoke details page under **Actions** > **Draft**.
    - To view run-time information about the spoke activities, click **Spoke activity log** in the spoke details page. Every time a spoke activity is performed, the system generates its information as a spoke activity log. Click the required **Number** to view the activity log. Every operation in the spoke activity log has one of these status values:

Status	Description
new	An event for the operation is created and this operation will be executed soon.
error	The operation has failed to execute.
processing	The operation execution is in progress.
success	The operation has been executed successfully.

You can create flows and subflows in the spoke details page and use them in your integration. For more information, see Building flows and Building subflows.

Along with **Spoke activity log**, you can also view details of the available flows, subflows, and actions in the spoke details page.

Use Now Assist to create spokes and build actions

Use Now Assist in Spoke Generator to create spokes and actions by providing the required third-party API documentation snippet as an input.

Capabilities

You can build custom spokes for the required third-party application by copying and pasting the required snippets from the API documentation. This feature is useful when the required third-party application doesn't have an OpenAPI Specification or Postman collection. This feature requires an Integration Hub subscription. For more information, see Legal schedules - IntegrationHub overview.

Supported versions

Spoke Generation skill is supported on these versions and later releases.

- Xanadu Patch 9
- Spoke Generator v4.0.0
- Workflow Studio v26.2.5
- Workflow Studio v26.2.6

Skill version

Spoke Generation skill v1.0.4 is the latest version.

App version

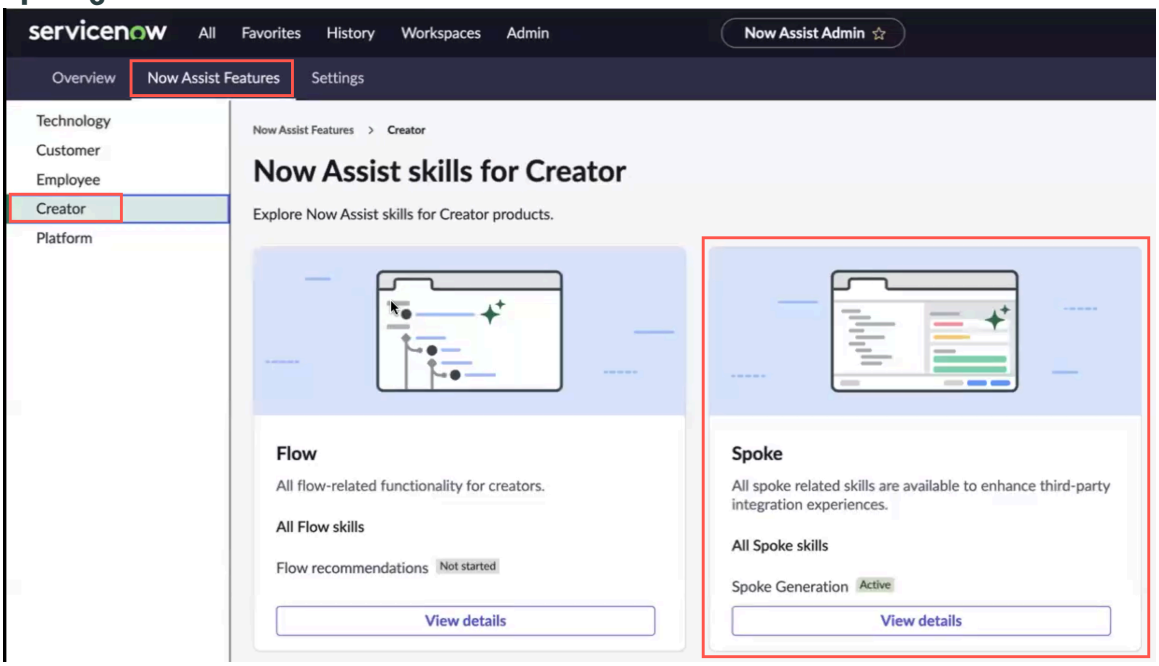
Now Assist for Spoke Generation v1.5.6 is the latest version.

Activation of the spoke generation skill

The spoke generation skill offers generative AI capabilities to build spokes. This skill is installed with the Now Assist for Creator (sn_now_creator) application. You can install this application from the ServiceNow Store website.

After installing the Now Assist for Creator (sn_now_creator) application, activate the Spoke Generation skill in the Now Assist for Creator feature.

Spoke generation skill in Now Assist Admin console



- For information about the Now Assist Admin console, see Now Assist Admin console.
- For steps to activate the required skill, see Activate a Now Assist skill.

Important:
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. Please check for availability updates in future releases.

Important:
Some Now Assist products/features are currently available only for customers in some regions. Be sure to check for availability updates in future releases.

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.

Install Now Assist for Creator

Install the Now Assist for Creator application to use the spoke generation skill.

Before you begin

- Role required: admin
- Review the Now Assist for Creator application listing in the ServiceNow Store for information on dependencies, licensing or subscription requirements, and release compatibility.
- Upgrade to Xanadu Patch 3 or later. For more information about this release, see Available patches and hotfixes.

Role required: admin

Procedure

1. Navigate to the Now Assist for Creator application on the ServiceNow Store .

   Important:
   Now Assist for Creator requires a separate subscription.

2. From the Now Assist for Creator application page, select **Request App**.
3. After approval has been granted, on your instance, navigate to **All** > **System Applications** > **All Available Applications**.
4. Find the Now Assist for Creator application (sn_now_creator) using the filter criteria and search bar.
5. Select **Install**.

What to do next

Configure Now Assist for Creator to turn on the spoke generation skill.

Turn on the spoke generation skill

Turn on the flow generation skill to use generative AI to create spoke.

Before you begin

Role required: admin

Important:
The spoke generation skill requires a separate subscription to Now Assist for Creator.

Procedure

1. Navigate to **All** > *Now Assist Admin* > **Features**.
2. In the workflow list, select **Creator**.
3. In the Spoke card, select **View details**.
4. Under All available Spoke skills, click **Turn on**.

Result

You can now create spoke and actions using Now Assist in Spoke Generator.

Create spoke and build actions using the spoke generation skill in Now Assist

Automate an integration and generate reusable actions by providing the required third-party API documentation snippet as an input.

Before you begin

Role required: admin.

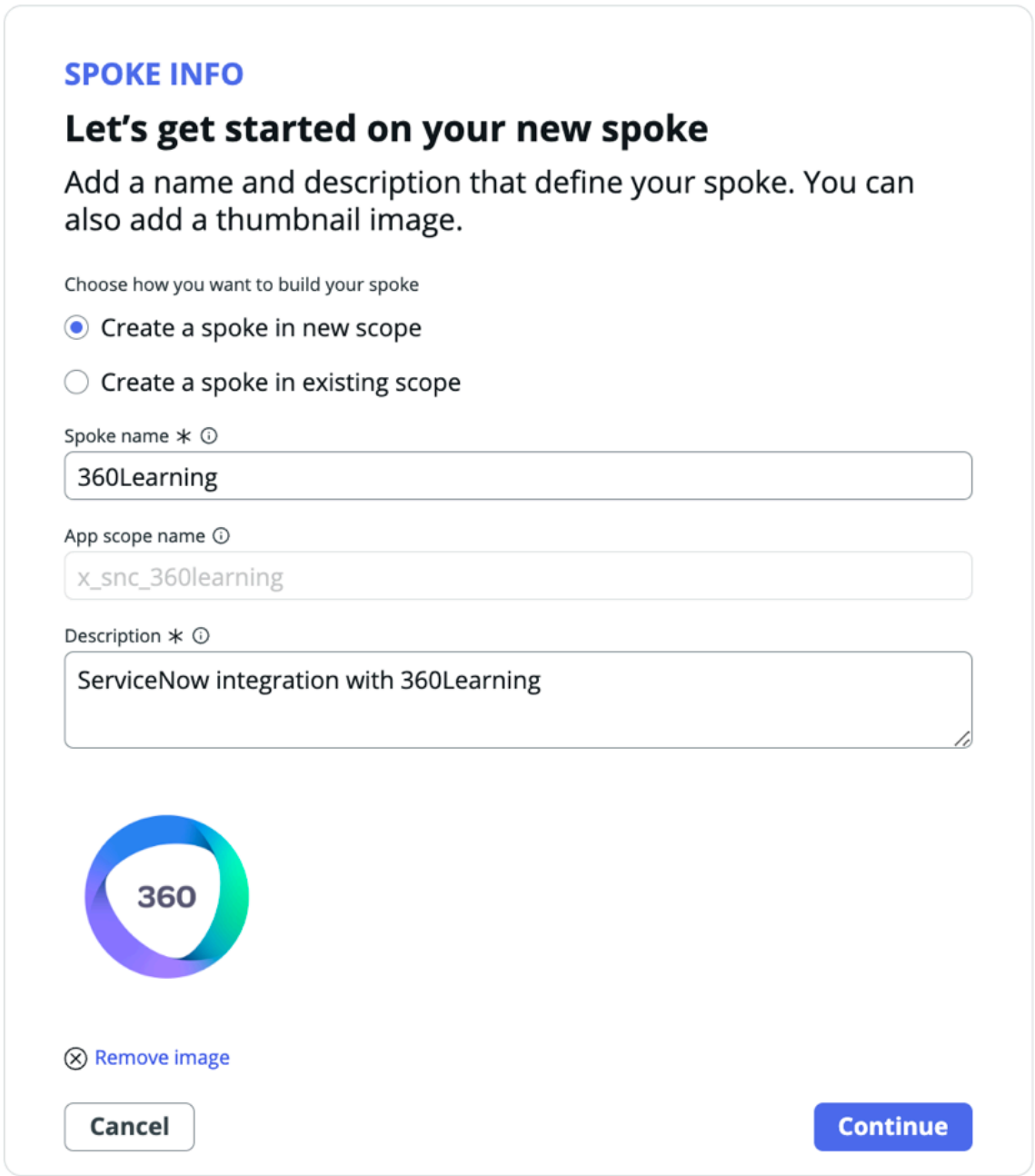
Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. Click **Create new** > **Spoke**.
3. On the Spoke Info screen, specify if you want to create the spoke in a new scope or an existing scope.

   a. If you choose to create the spoke in a new scope, select an image as the logo for your integration and fill in the fields.

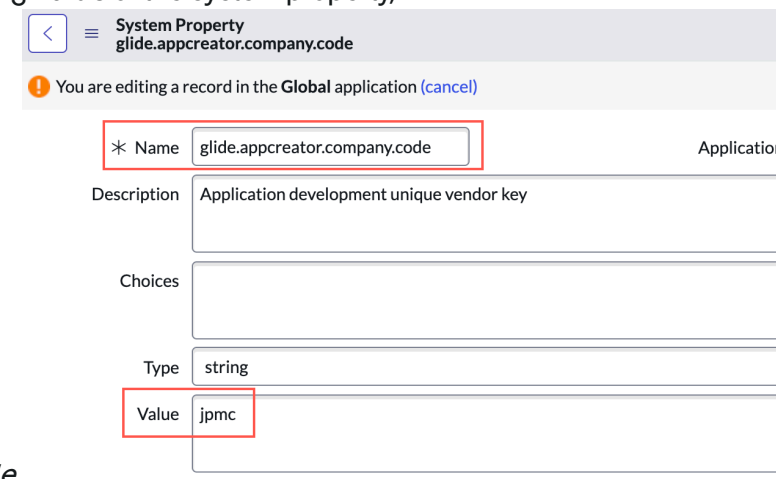
   Here, we will go through an integration with 360Learning as an example.

Field	Description
Spoke name	Name to identify the custom spoke.
Description	Description about the custom spoke.



Note:

The value of **App scope name** is the format: `x_<company-code>_<spoke-name>_<spoke>`. By default, the <company-code> is, `snc`. You can configure the company code by configuring **Value** of the system property,



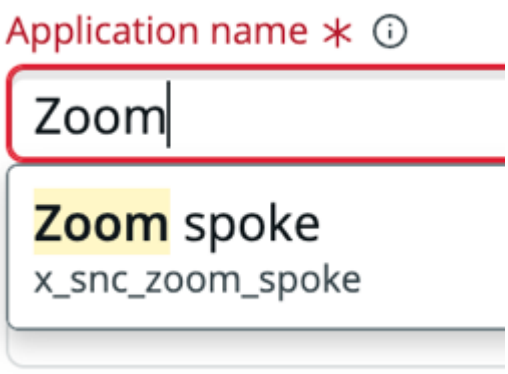
glide.appcreator.company.code.

This configured value is used when the value **App scope name** is

generated.

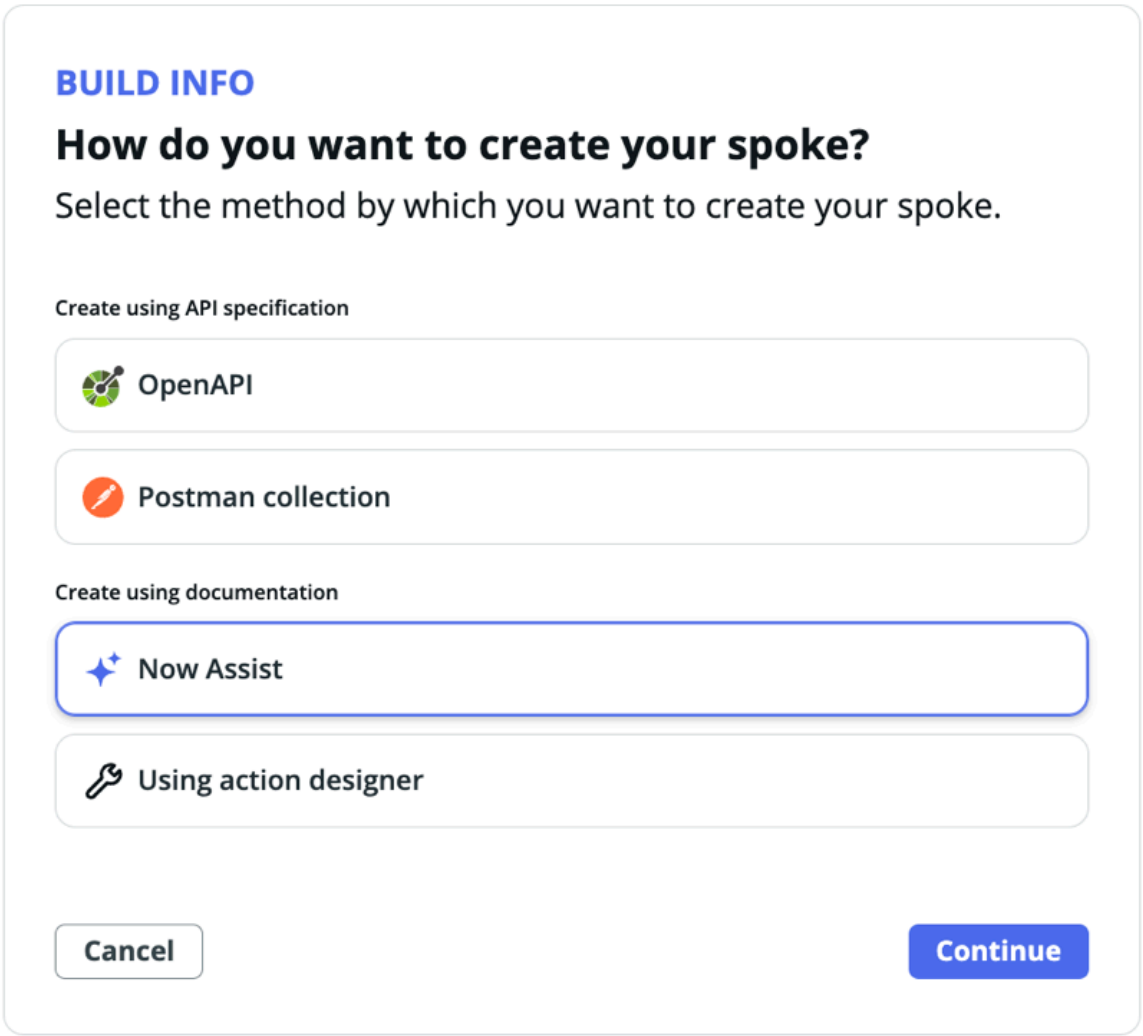
b. If you choose to create the spoke in an existing scope, select an image as the logo for your integration and fill in the fields.

Here, we will go through an integration with 360Learning as an example.

Field	Description
Application name	An existing application name or scope. 

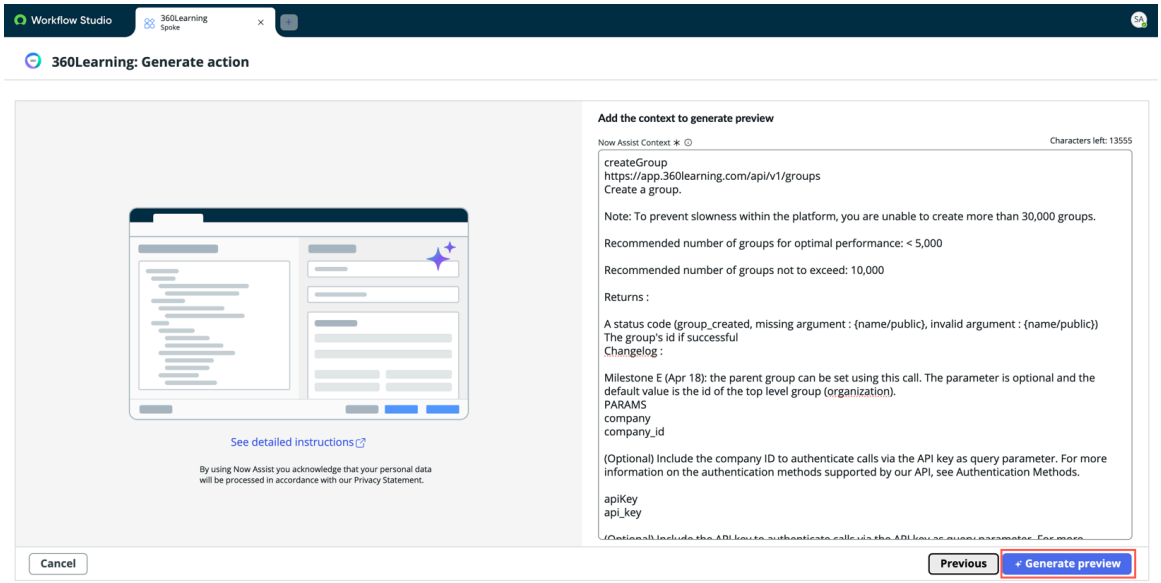
Field	Description
App scope name	Scope name that is auto-populated based in the selected **Application name**.
Description	Description about the custom spoke.

4. Click **Continue**.
5. On the Build Info screen, select the method using which you want to build your spoke.
6. Select **Now Assist** and click **Continue** to generate reusable actions by providing the required third-party API documentation snippet.

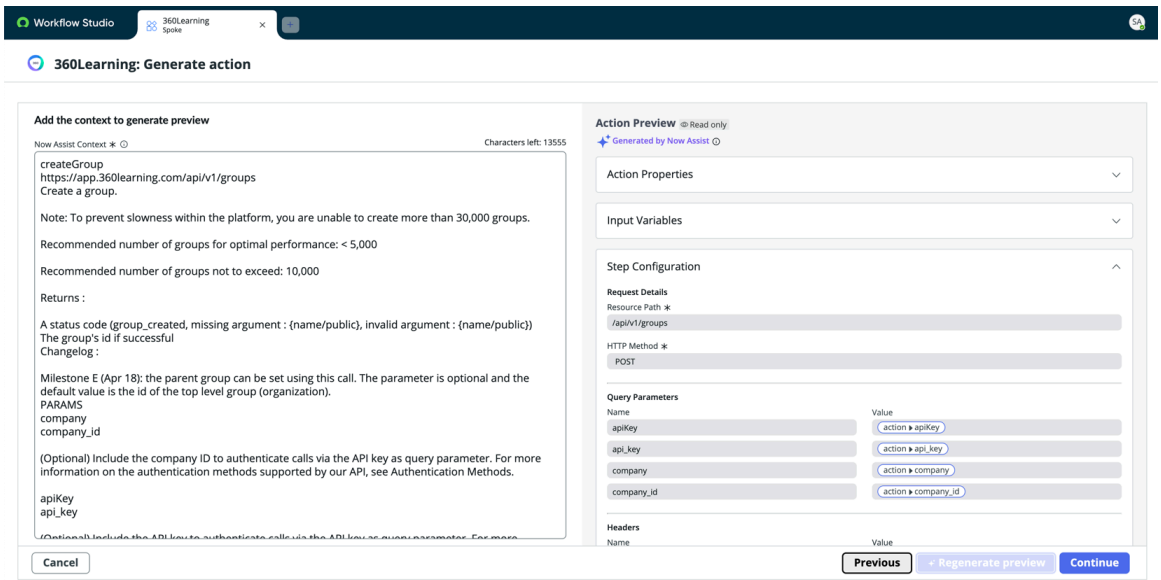


7. On the Generate action screen, paste the required content from the 360Learning API documentation in **Now Assist Context**.
   In this example, we will copy the documentation related to the createGroup action and paste it in **Now Assist Context**.

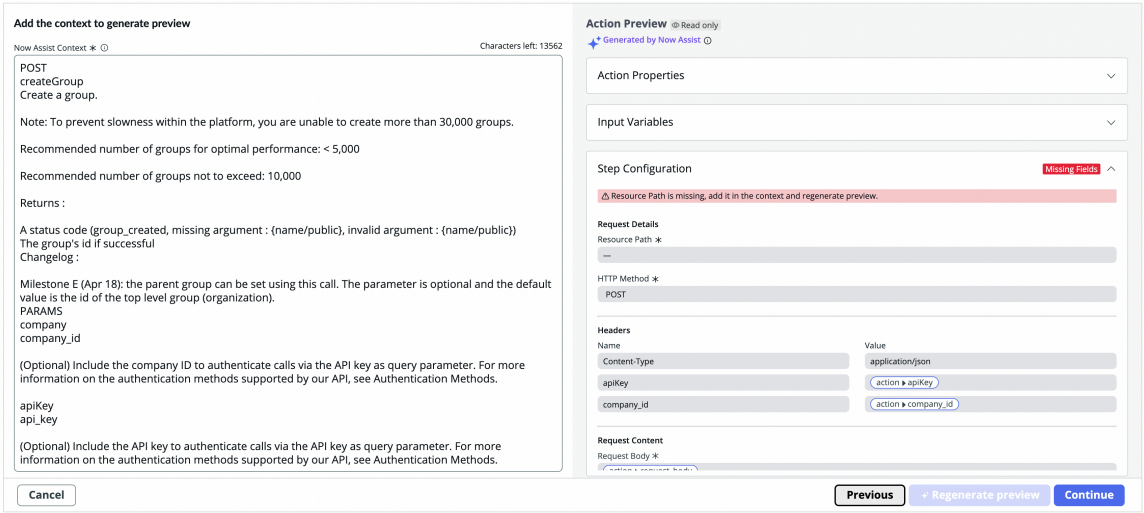
> **Note:**
> Ensure that you paste the documentation related to only one action at a time.



8. Click **Generate Preview**.
   The action preview is generated. Details of the action properties, inputs, outputs, and steps are displayed.

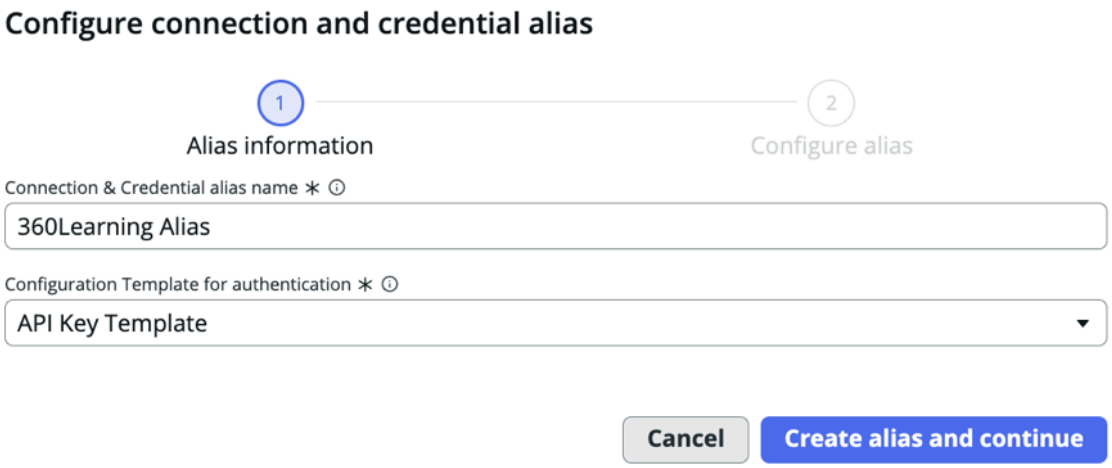


9. **Optional:** If you want to modify the generated action, modify the provided content in **Now Assist Context** accordingly and click **Regenerate preview**.
   If there are any missing fields in the content provided for **Now Assist Context**, an error message is displayed.



10. Click **Continue**.
11. On the Select connection and category screen, for **Connection and credential alias**, click **Create new**.
12. On the Create new connection and credential alias screen, fill in the fields and provide the alias information.

Field	Description
Connection & Credential name	Name to identify the connection and credential alias record.
Configuration Template for authentication	Required authentication mechanism for this integration. Ensure that the authentication mechanism is compatible with the third-party application.

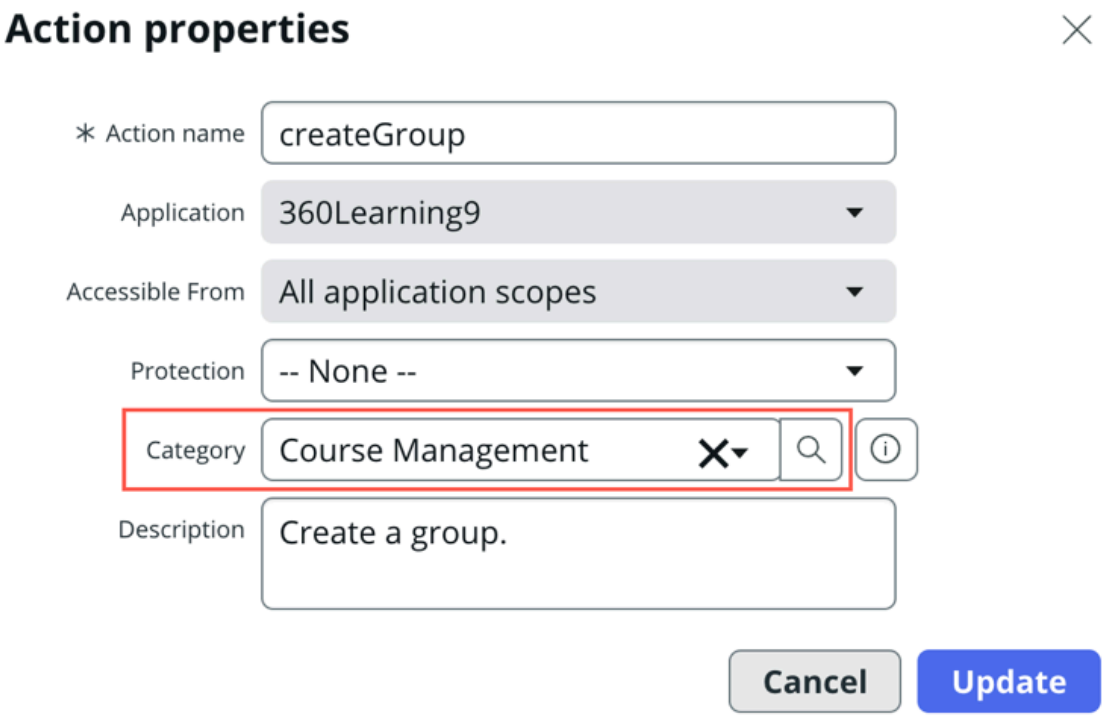


13. Click **Create alias and continue**.
14. On the Configure connection and credential alias screen, fill in the fields to configure the alias.

Field	Description
Connection Information	
Connection Name	Name to identify the connection record.
Connection URL	Base URL to connect to the third-party instance or server.
Credential Information	
Based on the configuration template you had selected, the relevant credential fields are displayed. Provide the required values to configure the credential record.	

If you want configure the alias record later, click **Do it later**.

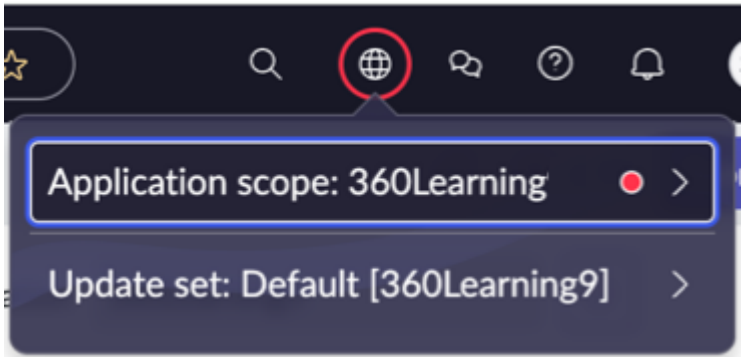
15. Click **Submit**.
    The connection alias record is created.
16. **Optional:** On the Select connection and category screen:
    - For a new spoke, leave the category empty. You can update the action properties later on, if needed.

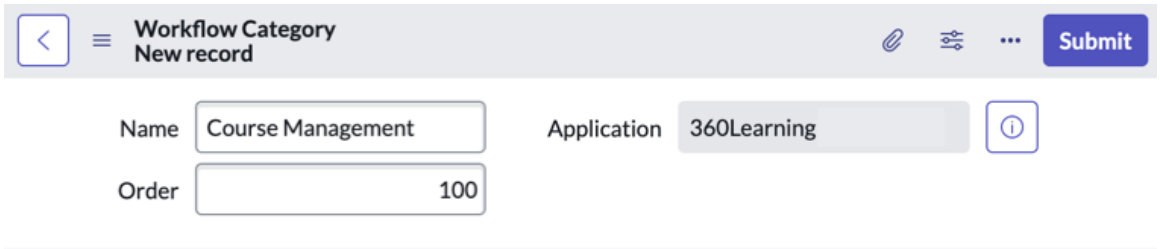


    (Optional)

    - For an existing spoke, assign an existing category.

      (Optional) If a category is unavailable, create the required action category for your integration in the Action Category [sys_hub_category] table. You must ensure that the category is created in the same **Application scope** as that of the spoke.





17. Click **Publish**.

> **Note:**
> If there are any missing fields, you can't publish the action. Instead you can save the action as a draft by clicking, **Save action as draft**.

The action is created. You can create another spoke action or navigate to the Spokes page.

If you have saved action as a draft, you can access these draft actions in the spoke details page under **Actions** > **Draft**.

Create spoke and build actions manually

Automate an integration and generate reusable actions manually.

Before you begin

- Install the Spoke Generator app from the ServiceNow Store.
- Role required: admin or action_designer

> **Note:**
> User with the action_designer role cannot add actions by importing the Open API Specification.

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. Click **Create new** > **Spoke**.
3. On the General Info screen, select an image as the logo for your integration and fill in the fields.

Field	Description
Spoke name	Name to identify the custom spoke.

Field	Description
Description	Description about the custom spoke.

4. Click **Create and continue**.
   The spoke is created and a confirmation message is displayed.
5. On the Build Info screen, select the method using which you want to build your spoke.
6. Select **Manually** and click **Continue**.
   The spoke details page is displayed.
7. Click **New action** to create an action.
   - If you want to create actions by importing the required OpenAPI Specification, select **From OpenAPI spec**. For information about selecting the required operations and creating actions, see Create spoke and build actions by importing an OpenAPI Specification.
   - If you want to create actions manually, select **Manually**. The Action Properties screen is displayed. You should create and publish the action in the Action Designer. For more information, see Building actions.

Add more actions to the custom spoke

Add more reusable actions to your custom spoke to automate an integration.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **Process Automation** > **Workflow Studio**.
2. Click the **Spokes** tab.
3. Click the required spoke tile.
4. On the spoke details page, click **New action**.
   - To create actions by importing the OpenAPI Specification, click **From OpenAPI Specification**. For information about selecting the required operations and creating actions, see Create spoke and build actions by importing an OpenAPI Specification.
   - To create actions manually, click **Manually**. The Action Properties screen is displayed. You should create and publish the action in the Action Designer. For more information, see Building actions.

   You can use the published actions to create flows or subflows as per your requirement.

Workflow Studio reference

Review additional information about Workflow Studio properties and component workflow applications.

Playbooks reference

Guides, lists, tables and more that you can reference as you work with Playbooks and their associated components.

Playbooks system properties

Review the system properties for Playbooks. You can configure these properties to control how the system handles Playbooks events.

The system properties for Playbooks provide advanced configuration options for how the system handles Playbooks events. To set Playbooks system properties, access the System Properties [sys_properties] table.

The following system properties relate to the Playbooks configuration:

Property	Description
com.glide.event_manager.process_automation.chunk_limit	Number of Process Automation events that the event handler can process in a single transaction
sn_agent_workspace.default_view_editable_tables	Default table views, as a comma-separated list, that you want to allow Workspace to render within a playbook card during runtime

Playbooks roles

Grant users one or more Playbooks roles to enable them to create triggers, playbooks, and activity definitions.

Roles

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

System administrators can grant users access to Playbooks by assigning delegated development permissions or directly assigning Roles. The following user roles are available for Playbooks:

Role	Description
playbook.admin	Enables users to: • Create, update, and delete trigger definitions. • Launch Workflow Studio to create, activate, edit, and delete playbooks. • Create, edit, and delete activity definitions. • View the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.
pd_author	Enables users to: • Launch Workflow Studio to create, activate, edit, and delete playbooks. • View all activity definitions. • View the Experience activity types (sys_pd_activity) and Experience activity

Role	Description
	properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.
pd_content_author	Enables users to: • Create, edit, and delete activity definitions. • Create, edit, and delete trigger definitions. • View the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.
pd_trigger_author	Enables users to create, update, and delete trigger definitions.
pd_operator	Enables users to view process executions, activity executions, and execution logs only.
pd_shared.user	Enables users to view the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.
pd_shared.admin	Enables users to edit the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.
pd_cancel	Enables users to cancel running playbooks without the playbook.admin role or write access to the parent record. For example if you want to grant an agent manager the ability to cancel playbooks, but not an agent.
pd_restarter	Enables users to restart active playbooks.
playbook.write	Enables users to: • Launch Workflow Studio to create, activate, edit, and delete playbooks. • View the Experience activity types (sys_pd_activity) and Experience activity properties (sys_pd_activity_type_prop) tables that are shared by Playbooks and Playbook Experience.
playbook.activity_def_read	Enables users to view all activity definitions.

Note:
Granting users Playbooks roles does not automatically allow them to access the Workflow Studio design environment. Granting users access to Workflow Studio may be helpful when creating activity definitions. For more information on Workflow Studio roles, see user access to Flow Designer.

Playbook statuses and activation states

View your playbook's status in the main header of the Playbooks builder. This status indicates whether the playbook is active or inactive.

The *Status* label in the main header of the Playbooks builder describes the state of configuration changes that were made to your playbook. The status also indicates whether your playbook is active or inactive. If your playbook is active, you can create an end user-facing view for it in Playbook Experience.

Playbook statuses include:

Draft

Your playbook is inactive. Any changes that you make are automatically saved to your draft playbook.

Published

Your playbook is active, you can create an end user-facing view for it in Playbook Experience. Any changes that you make after selecting **Activate** are automatically saved. However, these changes aren't activated in your published playbook until you select **Activate** again.

Playbook status and activation state flow

Creating a new playbook sets its status to Draft. Once you activate your playbook, its status changes to Published. If you make changes after activating, your new changes won't be published until you activate your playbook again.

Process executions

A process execution is a single, runtime instance of a playbook. Process execution records provide runtime information about playbooks, such as the current state and input record.

A process execution represents a runtime execution of your playbook. Each time a playbook is triggered, a record is automatically in the Process Executions [sys_pd_context] table.

To access the records for an In Progress process execution, navigate to **Process Automation** > **Process Automation Administration** > **Active Processes**. Alternatively, you can see the process executions for all playbooks that ran today by navigating to **Process Automation** > **Process Automation Administration** > **Today's Executions**.

Fields

By default, each process execution record contains the following information:

Process Execution record fields

Field	Description
Name	Name of the playbook that triggered this process execution
Created	Date and time when the playbook triggered
Input Record	Table name and record number that triggered this process execution
State	Current status of the overall process execution. For more information, see Process execution states.

Process execution states

A process execution record can have one of the following states:

State	Description
Queued	The playbook triggered, but the system hasn't started running the process execution yet.
In Progress	The playbook triggered, and the process execution is currently running. One or more activities in the playbook have a state of Ready or In Progress.
Complete	The playbook triggered, and the process execution is done running. All activities in the playbook have a state of Skipped or Complete.
Error	The playbook triggered, but an activity has an Error state. Errors can occur when the action, subflow, or flow specified in the activity definition's automation plan fails to run.
Cancelled	A user with the admin or playbook.admin role explicitly canceled the process execution, and the execution has stopped running.

Activity executions

Activity execution records provide runtime information about activities in a playbook, such as the activity's current state and associated record.

An activity execution represents a runtime execution of your activity. Each time one of your playbooks triggers, records are automatically created for each activity that runs within the triggered playbook.

To access the activity execution records for an In Progress playbook execution, navigate to **Process Automation** > **Process Automation Administration** > **Active Processes**. Alternatively, you can see the activity executions for all playbooks that ran today by navigating to **Process Automation** > **Process Automation Administration** > **Today's Executions**. Select a playbook execution record from the list. Then, you can view the associated activity execution records in the Activity Executions related list.

Fields

By default, each record in the Activity Executions related list contains the following fields:

Activity Execution fields

Field	Description
Label	Name of the activity
Stage	Stage in which the activity runs
State	Execution status for the activity. See Activity execution states.
Activity Type	Experience type for the activity. See Experience types.
Associated Record	Record whose data renders within the playbook card.
Execution index	Sequential order in which the activity runs, starting with 1 (one).

Activity execution states

Activity execution states indicate the status of an activity in a triggered playbook. Activities that render in a playbook card can display this state to agents. An activity execution record can have one of the following states:

State	Description
Pending	The playbook triggered, but the activity execution is waiting on preceding activities to complete before it can start running.
Ready	The playbook triggered, and the activity will start running soon.
In Progress	The playbook triggered, and the activity execution is running.
Complete	The playbook triggered, and the activity execution is done running.
Skipped	The playbook triggered, but a user chose to skip this activity execution and move on to the next activity. Also, if the activity contains a condition that evaluates to false, the system skips the activity.
Error	The playbook triggered, but an error with the activity's automation plan or activity experience occurred. Errors can occur when the underlying action or subflow in the activity's automation plan fails to run.
Cancelled	A user with the admin, flow_designer, or action_designer role explicitly canceled the underlying action or subflow in the activity's automation plan.

Create Task activity

Create a task record from previously gathered or generated data.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered.

Input	Type	Description
		• **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Assigned To	Reference.User [sys_user]	User responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Short Description	String	Summary of the task to complete.
Wait for user input	Choice	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assignment Group for this Process Step	Reference.Group [sys_user_group]	Assignment group allowed to perform this playbook activity. If you don't set any values for **Assignment Group** or **Assigned To**, any user can read and edit the collected data. The Assignment group and Assigned to fields limit who has access to do so. To specify only individual users,

Input	Type	Description
		use the **Assigned To** field. The same users do not need to be specified in both fields. **Note:** By default, these fields are mapped to the **Assignment Group** and **Assigned To** fields of the trigger record. This means that users assigned to work on the parent record have access to submit, view and edit the collected data by default.
Assigned to this Process Step	Reference.User [sys_user]	User allowed to perform this playbook activity. If you don't set any values for **Assignment Group** or **Assigned To**, any user can read and edit the collected data. The Assignment group and Assigned to fields limit who has access to do so. To specify only individual users, use the **Assigned To** field. The same users do not need to be specified in both fields. **Note:** By default, these fields are mapped to the **Assignment Group** and **Assigned To** fields of the trigger record. This means that users assigned to work on the parent record have access to submit, view and edit the collected data by default.
Task Table	Table Name	Name of the task table extension in which you want to create a task.
Add Field	Template Value	Field values to assign when creating the task record.
Fields to show after creation	String	Comma-separated list of fields to display for a created record.

Input	Type	Description
Wait for Task completion	True/False	Pause the playbook until the created Task record is complete.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Table	Table Name	Table containing new record.
Record	Reference.Task[task]	Reference to record created.

Instruction activity

Display a simple message to guide end users through your playbook.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running:

Input	Type	Description
		• **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Message	String	Message text to display to end users.
Wait for user input	Choice	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Description
Record	Reference to the record associated with the activity.

User Form activity

Use this activity to surface a record to the end user. This activity requires you to select a table and record, and the desired form view that should be surfaced to the end user in your playbook. The end user can then interact with this record accordingly. Display a form during runtime to collect input values for your playbook.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered.

Input	Type	Description
		• **After specific activities**: Your activity starts running after specified activities have finished running.
Tagline	String	Header tagline to display to end users.
Record fields	Reference	Any additional record fields you want in the form. **Note:** This field cannot be edited during a playbook run.
Associated table	Table Name	Table containing activity or playbook data.
Associated record	Reference	Reference to the record you want to display.
Form View	String	Form view to use for data collection. If you don't provide a form view, the system uses the default view. Use the **new tab** button  to open and edit a form view with Form Builder in a new Workflow Studio tab. **Note:** Many form views are not supported in Workspace.
Form Fields	String	Comma-separated list of fields names to display on the card for the form.
Attachment source	Choice	The attachment record containing the attachments that end users can upload during runtime.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assignment Group	Reference.Group [sys_user_group]	Group responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Assigned To	Reference.User [sys_user]	User responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Display Order	Choice	The order in which the activity card appears during a playbook run. **Note:** **Show additional options** to see this field.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Restart rules	Choice	What an activity or stage does during restart: • **Skip on restart**: The stage or activity only runs during a playbook's initial run. It does not run on restart.

Input	Type	Description
		Note: This setting is helpful if you don't want new tasks or records to be created during a restarted run, because the original execution and resulting record is still relevant. • **Run always**: The stage or activity always runs, whether during an initial or restarted run. • **Skip on first run**: The stage or activity runs only on restart. It never runs during an initial run. To learn more about enabling and configuring restart for playbooks, stages, and activities, see Enable and Configure Restart for Playbooks. **Note:** **Show additional options** to see this field.
Experience Status Table	Table Name	Table containing the Experience Status Record. By default, set to the table containing the flow data record created by this activity's underlying flow, subflow, or action. **Note:** **Show additional options** to see this field.
Experience Status Record	Reference	The flow data record with the status that you want to map with this activity's state. Changes made to the state in the activity card are mapped to the flow data record, and from the flow data record to the activity card. By default, this is the data record created by this activity's underlying flow, subflow, or action.

Input	Type	Description
		Note: **Show additional options** to see this field.
Icon	Reference.Icon [st_sys_design_system_icon]	Displays an icon on the activity card. Start entering the icon name and select the icon you want to appear. **Note:** **Show additional options** to see this field.
Title	String	Title displayed to end users in the activity card. You can enter text or use the pill picker. **Note:** **Show additional options** to see this field.
Description	Reference	Details to display to end users in the activity card. This is not a string field, use the pill picker to choose a description to show in the activity card. **Note:** **Show additional options** to see this field.
Pending State Title	String	Title displayed to end users for a pending activity. You can enter text or use the pill picker. **Note:** **Show additional options** to see this field.
Pending State Description	String	Details to display to end users in a pending activity. **Note:** **Show additional options** to see this field.
Attachments read only	True/False	If enabled, prevents end users from renaming or deleting the existing attachments.

Input	Type	Description
		Note: **Show additional options** to see this field.
Footer	String	Footer text to display to end users. **Note:** **Show additional options** to see this field.
Show Attachments	True/False	Option to include attachments in the user form. **Note:** **Show additional options** to see this field.
Show SLA	True/False	Option to show SLA countdown details for the task. **Note:** **Show additional options** to see this field.
Show Checklist	True/False	Option to show record checklist items. **Note:** **Show additional options** to see this field.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Table	Table Name	Name of the table associated with the activity.
Record	Reference.Task[task]	Reference to the record associated with the activity.

Placeholder activity

Display an activity card in Playbooks to indicate what a future activity does.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Edit	Choice	Activities that you can select to replace the placeholder.
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome. You can use this input to describe the activity needed to replace this placeholder.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered.

Input	Type	Description
		• **After specific activities**: Your activity starts running after specified activities have finished running.

Outputs

This activity does not produce output.

Design considerations

Add placeholders activities to skip activity configuration

Playbooks designers can use placeholder activities to communicate the intent of an activity or playbook without having to do any configuration. Playbook designers can defer configuration to another time or delegate the configuration to another user who may better understand the playbook data model.

Edit placeholder activities to replace them with other activities

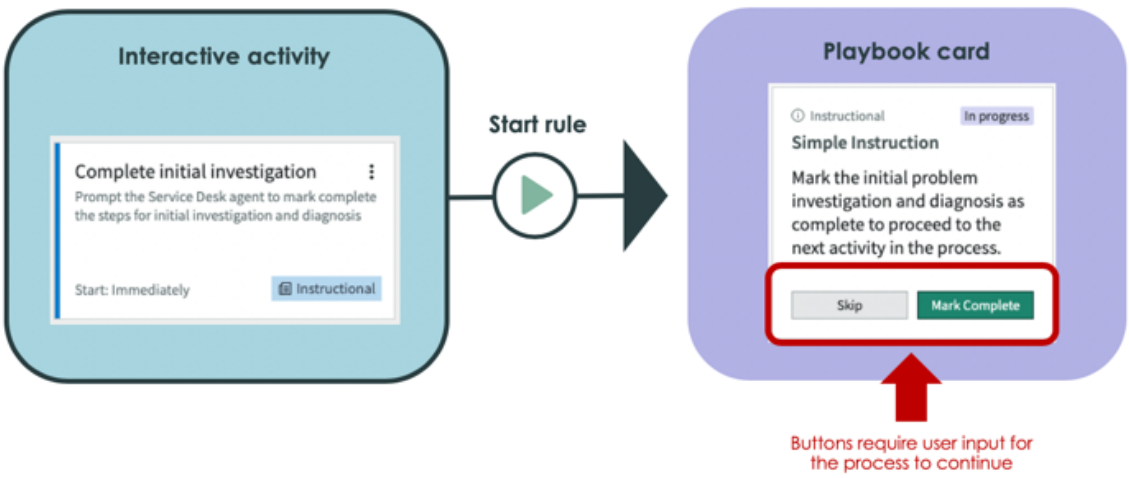
Placeholder activities help playbook designers create playbook, but do not produce any output or advance a playbooks when run. Edit Placeholder activities when you are ready to replace them with another activity type and configure them.

Interactive activities

An interactive activity prompts a user for input in your playbook as it runs.

An interactive activity requires user input to complete. Agents can provide the input within the activity card during the playbook run. For example, if your activity requires a user to enter work notes for a record, you can configure the activity inputs to prompt a playbook agent to add work notes information in the playbook card.

Interactive activities



Interactive activities turn into activity cards that require user input in a Playbook experience. Playbook agents must interact with these cards so that the playbook can continue running.

To learn how to design a playbook with interactive activities, see design an automated process.

Adobe Acrobat Sign activities

Enable agents and fulfillers to collect electronic signatures during a playbook run, via Workflow Data Fabric's Adobe Acrobat Sign spoke.

Roles and availability

- These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.
- You must have access to Workflow Data Fabric's Adobe Acrobat Sign spoke.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts

Input	Type	Description
		running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Group record	Choice	Record that represents the group connection and credential alias.
Attachment	Choice	File or document to be signed.
Participants email	String	List of signatory email addresses.
Email body	String	Message displayed to signatories.

Outputs

Your signed document, available for review.

Advanced Instruction activity

Display detailed instructions to guide end users through your playbook.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.

Input	Type	Description
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Title	String	Title displayed to end users.
Description	String	Details to display in the message to end users.
Wait for user input	Choice	Option to pause the playbook until the end user manually

Input	Type	Description
		completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assignment Group	Reference.Group [sys_user_group]	Group responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Assigned To	Reference.User [sys_user]	User responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Tagline	String	Header tagline to display to end users.
Footer	String	Footer text to display to end users. **Note:** **Show additional options** to see this field.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Record	Reference.Task [task]	Reference to the record associated with the activity.

Checklist Task activity

Prompt an agent to complete all items in a task checklist.

The key input for this activity is the **Checklist Template** field under the Inputs section. Before you can provide this input, you must:

- Create a checklist, and
- Create a checklist template for the target task table.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Common properties

These properties are common to all to activities in Playbooks.

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

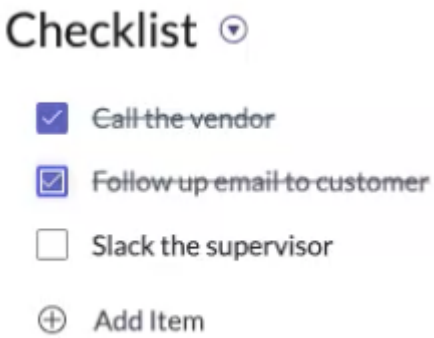
Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running:

Input	Type	Description
		• **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.

Inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Skip Assignment Group	Reference.Group [sys_user_group]	Group that is allowed to skip this activity when **Can Skip** is enabled. **Note:** If **Can Skip** is enabled, and you do not specify a **Skip Assignment Group**or **Skip Assigned To**, all users will be able to skip this activity.
Skip Assigned To	Reference.User [sys_user]	User that is allowed to skip this activity when **Can Skip** is enabled.

Input	Type	Description
		Note: If **Can Skip** is enabled, and you do not specify a **Skip Assignment Group**or **Skip Assigned To**, all users will be able to skip this activity.
Checklist Template	Reference.Checklist Template [checklist_template]	Template for the checklist to create for each run. Before you can provide this input, you must: • Create a checklist, and • Create a checklist template for Visual Task Board tasks
Task	Reference.Task [task]	Reference a record to associate with the checklist activity. The checklist is also displayed in the associated task record. **Note:** When a specific task record is associated with the checklist, the state of the checklist does not change if you run the playbook again. When you don't provide a specific task record, a new private task record is created for each run. 
Show SLA	True/False	Option to show SLA countdown details for the task.

Input	Type	Description
		Note: **Show additional options** to see this field.
Can Skip	True/False	Option to allow agents to skip the checklist and to continue through the playbook.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Checklist Task	Reference.Task [task]	Task record that updates when this activity completes.

UI Layouts

Set properties for how the activity renders during a playbook run.

Collect User Data activity

Collects inputs from a user during a playbook run to use later in the playbook.

Important:
As of the 26.1 release, the **Collect user data** activity is no longer available in the activity picker. The activity will continue to function wherever it is used, but for new activities use the **Questionnaire** activity instead. The **Questionnaire** activity does not require you to create a data definition. To learn more about the **Questionnaire** activity, see Questionnaire activity.

Use the Collect User Data activity if:

- You don't have a table already,
- You don't need to run reports on the collected data,
- And you don't need to use the data outside of the playbook.

If you already have a table to store the collected data, use the User Form activity.

The key input for this activity is the **User form for data collection** field, under the Inputs section. To provide this input, you must create a data definition first.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

During a playbook run, you can use data definitions to potentially:

- Collect a shipping address, then reference the address when generating a shipping label.
- Ask the user "yes" or "no" questions, and determine subsequent activities based on the user's responses.

Common properties

These properties are common to all to activities in Playbooks.

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity

Input	Type	Description
		or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.

Inputs

Many of these inputs are common to activities in Playbooks. The key input for this activity is **User form for data collection** field.

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assignment Group for this Process Step	Reference.Group [sys_user_group]	Assignment group allowed to perform this playbook activity. If you don't set any values for **Assignment Group** or **Assigned To**, any user can read and edit the collected data. The Assignment group and Assigned to fields limit who has access to do so. To specify only individual users, use the **Assigned To** field. The same users do not need to be specified in both fields. **Note:** By default, these fields are mapped to the **Assignment Group** and **Assigned To** fields of the trigger record. This means that users assigned to work on the parent record have access to submit, view and edit the collected data by default.
Assigned to this Process Step	Reference.User [sys_user]	User allowed to perform this playbook activity. If you don't set any values for **Assignment Group** or **Assigned To**, any user can read and edit the collected data. The Assignment group

Input	Type	Description
		and Assigned to fields limit who has access to do so. To specify only individual users, use the **Assigned To** field. The same users do not need to be specified in both fields. **Note:** By default, these fields are mapped to the **Assignment Group** and **Assigned To** fields of the trigger record. This means that users assigned to work on the parent record have access to submit, view and edit the collected data by default.
User form for data collection	Choice	Type of input form used to collect data. Playbook authors define the data that they want agents or fulfillers to collect during a playbook run in the `sys_flow_data_definition` table. When the information is collected, it is stored in the `sys_flow_data` table for use later in the playbook run, instead of in the record table. To define the data you want an agent or fulfiller to collect, see Create a Data Definition.
Wait for user input	Choice	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Record	Reference.Flow Data	Reference to record containing collected data. Use the pill-picker to dot-walk to

Output	Type	Description
		Outputs > **Record** > **Vars** to see all collected data. To learn more about the pill-picker, see Dot-walking examples.

UI Layouts

Set properties for how the activity renders during a playbook run.

Create Record activity

Pause the playbook and prompt the end user to create a record in a form view. Use this activity to allow the end user to create a record. This activity requires you to configure the desired table for which record to create, and the desired form view that the end user will see when creating the record.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running:

Input	Type	Description
		• **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Table	Table Name	Table in which to insert new record.
Create Record View	String	Form view to use for record creation. If you don't provide a form view, the system uses the default view. Use the **new tab** button  to open and edit a form view with Form Builder in a new Workflow Studio tab. **Note:** Many form views are not supported in Workspace.
Template Fields	Template Value	Field values to set during record creation.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Table	Table Name	Table containing new record.
Create Record View	String	Form view used for record creation.
Record Created	Sys ID	Unique identifier of the record that this activity created.

Design considerations

Create form views for activities that you want to render in a playbook during runtime.

Use a form view to display only the fields your users need to create a record. Your view should display required fields or those fields validated by other business logic. See View Management.

Run non-interactive activities before interactive activities

While a Create Record activity interactively gathers data from users, it prevents the playbook from starting any dependent activities. For example, a Create Record activity would prevent starting **After Previous** activities, which may be in other stages. Where possible, design your playbooks to run non-interactive activities before interactive activities that could block them.

Create Child Case activity

Enable agents and fulfillers to create a child case during a playbook run.

Roles and availability

These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions.

Input	Type	Description
		Note: **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Parent	Choice	Parent record of the child case.
Assignment Group	Choice	User group that the child case is assigned to.
Assigned To	Choice	User that the child case is assigned to.
Short Description	String	Summary of the child case.

Input	Type	Description
Description	String	Detailed information to display about the child case.
Priority	Choice	Prioritization level of the child case.
Collect data from user	True/False	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.

Outputs

Your child case record.

Create Child Task activity

Enable agents and fulfillers to create a child task during a playbook run.

Roles and availability

These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running:

Input	Type	Description
		• **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Parent	Choice	Parent record of the child task.
Assignment Group	Choice	User group that the child task is assigned to.
Assigned To	Choice	User that the child task is assigned to.
Short Description	String	Summary of the child task.
Description	String	Detailed information to display about the child task.
Priority	Choice	Prioritization level of the child task.
Collect data from user	True/False	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the

Input	Type	Description
		activity waits for user action or data before it completes.

Outputs

Your child task record.

Docusign activities

Enable agents and fulfillers to collect electronic signatures during a playbook run, via Workflow Data Fabric's Docusign spoke.

Roles and availability

- These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.
- You must have access to Workflow Data Fabric's Docusign spoke.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when

		your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Attachment	Choice	File or document to be signed.
Email subject	String	Subject of the email sent to signatories.
Email body	String	Message displayed to signatories.
Signatory	Choice	User record from the sys_user table that specifies who the signature request is sent to.
DocuSign account	String	Account record from the sn_docusign_spoke_accounts table that specifies where the signature request is sent from.

Outputs

Your signed document, available for review.

Guided Decision activity

Choose a decision tree from your Guided Decisions framework to step agents through how to proceed with a task.

Roles and availability

This activity is available with a subscription to App Engine or Customer Service Management (CSM). For more information on how to enable this activity for use in Playbooks, see Activate Playbooks for Customer Service Management (CSM).

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this

Input	Type	Description
		input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Decision Tree	Reference	The Decision Tree [ga_decision_tree] record whose decision inputs and guidance you want to show to agents in the Workspace playbook. For more information on how to set up the Guided Decisions framework, see Configuring Guided Decisions.
Task	Reference	The triggering Case [sn_customerservice_case] record.

Outputs

There are no outputs for this activity.

Invoke PaCE activity

Enable the Policy as Code Engine (PaCE) activity in Playbooks to develop playbook processes.

Roles and availability

These activities are available as part of the application that is automatically bundled with PaCE. Users with the admin, pd_admin, pd_author, or pd_content_author can add these activities to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Name	Type	Description
Label	String	Title to display as activity and playbook card.

Name	Type	Description
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity. You can use data from prior activities to build conditions.
When to start	Choice	Select a start rule for when your activity should start running: • **Immediately**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **Before previous**: Your activity starts before specified activities have finished running. • **After previous**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties.
Service	String	The calling service that is invoking the PaCE API.
Category	String	The category type.
Policy tag	String	Tag name of the policy.
Document Ids	Array.Object	Name of document IDs.
Data	JSON	The policy-specific data that includes all possible caller inputs.
Options	Object	Options of the policy.

Outputs

The outputs are configurable in Flow Designer and are set by the Automation plan.

Label	Type
Root execution id	Sys ID (GUID)
Don't Treat as Error	True/False

Label	Type
Response	JSON
Action Status	String

Microsoft Teams activities

Enable agents and fulfillers to send direct messages and post in Microsoft Teams channels during a playbook run, via Workflow Data Fabric's Microsoft Teams spoke.

Roles and availability

- These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.
- You must have access to Workflow Data Fabric's Microsoft Teams spoke.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Condition to run	Condition Builder	Criteria that must be met to run the activity. You can use the output data from prior activities to build the conditions to run this activity.
When to start	Choice	Option to specify when the activity runs. Options include • With Previous - the activity runs at the same time as the previous activity • After Previous - the activity only runs after the previous activity completes running • Immediately - the activity runs immediately without waiting for other activities
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties.

Input	Type	Description
		Note: **Show additional options** to see this field.
Chat type	Choice	Choose whether to post to a Microsoft Teams channel or direct message.
Message	String	Message text to display to end users.
Team ID	String	Sender of the Microsoft Teams message.
Channel name	String	Microsoft Teams channel that the message is posted in.
To members	String	Microsoft Teams user that the message is sent to.

Outputs

Your message posted in a Microsoft Teams channel or sent as a direct message.

Request Multi-Level Approval activity

Enable agents and fulfillers to submit an approval requests to first and second-level managers during a playbook run.

Roles and availability

These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.

Input	Type	Description
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Assigned to	Choice	Individual user who the first and second level manager approval requests are generated for.
Due date	Date/Time	Date the approvals are due by.
Comments	String	Option for the approvers to add comments in the approval requests.
Table	Choice	Table that you want to choose the associated record from.
Record	Choice	Associated record for the approval requests.

Outputs

The state of your approval requests.

Request Ad Hoc Approval activity

Enable agents and fulfillers to specify which user(s) should complete approval request(s) during a playbook run.

Roles and availability

These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts

Input	Type	Description
		running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Approver	Choice	Individual users that will receive approval requests.
Approver group	Choice	User Group that will receive approval requests.
Table	Choice	Table that you want to choose the associated record from.
Record	Choice	Associated record for the approval requests.

Outputs

- Records of your approval requests.
- The states of your approval requests.

Request Manager Approval activity

Enable agents and fulfillers to submit an approval request to a manager during a playbook run.

Roles and availability

These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.

Input	Type	Description
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Table name	Choice	Table that you want to choose the associated record from.

Input	Type	Description
Record	Choice	Associated record for the approval request.
Approver	Choice	Individual user that will receive approval requests.
Due date	Date/Time	Date the approval is due by.
Comments	String	Option for the approver to add comments in the approval request.

Outputs

Record of your approval request.

Send Email activity

Create an email from previously gathered or generated data. Use this activity to send an email. This activity requires the playbook author to define who the email should be sent to, the subject, and the body. This activity surfaces the pre-defined content for the email to the end user so that the end user can confirm before sending the email.

Roles and availability

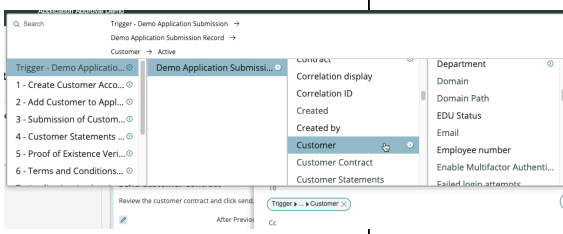
This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running:

Input	Type	Description
		• **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
To	List of Users	Recipient list for the email. There are 2 ways to configure this field: • Enter email addresses as strings if you want the emails to go to the same addresses every time. • When the email address value varies, use the pill picker to indicate where to find the recipient email address. For example, you would use the pill picker if you wanted to automatically notify credit card applicants of a rejection, because each

Input	Type	Description
		applicant has a different email. You would select the customer record from the trigger (when the application was submitted), because that is where the applicant's email address is found. 
Cc	List of Users	Cc recipient list for the email.
Subject	String	Subject of the email.
Body	HTML	Body of the email.
Wait for user input	Choice	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.
Tagline	String	Activity tagline to display to end users.
Record fields	Reference	Any additional record fields you want in the email. **Note:** This field cannot be edited during a playbook run.
Footer	String	Footer content for your email. You can enter a footer as a string, or use the pill picker to reference data (e.g. a timestamp) that you want to display at the bottom of the email.

Input	Type	Description
		Note: This field cannot be edited during a playbook run.
Form View	String	Form view to use for sending an email. If you don't provide a form view, the system uses the default view. Use the **new tab** button  to open and edit a form view with Form Builder in a new Workflow Studio tab. **Note:** Many form views are not supported in Workspace.
Form fields	Reference	Any additional fields you want in the playbook activity card. **Note:** This field cannot be edited during a playbook run.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assigned To	Reference.User [sys_user]	User responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Assignment Group	Reference.Group [sys_user_group]	Group responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.

Input	Type	Description
To Email Address	String	Comma-separated list of email addresses.
Cc Email Address	String	Comma-separated list of copied email addresses.
Bcc	List of Users	Bcc blind copy recipient list for the email.
Bcc Email Address	String	Comma-separated list of blind copied email addresses.
Target Record	Reference.Task [task]	Reference to the record that the system attaches the email to.
Table	Table Name	Table containing activity or playbook data.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Automated	True/False	Whether the flow automatically completes or requires user input. If set to **True**, the activity card in Playbook Experience displays a **Automated Task** tagline.
Record	Reference.Task[task]	Reference to record created.
Email	Reference.Notification[sysevent_email_action]	Reference to the newly created email notification record

Show Knowledge Article activity

Display a knowledge article to end users.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.

Input	Type	Description
Title	String	Title displayed to end users.
Knowledge Article	Reference.Knowledge [kb_knowledge]	Knowledge article displayed to end users.
Wait for user input	Choice	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assignment Group	Reference.Group [sys_user_group]	Group responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Assigned To	Reference.User [sys_user]	User responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Record	Reference.Task[task]	Reference to the record associated with the activity.

Show List of Records activity

Display a list records that match a set of conditions.

Roles and availability

- This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered.

Input	Type	Description
		• **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Table	Table Name	Table containing records you want to display to end users.
Fields to show	String	Comma-separated list of fields to display as columns of a list.
Wait for user input	Choice	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assignment Group	Reference.Group [sys_user_group]	Group responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Assigned To	Reference.User [sys_user]	User responsible for completing the task associated with the activity.

Input	Type	Description
		Note: **Show additional options** to see this field.
Conditions	Condition Builder	Criteria that you want your list of records to meet.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Record	Reference.Task [task]	Reference to the record associated with the activity.
Fields	String	Comma-separated list of system field names to display as columns in the list.

Slack activities

Enable agents and fulfillers to send direct messages and post in Slack channels during a playbook run, via Workflow Data Fabric's Slack spoke.

Roles and availability

- These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.
- You must have access to Workflow Data Fabric's Slack spoke.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions.

Input	Type	Description
		Note: **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Channel/Member ID	String	Slack channel or member that the message is sent to.
Message	String	Message text to display to end users.
Blocks	String	Code blocks to include in the message to end users.
Username	String	Sender of the Slack message.

Input	Type	Description
Icon	String	Slack user profile icon of the sender.
Collect data from user	True/False	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.

Outputs

Your message posted in a Slack channel or sent as a direct message.

Two Step Instruction activity

Display a different message to end users based on the current activity state. You can specify an initial state message, a skipped state message, and a completed state message.

Roles and availability

- This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running:

Input	Type	Description
		• **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Initial Message	String	Message text to display to end users when the activity state is In Progress.
Completed Message	String	Message text to display to end users when the activity state is Completed.
Skipped Message	String	Message text to display to end users when the activity state is Skipped.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assignment Group	Reference.Group [sys_user_group]	Group responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Assigned To	Reference.User [sys_user]	User responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Record	Reference.Task[task]	Reference to the record associated with the activity.

Update Record activity

Update a record with the field values you specify.

Roles and availability

- This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage.

Input	Type	Description
		You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Table	Table Name	Table containing the record to update.
Record	Reference	Reference to the record you want to update.
Fields	Template Value	Field values to change during record update.

Input	Type	Description
Wait for user input	Choice	Option to pause the playbook until the end user manually completes or skips the activity. Only users with the playbook.admin role can edit this field. By default, the activity waits for user action or data before it completes.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Assignment Group	Reference.Group [sys_user_group]	Group responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Assigned To	Reference.User [sys_user]	User responsible for completing the task associated with the activity. **Note:** **Show additional options** to see this field.
Fields to show after update	String	Comma-separated list of fields to display for an updated record.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Table	Table Name	Table containing updated record.
Record	Document ID	Reference to record updated.

Design considerations

Create form views for activities that you want to render in a playbook

Use a form view to display only the fields your users need to update a record. Your view should display required fields or those fields validated by other business logic. See View Management.

Run non-interactive activities before interactive activities

If an Update Record activity interactively gathers data from users, it prevents the playbook from starting any dependent activities. For example, an Update Record activity would prevent starting **After Previous** activities, which may be in other stages. Where possible, design your playbooks to run non-interactive activities before interactive activities that could block them.

View Approval Requests activity

Display a list of approval requests from within Playbook Experience.

Roles and availability

These activities are available as an application in the ServiceNow Store. Users with the playbook.admin or pd_author role can add these activities to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when

Input	Type	Description
		your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Table	Choice	Table that you want to choose the approval requests from.
Conditions	Condition Builder	Criteria that you want your list of approval requests to meet.

Advanced Inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Order by	String	Field you want to use to sort a list of records.
Sort Type	Choice	Option to sort records alphabetically or reverse alphabetically.
Max Results	Integer	Maximum number of results to display to end users.

Wait For Condition activity

Pause the playbook until a record has field values that match a set of conditions.

Roles and availability

- This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered.

Input	Type	Description
		• **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Record	Reference	Reference to the record that pauses the playbook until conditions match.
Table	Table Name	Table containing the record to update.
Conditions	Template Value	Criteria that a record must meet for the playbook to continue.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Enable timeout	Choice	Option to time out the Wait for Condition activity if the conditions aren't met after a specified Duration.
Duration	Date/Time	Amount of time to wait before the activity times out and its state is set to Skipped. This input requires setting the Enable timeout input.
Schedule	Choice	Schedule used to compute duration values. This input requires setting the Enable timeout input.

Outputs

The Wait for Condition activity has no outputs.

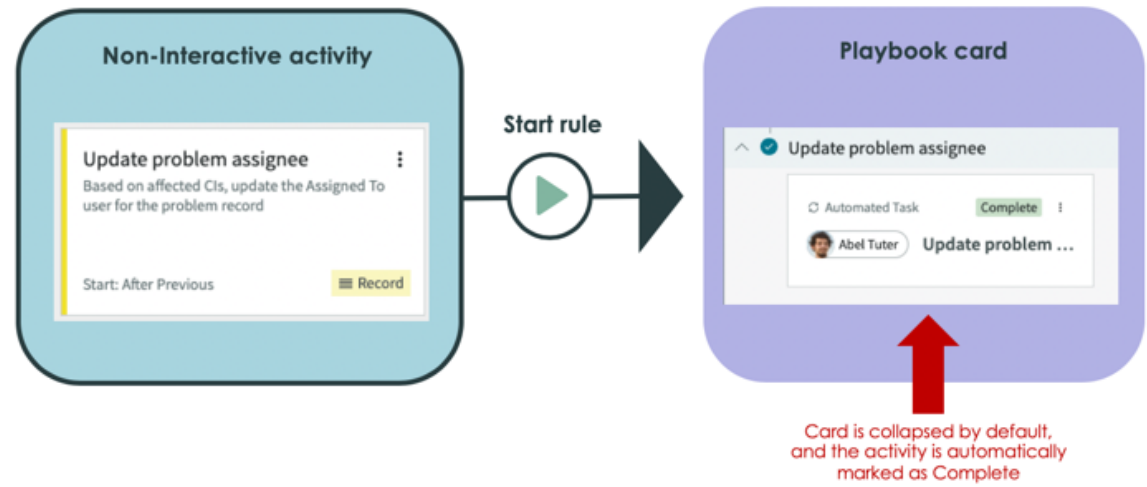
Non-Interactive activities

A non-interactive activity runs entirely behind-the-scenes on the ServiceNow AI Platform® and doesn't require any user input.

A non-interactive activity is an entirely automated operation on the ServiceNow AI Platform which doesn't require any user input to proceed to completion. Non-interactive activities still render in a playbook, but only display information to agents. Configure the activity inputs for an interactive activity so that the activity is a fully automated operation on the ServiceNow AI Platform.

When non-interactive activities run, they automatically proceed to completion or are skipped. For example, if your activity automatically updates the Assigned To user for a record, the Playbook card can display the newly updated Assigned To user's name to the playbook agent, but the card's status is automatically set to Complete.

Non-interactive activities



Non-interactive activities turn into activity cards that are collapsed and marked complete in a playbook. The playbook continues running without any input from the playbook agent.

To learn how to design a playbook with non-interactive activities, see design an automated process.

Automated Create Record activity

Create a record without pausing the playbook to ask for user input. When the activity runs, it immediately creates the record and continues to the next activity in the playbook. The record must meet server-side validation rules such as data policies, business rules and dictionary-defined mandatory fields but ignores UI policies.

Roles and availability

- This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.

Input	Type	Description
Table Name	Table Name	Table in which you want to create a new record.
Fields	Template Value	Field values to set during record creation.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Table Name	Table Name	Table containing new record.
Record	Reference.Task [task]	Reference to record created.

Automated Send Email activity

Create an email from previously gathered or generated data without pausing the playbook to ask for user input. When the activity runs, it immediately sends the email and continues to the next activity in the playbook.

Roles and availability

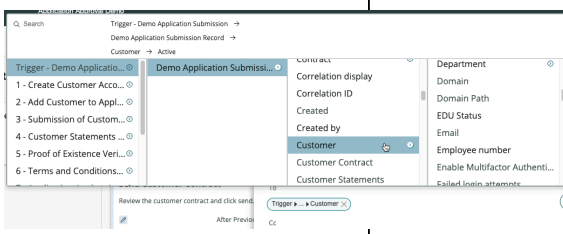
This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running:

Input	Type	Description
		• **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
To	List of Users	Recipient list for the email. There are 2 ways to configure this field: • Enter email addresses as strings if you want the emails to go to the same addresses every time. • When the email address value varies, use the pill picker to indicate where to find the recipient email address. For example, you would use the pill picker if you wanted to automatically notify credit card applicants of a rejection, because each

Input	Type	Description
		applicant has a different email. You would select the customer record from the trigger (when the application was submitted), because that is where the applicant's email address is found. 
Cc	List of Users	Cc recipient list for the email.
Subject	String	Subject of the email.
Body	HTML	Body of the email.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
To Email Address	String	Comma-separated list of email addresses.
Cc Email Address	String	Comma-separated list of copied email addresses.
Bcc	List of Users	Bcc blind copy recipient list for the email.
Bcc Email Address	String	Comma-separated list of blind copied email addresses.
Target Record	Reference.Task [task]	Reference to the record that the system attaches the email to.
Table	Table Name	Table containing activity or playbook data.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Record	Reference.Task[task]	Reference to record created.
Email	Reference.Notification[sysevent_email_action]	Reference to the newly created email notification record

Automated Update Record activity

Update a record without pausing the playbook to ask for user input. When the activity runs, it immediately updates the record and continues to the next activity in the playbook. The record must meet server-side validation rules such as data policies, business rules and dictionary-defined mandatory fields but ignores UI policies.

Roles and availability

- This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running.

Input	Type	Description
		Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Record	Reference	Reference to the record you want to update.
Table	Table Name	Table containing the record to update.
Fields	Template Value	Field values to change during record update.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Table	Table Name	Table containing updated record.
Record	Reference.Task[task]	Reference to record updated.

Autocompleting User Form

Display a form during runtime to collect input values for your playbook.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Completion Condition	Condition Builder	Criteria that must be met to complete the activity and save the record.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Look Up Records activity

Find system records that match a set of conditions.

Roles and availability

- This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Input	Type	Description
Label	String	Title to display as activity and playbook card.
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions.

Input	Type	Description
		Note: **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Table	Table Name	Table whose records you want to look up.
Conditions	Condition Builder	Criteria that you want your list of records to meet.

Advanced inputs

After configuring the required inputs for your activity, you can also choose to configure additional inputs. In the side panel, select **Show additional options** to display these advanced inputs. For more information, see Activity experience.

Input	Type	Description
Order by	String	Field you want to use to sort a list of records.
Sort Type	Choice	Option to sort records alphabetically or reverse alphabetically.
Max Results	Integer	Maximum number of results to display to end users.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Records	Records	References to the records that meet your specified conditions
Count	Integer	Number of records found

Make a Decision-First Match activity

Execute rules in a decision table. When this activity runs, it will immediately finish and continue the process execution. The activity returns the result of the first matching decision rule, based on the rank of the rules. To return results from all matching rules, add the decision table to a subflow.

Roles and availability

This activity is available as a common activity. Users with the admin, playbook.admin, or pd_author can add this activity to a playbook.

Inputs

Open the activity properties panel and configure your activity to add values for the following inputs. If the input value varies, use the pill-picker to show where to get the value. To learn more about the pill-picker, see Dot-walking examples.

Note:
You must select the decision table to run. Any additional inputs and outputs are dynamic based on the decision table.

Input	Type	Description
Label	String	Title to display as activity and playbook card.

Input	Type	Description
Description	String	Information to display about activity usage or outcome.
Run condition	Condition Builder	Conditions that must be met to run an activity or stage. You can use data from prior activities to build conditions. **Note:** **Show additional options** to see this field.
Start Rule	Choice	Under **Schedule** > **Start Rule**, select a start rule for when your stage should start running: • **When process starts**: Your stage starts running as soon as the playbook starts. • **After specific stages**: Your stage starts running after specified stage(s) have finished running. Under Schedule > Start Rule, select a start rule for when your activity should start running: • **When stage starts**: Your activity starts running as soon as its stage starts running. Your stage starts running when your playbook is triggered. • **After specific activities**: Your activity starts running after specified activities have finished running.
Start with delay	True/False	Option to wait for a duration of time before running an activity or stage. When enabled, this input displays the Start with delay input properties. **Note:** **Show additional options** to see this field.
Table Name	Table Name	Table in which you want to create a new record.

Input	Type	Description
Fields	Template Value	Field values to set during record creation.

Outputs

These outputs can provide data to other activities in your playbook. You can access this data as activity inputs when you configure your activity:

Output	Type	Description
Table Name	Table Name	Table containing new record.
Record	Reference.Task [task]	Reference to record created.

Flows, subflows, and actions reference

Get details about Workflow Studio flow components like actions, flow logic, and properties.

Available references by component type

Workflow Studio component	Available references
Flows	• Types of flows and when to use them • Workflow Studio flow trigger types • Workflow Studio actions • Workflow Studio flow logic • User preferences for flows
Subflows	• Types of flows and when to use them • Workflow Studio actions • Workflow Studio flow logic • User preferences for flows
Actions	• Workflow Studio input and output data variables • Workflow Studio steps • Supported Service Catalog variable types
All	• General guidelines for Workflow Studio flows, subflows, and actions • Transform functions • Workflow Studio flow integrations • Workflow Studio flow system properties

General guidelines for Workflow Studio flows, subflows, and actions

Create, run, troubleshoot, and monitor your Workflow Studio components more effectively. Use these guidelines to optimize the performance of your Workflow Studio components.

Overview of Workflow Studio

Integrate workflow authoring, configuring, and monitoring into a single page experience. Consolidate playbooks, flows, actions, decision tables, and integrations into one design environment.

Application development

When designing an action or a flow, use these general guidelines.

Use standard ServiceNow AI Platform application development capabilities to create, manage, protect, and deploy Workflow Studio content. Flow and action designers typically perform the following application development tasks:

- Create a custom application to store flows and actions.
- Set application permissions to share or restrict access to application data.
- Grant application developers access to Workflow Studio.
- Publish custom applications to the application repository to deploy flows and actions on other instances.

Flows

Flows should be short, modular, reusable collections of work. If they take more than an hour to execute, they're probably too long and can be more efficient.

Any general guidelines that apply to flows also apply to subflows.

Prevent conflicting or duplicate business logic

Automations can be created with Flow Designer, business rules, workflows, and Integration Hub. Before you start using Workflow Studio , make sure you understand how existing ServiceNow AI Platform automations work. Deactivate automations before replacing them with Workflow Studio flows and actions. See the Architecture Overview to learn how Workflow Studio works within the ServiceNow AI Platform.

Review Flows, Sub-flows, and Actions documentation, if necessary.

Determine whether your flow needs a trigger or variable input

Flows always run when their trigger conditions are met, and triggers always provide the same data as input for flows. If you need variable input to initiate a flow instead, create a subflow.

Reuse business logic

Create a set of reusable operations as a subflow that can then be used in multiple flows.

Grant flow roles to access role-protected data and preserve user information

Flow roles help keep permissions for your flows simple. Use flow roles to preserve user information and grant access to data, instead of running a flow as the system user. Adding flow roles also gives access to additional data that a user-initiated flow does not usually have. The roles granted only apply to the flow. They do not apply to the user who initiated the flow.

Use flow logic or a schedule-based trigger to control flow timing

Flow logic or schedule-based triggers help to optimize the performance of your flows. Do not use the `gs.sleep()` method to wait within a flow. The `gs.sleep()` method prevents the thread from performing other work. To run a flow at a specific time, use a schedule-based trigger. To pause a flow for a specific duration, use the Wait for a duration or wait for condition flow logic.

Avoid dependencies

Parallel branches that depend on each other stall a flow when a branch has to wait for output from another branch. Instead of creating parallel branches within a flow, call a subflow and return the results to the main flow.

Scope loop counters

Script loops don't have a maximum number of iterations, so loops execute infinitely when there is not a valid exit condition.

To make sure that there is a valid exit condition, scope loop counters in inline scripts or in script steps within an action. Add `var` to `for (i=0; i< length; i++)`:`for (var i=0; i< length; i++)`

Limit For Each and Do Until loops to 1000 iterations

Iterations with 1000 or more loops can lead to memory issues from having to store execution details and context records.

- Set max records on Look Up Records.
- Avoid changing property sn_flow_designer.max_iterations, which defaults to 1000.
- For nested loops, each loop has its own maximum number of iterations.
- For large amounts of data processing, consider batching into smaller batches.
- For bulk imports, consider concurrent imports.

Use QuickAPI for faster executions (business rule alternative)

- QuickAPI executions are much faster, but there is less debugging capability.
- Foreground QuickAPI executions run in the user session as the user who called the flow.
- Background QuickAPI executions run in a background thread and are run in the 'system' user session.

Use Do Until loops instead of calling flows from themselves

Direct recursion where a flow calls itself is not allowed and errors out. Indirect recursion where flow A calls flow B, which calls flow A is allowed up to three times. Instead of calling a flow recursively, use the Do Until flow logic to continue working on records until a certain condition is met.

Execute flows in the background

Executing flows in the background allows UI threads to be released rather than pausing the user session until the flow execution completes. By default, flows run asynchronously in the background. Running flows in the background allows users to continue working in the UI while the flow runs.

Avoid flow logic that waits after collecting a large output

Using a large payload immediately after it is retrieved can help prevent memory issues. Rather than store a large payload in memory, add actions to process the payload. The sooner you process a retrieved payload, the sooner the system can free up memory to process other actions.

Minimize switching between environments

Constantly switching back and forth between instance and MID Server steps in a flow can lead to delays in processing. To minimize the risk of delays, limit switching between instance and MID to only one time.

Include sys_complex_object records generated by the flow in update sets

Missing complex data schemas can cause execution issues. Make sure you include sys_complex_object records generated by the flow in update sets. Rather than manually build update sets, consider transferring flows from one instance to another by using the application repository.

Call flows from a script when you need a custom trigger

If none of the existing triggers meet your business needs, you can create a script to start a flow when its custom trigger conditions are met. Rather than creating a flow with an unneeded trigger, consider instead creating a subflow, which does not have a trigger. Use your script to provide the necessary subflow inputs only when your script conditions are met. Calling a subflow rather than a flow avoids the possibility of the flow trigger conditions being met and running the flow unexpectedly.

Avoid deploying newer release flows to instances on older releases

Workflow Studio does not support deploying newer release flows to instances running on earlier releases.

> **Danger:**
> The flow data model can change between releases, which can prevent newer flows from running or produce unexpected results when running on earlier release instances. Upgrade your instances to be on the same release versions before deploying them.

Turn flow reporting off in production

Minimize the amount of memory required to run flows by disabling Flow reporting. Flow reporting stores configuration and runtime information for the Execution Details page. These reports are good for troubleshooting, but requires a large amount of data to be retained both in memory and in the database. By default, flow reporting is disabled, and the system only generates execution details when you manually test a flow or action. Instead you can use log files, which are still available when reporting is turned off.

Reduce the amount of memory consumed in flows with nested looping

When reporting is activated, set com.snc.process_flow.reporting.iteration.lastn to a value of "1" to reduce the amounts of the amounts of memory that previous loop iterations consume. The more iterations you report on, the more memory is required.

Subflows

General guidelines that apply to flows also apply to subflows.

Reasons to use a subflow instead of a flow include the following:

Reuse business logic

Create a set of reusable operations as a subflow that can then be used in multiple flows.

Reuse business logic

Create a set of reusable operations as a subflow that can then be used in multiple flows.

Configure different input values for each call

Configure a subflow's input values differently each time you call it. For example, design a subflow to accept different record types as an input run. Reuse this generic record subflow instead of writing a specific flow for each record type.

Improve performance and readability of large flows

Use subflows when a flow exceeds 25 actions. 50 is the maximum number of actions specified by the sn_flow_designer.max_actions system property, but limit a flow to 25 actions for the best performance.

Pass inputs and outputs with subflows

Call subflows if you want to pass inputs and outputs. Use subflows if you want to specify the inputs available to a subflow when it starts, or if you want to specify the outputs available to the parent flow after a subflow ends.

Trigger multiple flows on a single event vs using parallel subflows

- Use parallel subflows if there are interrelated outputs or some action must be taken when all are available. If not, then it's simpler to trigger multiple flows.
- To configure parallel subflows, launch each subflow without a wait and then use wait for condition to wait for each subflow to be terminal (complete, error, canceled)

Use dynamic flows if you have multiple subflows with similar functionality

Dynamic flows let you compartmentalize your processes by applying a template to handle the inputs of multiple similar subflows. Compartmentalization lets you distinguish between subflows that perform similar functions, such as subflows for IntegrationHub spokes.

Avoid the 10-item limit in the error-handling-process

Rather than force your error-handling-process to fit within a 10-item limit, call subflows, which can contain many more items. You can also use the subflow outputs to trigger automation in other flows.

Take corrective actions

Rather than recreate the same sequence of actions in multiple flows, create reusable subflows to correct errors to your record data. When a flow error leaves your record data in an undesired state, use subflows to correct these records. You can use the error handler to identify such record data as a subflow output.

Triggers

Follow these general guidelines when creating record triggers.

Determine whether your flow needs a trigger or variable input

Flows always run when their trigger conditions are met. Triggers always provide the same data as input for flows. If you need variable input to initiate a flow instead, create a subflow.

Add conditions to specify what record values start your flow

Starting a flow only when needed consumes fewer system resources than starting a flow, pausing it, and waiting to resume the flow until a specific record condition applies. Instead of creating a flow that starts with a Wait for condition action, redesign the flow to include the wait condition as part of the record trigger.

Create unique conditions for record triggers on the same table

To prevent flows from overwriting each other, create unique conditions for each flow running on the same table. If multiple flows on the same table the same filter, there

is no way to know the order in which the flows run. Using conditions also helps to optimize flow performance by returning a more precise, smaller set of records.

Ignore records added or updated by import and update sets

Record triggers ignore records added or updated by applying an update set or importing an XML file. These operations apply to the entire application or table rather than an individual record.

Replace record triggers on Service Catalog tables with Service Catalog application triggers

Flow Designer no longer displays Service Catalog tables as options for record triggers. Instead, create flows that use the Service Catalog application trigger type.

Wait conditions

Follow these general guidelines when creating flows that wait for a condition.

Use record triggers instead of wait conditions to start flows

If you only want a flow to run when certain record conditions are met, create a flow with a record trigger instead of starting and pausing a flow. A waiting flow consumes more system resources than a flow trigger.

Cancel flows whose resume conditions can never occur

Prevent your flows from waiting indefinitely by specifying flow stop conditions with End Flow flow logic. To free up system resources, you can also cancel any flow whose resume conditions can never be met. For example, cancel flows waiting for incident record updates where the related incident is closed.

Restrict wait conditions to fields present on the current table

The Wait For Condition action can only monitor changes to the fields of the table to which the record belongs. The action cannot detect changes to fields in related records or catalog variables. For example, if an action waits for changes to an Incident record, then it cannot detect changes to a related record such as a catalog item or change task record. Avoid building wait conditions that dot-walk to another record as these fields actually belong to the related record. Avoid building wait conditions that rely on catalog variables.

Flows or subflows with stages

Follow these general guidelines when creating flows or subflows with stages.

Avoid defining stages that depend on a For Each flow logic

Flow Designer prevents you from adding stages within a **For Each** block. You can only add stages before or after a **For Each** block.

Avoid creating stages for the same records in different flows or subflows

A stage field always displays the stage information provided by the last flow or subflow to run on a table's record. If multiple flows or subflows run on the same records, then the stages defined in one flow or subflow can in theory overwrite the stages from another flow or subflow. To avoid multiple flows or subflows overwriting each other's stages, define unique trigger or start conditions for each flow or subflow.

Avoid updating stage fields from outside a flow or subflow

If you manage stages with a flow or subflow, avoid directly updating record stage fields from outside the flow or subflow. Manually updating the value of a stage field may produce unexpected or undesired results.

Ensure that each flow on a table has unique trigger conditions

Adding unique trigger conditions to each flow ensures the flows only run under those conditions and prevents the stages from one flow overwriting the stages of another flow. Specifying unique trigger conditions makes it easier to troubleshoot flows by limiting the number of flow executions that can produce record changes.

Use error stages to communicate with the user

The flow error state does not affect flow execution. A flow continues running even if it reaches an error stage. Use a conditional flow logic block to set the error stage and communicate to the user that the state of the current stage is Error. For example, if an approval is not approved within the required limit, you may want to communicate an error to the user.

Use the error stage to stop processing a flow

Use a conditional flow logic block to identify when a flow enters the error stage. Use the flow logic to stop processing the flow or take some kind of remediation action. For example, you may want to change the record state or assignment when a flow reaches an error state.

Do the following in parallel flow logic

Avoid creating data dependencies between paths

Since a flow can run paths in any order, avoid creating data dependencies between separate paths. For example, do not have one path that creates a record and another path that updates the same record. The update record path may run before the create record path.

Do not share data between paths

Workflow Studio prevents you from dragging data pills between paths because the system cannot determine which path will finish first to supply the output value.

Dynamic flows flow logic

Use dynamic flows if you have multiple subflows with similar functionality

Dynamic flows let you compartmentalize your processes by applying a template to handle the inputs of multiple similar subflows. Compartmentalization lets you distinguish between subflows that perform similar functions, such as subflows for IntegrationHub spokes.

Ensure dynamically called subflow inputs match template flow inputs

The system throws an error and the main flow can't run properly when the inputs of a dynamic flow and flow template don't match.

Use the correct context when getting flow outputs

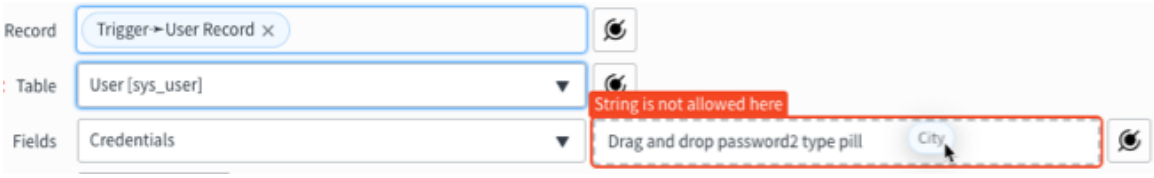
A context record uniquely identifies the flow run. If you run a dynamic flow multiple times, there are multiple context records to choose from. When you use dynamic flow multiple times within a flow, make sure to pick the right context record from the right run each time you get flow outputs.

Password2 data pills

Follow these general guidelines when designing flows containing Password (2 Way Encrypted) data.

Assign values using existing Password (2 Way Encrypted) data pills.

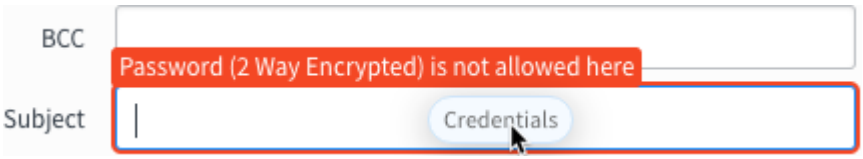
You can only assign a value to a password2 variable by selecting an existing password2 data pill. Selecting values from other field types is not supported. Workflow Studio presents a warning message when invalid data pill types are selected.



Note:
You cannot manually enter Password (2 Way Encrypted) values.

Use Password (2 Way Encrypted) variables only for valid field types

Workflow Studio prevents selecting Password2 data pills as the value for invalid field types. The system presents a warning message when the field is an incompatible type.



Workflow Studio only allows Password2 data pills to be dragged into the following field types:

- Email body fields
- HTML fields
- Password 2 Fields
- PowerShell Input Variables
- REST fields
  - Variables
  - REST payload body
  - Query parameters
  - Headers
  - REST multi-part form values
  - Form URL-encoded values
- SOAP fields
  - Headers
  - Envelope

Note:
you cannot use Password (2 Way Encrypted) variables as conditions

Flow Designer performs a validation check when a user saves, publishes, or tests actions and flows. This check shows that an alert for any data pills dropped in restricted field types and prevents the action or flow from executing. Update the action or flow to remove the invalid data pill and then retry the action.

Set up encryption modules for decryption

Only users with a valid encryption module access can decrypt and view the contents of password2 variables. To specify the encryption algorithm and which roles can access encrypted data, see Password2 encryption with KMF .

SLA Percentage Timer actions

Follow these general guidelines when creating flows that contain Service Level Agreement (SLA) Percentage Timer actions.

Add SLA Percentage Timer actions only to flows with an SLA Task trigger

An SLA Percentage Timer action can only run when the flow starts from an SLA Task trigger. You cannot activate a subflow containing an SLA Percentage Timer action.

Create conditional flow logic for expected Status values

Use the value of the **Status** field as a condition for flow logic. Build flow logic for expected **Status** values such as **Completed**, **Repair**, and **Skipped**. For example, add an **If** flow logic block to send a notification when the SLA Percentage Timer has a status of **Completed**.

Assign each SLA Percentage Timer action a unique cumulative Wait for percentage value

Each SLA Percentage Timer action computes its own Scheduled End Date/Time using its Wait for percentage value. If you create multiple SLA Percentage Timer actions, give each action its own unique cumulative Wait for percentage value. For example, create three separate actions with different percentage complete values such as 25%, 50%, and 75% complete. Setting all three actions to the same percentage complete value such as 25% causes the timers to complete at the same time.

Copy existing flows to make customizations

Reduce development time by copying the default SLA flows and customizing the copies with your own logic. Select a customized flow to run from the SLA definition. See Create an SLA definition .

Dynamic inputs

Consider dynamic inputs for third-party integrations

Dynamic inputs let you create flows that fetch data dynamically from external sources. In third-party integrations, dynamic inputs can provide data values that pertain to a particular endpoint. For more information on setting up third-party integrations with Workflow Studio, see IntegrationHub .

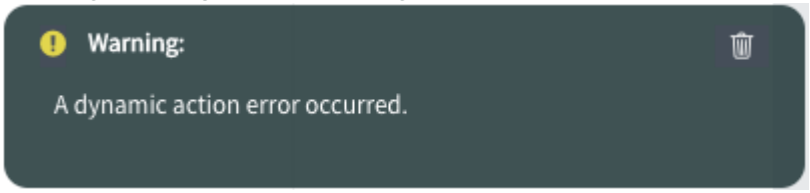
Be aware of the time required to retrieve large amounts of data

By default, dynamic inputs have up to 300 seconds to gather data before they time out. If your data gathering action needs more time to gather data, set the *sn_flow_designer.sync_action_execution_timeout_in_seconds* system property to a higher value. However, don't use long timeout values for interactive flows where an end user must enter or select a value.

Be aware of scripting errors

Because all data gathering actions use a script step, potential errors could occur from scripting. When using scripts to output JSON variables for your dynamic inputs, you may encounter errors that prevent inputs from receiving the JSON values they need. When a dynamic input scripting error occurs, the following warning message may appear.

Message displayed for scripting error



Limit dynamic inputs type inputs to 40 input values

A dynamic inputs type input can only render a certain number of inputs before the JSON object becomes to too big to store in memory. Limiting your dynamic inputs to 40 input values minimizes the chances that you will run out of memory and experience unexpected behaviors such as rendering errors or data truncation.

Limit JSON output to 5000 array items for dynamic templates and dynamic choices

Dynamic choice and dynamic template inputs can only display up to 5000 array items. A dynamic choice can only display up to 5000 choice list options, and a dynamic template can only display up to 5000 field template values. If your data gathering action collects data for a dynamic template or a dynamic choice, restrict the maximum number of array items it returns to 5000. The 5000 array items limit prevents the instance from having performance issues when rendering the choices or field values.

Dynamic outputs

Use dynamic outputs for third-party integrations

Use dynamic outputs to introspect and fetch data from external systems during the flow design. For example, you can specify service endpoints or call actions that interact with specific endpoint APIs. For more information on setting up third-party integrations with Workflow Studio, see IntegrationHub.

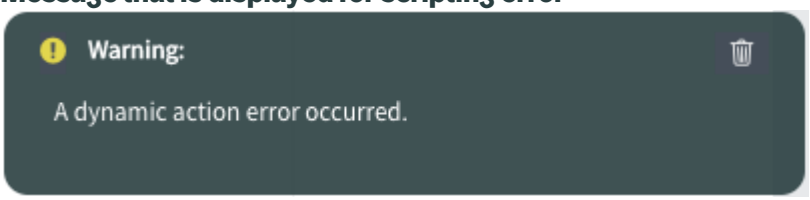
Note the time that is required to retrieve large amounts of data

By default, dynamic outputs have up to 300 seconds to gather data before the system stops them. If your data gathering action needs more time to gather data, set the *sn_flow_designer.sync_action_execution_timeout_in_seconds* system property to a greater value. Avoid long timeout values for interactive flows where an end user is expecting to enter or select a value.

Be aware of scripting errors

Because all data gathering actions use a script step, potential errors could occur from scripting. Review any scripts that are used to output JSON variables because script errors may prevent the outputs from receiving the JSON values that they need. When a dynamic output scripting error occurs, the following warning message may appear.

Message that is displayed for scripting error



List.[Table] data

Add a reference qualifier to filter list records

Filter the records that the list variable displays as valid options by adding a reference qualifier. The reference qualifier acts as a required list filter and causes the list variable to display only records that match the reference qualifier conditions. For example, to only displays active incident records add the reference qualifier condition **[Active][is][true]**.

Avoid selecting default records for actions intended for ServiceNow Store

Avoid selecting default records for a list unless you know that all instances have access to the selected records. Spoke developers typically do not have access to the data of the customers who install their custom action. If you want to publish a custom action on the ServiceNow Store, you may need to provide default records as demo data.

Use List variables in For Each flow logic

You can use a List variable to specify the records to process within For Each flow logic. The For Each flow logic ignores any non-record sys_id present in the data. For example, if the List variable contains an email address, the flow logic ignores it.

Approval rules

Provide a default value

Create or select an approval rule as a default value.

Transform functions

Apply transform functions to valid types of data pills for the input

Be sure to check the type of data pill for the input before applying a transform function. Applying a transform function to an invalid data pill type results in the system skipping the transform. An error also occurs if transform functions produce results that the system cannot parse. For example, when transforming a string into a date, the system throws an error if the transform does not produce a valid date.

Confirm applied transform functions for multiple inputs with the same data pill

A transform function creates a new value at runtime for a specific input, and does not change the original data pill. If you use the same data pill across multiple actions or steps, transform functions must therefore be applied to each individual input.

View final transformed values in the flow execution details

Only the final transformed value, and not the value for each applied transform, appears in the flow execution details.

Test transform functions to verify they produce expected results

Make sure that your transform functions produce the expected runtime values for the data pills. For more information, see Test a flow and Test an action.

Inline scripts

Follow these general guidelines to create reusable and maintainable inline scripts.

Write inline script for small non-reusable logic

Use inline script format or modify the data for specific inputs and use cases. For reusable logic, create an action or subflow instead.

Review available transform functions

Workflow Studio provides a list of standard transform functions for data conversions and formatting operations. Rather than write and maintain a custom script solution, select an existing transform function if one is available.

Call script includes from inline script

Call a script include from your inline script to reduce the amount of code you write and also to maintain common code in a single location. Use the class constructor to call your script include. For more information about creating a script include, see Script includes.

```
var si = new MyScriptInclude();
si.functionOne();
```

Create custom actions or subflows for reusable code rather than inline script

Create custom actions or subflows for reusable or complex data logic such as changing the data type of source data. You may also want to provide custom actions or subflows for flow designers who are not comfortable with code.

Avoid duplicating action and flow functionality

Avoid writing inline script that duplicates action and flow functionality. For example, rather than write inline script to perform record operations, use the create and update record baseline actions.

Avoid data type changes

Avoid runtime errors by verifying that your inline script provides information in the same data type as the input or output expects.

Create variables by declaring them with the var keyword

Use the `var` keyword to declare variables so that they remain within the proper JavaScript scope. When you create a variable by assigning it a value, JavaScript may attach it to the global object, which can result in variable values persisting outside of the local scope and causing errors.

Process records outputs with For Each flow logic and the flow data object

Inline script can only access the **records** output of a Look Up Records action from For Each flow logic. Add a Look Up Records action to the flow to generate the records output. Add a For Each flow logic to the flow to process each record in the records output. Create an inline script reference to the For Each flow logic using the fd_data and item objects. For example, this reference assumes that the For Each flow logic is the second item in your flow outline `fd_data._2__for_each.item`.

Use type ahead suggestions to generate references to flow and action data.

Create references to flow and action data using the fd_data object. The script editor displays type ahead suggestions for existing flow and action data when you type `fd_data`. Select a suggestion to build references to flow and action data.

Note:
Refer to record data in a For Each flow logic using the **item** object.

Scope loop counters

Script loops don't have a maximum number of iterations, so loops execute infinitely when there is not a valid exit condition.

To make sure that there is a valid exit condition, use scope loop counters in inline scripts or in script steps within an action. Add `var` to`for (i=0; i< length; i++)` and get `for (var i=0; i< length; i++)`

Complex data

Follow these general guidelines to create reusable and maintainable data structures.

Minimize the number of child levels in the hierarchy

The more child levels a data structure has, the more difficult it is to view and select a data variable from the hierarchy. While you can build data structures with any number of child levels, it becomes difficult to navigate and understand data structures with more than seven child levels. For the best user experience,

avoid creating data structures that have so many child levels that you must scroll horizontally to see and populate them.

Create a separate object for each type of record data

Most Workflow Studio data is record data whether it is from an instance or an external system. This design method ensures that you know what the object contains and where the data came from.

Recreate record data structures

When building objects that receive or transmit record data, review the database dictionary entries for these records and create matching object data structures. For example, suppose that you want an object to contain data from Incident and Configuration Item tables. You might create a string element for the **Short description** field in the Incident table, and an array of strings element for the **Class** field in the Configuration Item table.

Create objects to combine different types of records

If you need information from multiple types of records, create an object that contains all the information you need. You can then use the object to format or parse data in Workflow Studio.

Scripting with complex data

Keep these general guidelines in mind when scripting with complex data.

Use string inputs to convert complex data into a JSON string

When you map complex data to a string input, Workflow Studio automatically converts it into a JSON string. Instead of writing a script, you can add a string input to a REST step and map it to complex data from a prior action or step.

Save your objects as templates

Save your objects as templates so you can reuse them in other actions, flows, and Script steps.

Create script input variables to access prior data

Create a script input variable for any data you want to access from the action input or a prior step. Map the script input variable to the input or step data pill. For example, map the script input variable to a list of user records you looked up in a prior step.

Create a script output variable to store complex data

Create a script output variable to store any complex data your script creates. The script output variables must match the values defined in the script. For example, create a contacts array of objects to store multiple contact objects. Save the contact object as a template so you can reuse it.

Map the action output to the script output variable

When you want a custom action to output complex data, add an action output and map it to the data pill for your Script step output variable. For example, create a contacts array and load the contact object template you saved earlier. Map the action output to the contacts array produced by your Script step.

Flow Designer and domain separation

Follow these general guidelines when using domain separation with Workflow Studio.

Ensure that tenant flows, actions, and subflows are run properly for domains

Since tenants cannot override Workflow Studio content, a service provider (SP) administrator from the TOP domain must author and manage them to ensure

they run properly for domains. While you can create domain-specific flows, users working from domains higher in the hierarchy may trigger multiple child domain flows. For example, a user working in the TOP domain can trigger flows in child domains such as ACME and INITECH.

> **Note:**
> Flow authors can see only Workflow Studio content available from their current domain and any parent domains in the hierarchy. Workflow Studio does not display content visible from Contains domains.

Provide a unique name for each flow, action, and subflow

Since all domains share Workflow Studio content, have an SP administrator in the TOP domain uniquely name each flow, action, and subflow. This ensures that a flow intended for one domain does not duplicate the name of a flow in another domain. For example, add the domain to the flow name such as `Validate incidents - TOP`, `Validate incidents - ACME`, and `Validate incidents - INITECH`.

Ensure that flows and actions only contain artifacts from current or parent domains

Workflow Studio prevents the activation of any flow containing artifacts unavailable to the current or parent domains. For example, if you create a domain-specific flow that belongs to the ACME domain, it cannot contain actions or subflows belonging to the sibling domain INITECH.

Edit Workflow Studio content in the domain to which it belongs

Users in a parent domain can't see flows, actions, and subflows in a child domain. They must change to the domain to which they belong to edit them. For example, an administrator in the TOP domain can't see flows from the ACME domain. The administrator must switch to the ACME domain to see and edit them.

Deployment

Avoid deploying newer release flows to instances on older releases

Workflow Studio does not support deploying newer release flows to instances running on earlier releases.

> **Danger:**
> The flow data model can change between releases, which can prevent newer flows from running or produce unexpected results when running on earlier release instances. Upgrade your instances to be on the same release versions before deploying them.

Flow error handling

Follow these general guidelines to achieve the benefits offered by flow error handling.

Avoid adding error handling items to the main section of the flow

A flow normally stops running when an action or subflow returns an error in the main section. A stopped flow cannot run any actions or subflows past the point where it returned an error. Adding error handling actions and subflows to the Error Handler section ensures they run them when there is an error.

Capture Error Status information

The Error Status object contains information about the action that produced an error. You can use this information to identify the cause of the error as well as record data that may need correction.

Suppress subflow error messages

You can enable the Error Handler for a subflow to prevent its errors from cascading to a parent flow. Leaving the subflow Error Handler section empty ensures that it always generates the **Completed (error caught)** state.

Use subflows to avoid the 10-item limit

Rather than force your error-handling-process to fit within a 10-item limit, call subflows, which can contain many more items. You can also use the subflow outputs to trigger automation in other flows.

Use subflows to take corrective actions

Rather than recreate the same sequence of actions in multiple flows, create reusable subflows to correct errors to your record data. When a flow error leaves your record data in an undesired state, use subflows to correct these records. You can use the error handler to identify such record data as a subflow output.

Action error evaluation

Follow these general guidelines to achieve the benefits offered by action error evaluation.

Allow only independent steps to continue running

Allow a step to continue running if it does not return data required by a later step. If a step provides data necessary for later steps, then you know that the later steps cannot run successfully.

Avoid more than 10 error conditions

While there is no limit to the number of error conditions you can create, each error condition requires evaluation. The more error conditions your action has to evaluate, the potentially slower your action can run.

Identify specific step failures

You can use the Step Status to identify when a specific step fails. Identifying a specific step can be useful when your action contains multiple instances of the same type of step. You may also want to identify a specific step so that a flow error handler can take appropriate corrective actions for the failure.

Put specific error conditions before general error conditions

Error evaluation stops when the action finds a matching error condition. Putting general error conditions first may prevent the action from ever matching specific error conditions.

Use descriptive error condition labels

Identify an error condition without having to edit it. By default, you can only see error conditions when you edit them.

Flow Administrator

Turn flow reporting off in production

Minimize the amount of memory required to run flows by disabling Flow reporting. Flow reporting stores configuration and runtime information for the Execution Details page. These reports are good for troubleshooting, but requires a large amount of data to be retained both in memory and in the database. By default, flow reporting is disabled, and the system only generates execution details when you manually test a flow or action. Instead you can use log files, which are still available when reporting is turned off.

Reduce the amount of memory consumed in flows with nested looping

When reporting is activated, set com.snc.process_flow.reporting.iteration.lastn to a value of "1" to reduce the amounts of the amounts of memory that previous loop iterations consume. The more iterations you report on, the more memory is required.

View final transformed values in the flow execution details

Only the final transformed value appears in the flow execution details, and not the value for each applied transform.

Flow Priority

Follow these design considerations when setting flow priority.

Avoid setting all flows to run at high priority

Use a mix of priorities rather than set all flows to high priority. Worker threads use the relative priority between flows to select work. If all of your flows run at high priority, then there are no lower-priority flows to make wait.

Avoid setting flow priority for flows that have to pause

Keep flows that have to pause at the default medium priority since a flow that pauses loses its priority value when it resumes running.

Use high priority for business critical flows

Limit high-priority to flows that have high business value, run rarely, and have a short runtime. Avoid setting high-volume flows to high priority as doing so limits the number of worker threads available to run other flows. A long-running high priority flow can also reduce the worker threads available to run other flows.

Use low priority for high-volume flows

Run high-volume flows at low priority so that other time-sensitive flows can run first. Low-priority flows shouldn't be time-sensitive.

Use medium priority for time-sensitive flows

Use the default flow priority when a flow has some time urgency when compared to other flows.

Related topics

https://learning.servicenow.com/

Workflow Studio actions

Actions can be added to any flow, enabling process analysts to automate ServiceNow AI Platform features without having to write code.

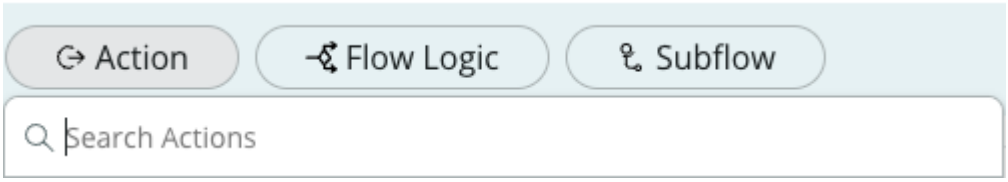
An action is a reusable operation that enables process analysts to automate ServiceNow AI Platform features without having to write code. For example, the **Create Record** action allows process analysts to generate records in a particular table with particular values when certain conditions occur. ServiceNow core actions like Create Record require some familiarity with ServiceNow AI Platform tables and fields. Action designers can create application-specific actions to pre-set configuration details. For example, creating a Create Incident Task action ensures that the process analyst uses the correct table and field configuration each time the action is used. You can add application-specific actions by activating the associated spoke.

In Workflow Studio, a process analyst adds actions to a flow and defines the configuration options.

Search actions

You can use the **Search Actions** filter to find an action by name or spoke. As you enter data, Workflow Studio displays a list of actions and spokes that match your search criteria.

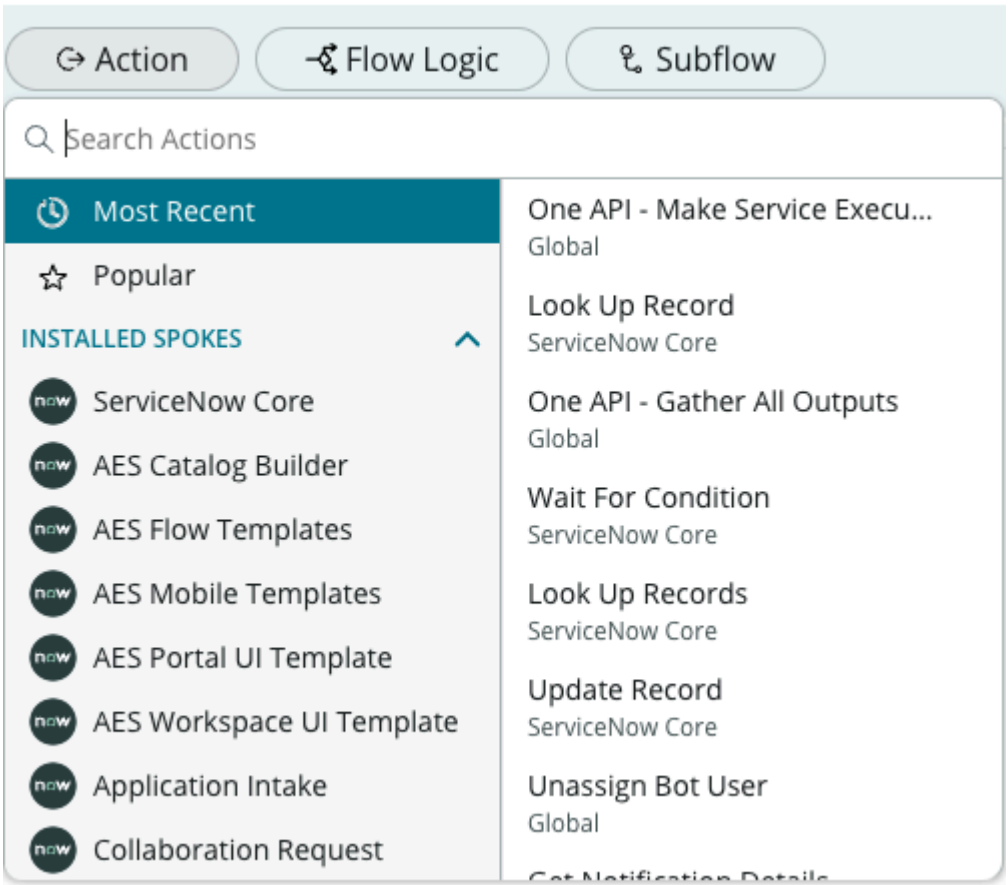
Search Actions filter



Most Recent

You can use the **Most Recent** option to display a list of the actions you recently selected. Each action displays the action name and the spoke to which it belongs underneath the name. You can use the information icon to see more information about the action such as its description, inputs, and outputs.

Most Recent actions



Popular

You can use the **Popular** option to display a list of actions that your organization frequently uses. The system runs a scheduled job every seven days to generate the list of popular actions.

Installed spokes

Workflow Studio displays actions for each installed spoke. You can select a spoke name to see a list of available actions for the spoke. All instances have a ServiceNow Core spoke.

A ServiceNow core action is an action available to any flow regardless of the spokes installed. ServiceNow core actions cannot be viewed or edited from the Workflow Studio flow design environment. For example, the **Ask for Approval** action is a ServiceNow core action that allows process analysts to use ServiceNow AI Platform approvals. Workflow Studio provides a set

of ServiceNow core actions to automate ServiceNow AI Platform processes. You can add application-specific actions by activating the associated spoke.

You can find any custom actions you created in the spoke to which they belong. Alternatively, use the Search Actions filter to search for actions by name.

Add Worknote Link to Context action

Add a journal field entry containing a link to the current flow context record. Use the link to view the flow execution details of the current flow. You can add a flow context link to any record that has a journal field.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Table

Data type: *`Table Name`*

Table name containing the record you want to update with a link.

Record

Data type: *`Record`*

Record where you want to add a link to the flow execution details. You can use the Look Up Record action to find an appropriate record.

Journal Field

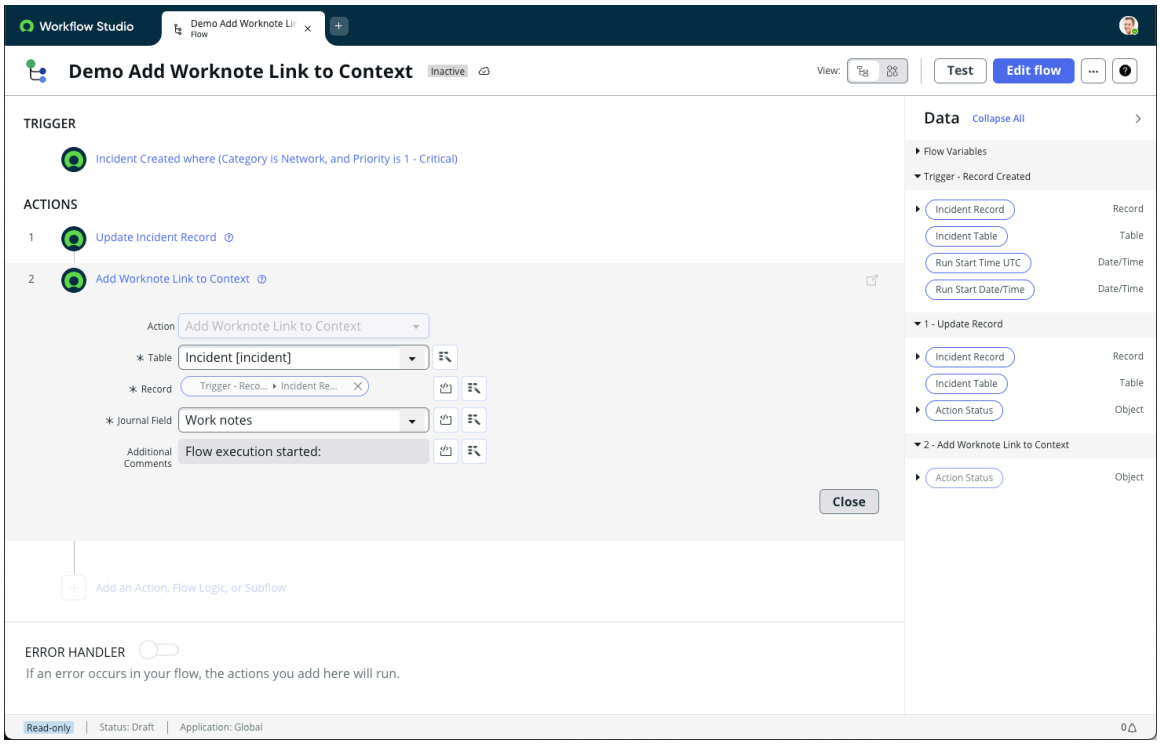
Data type: *`Field Name`*

Journal field to insert the link to the flow execution details. For example, the work notes field of a task record.

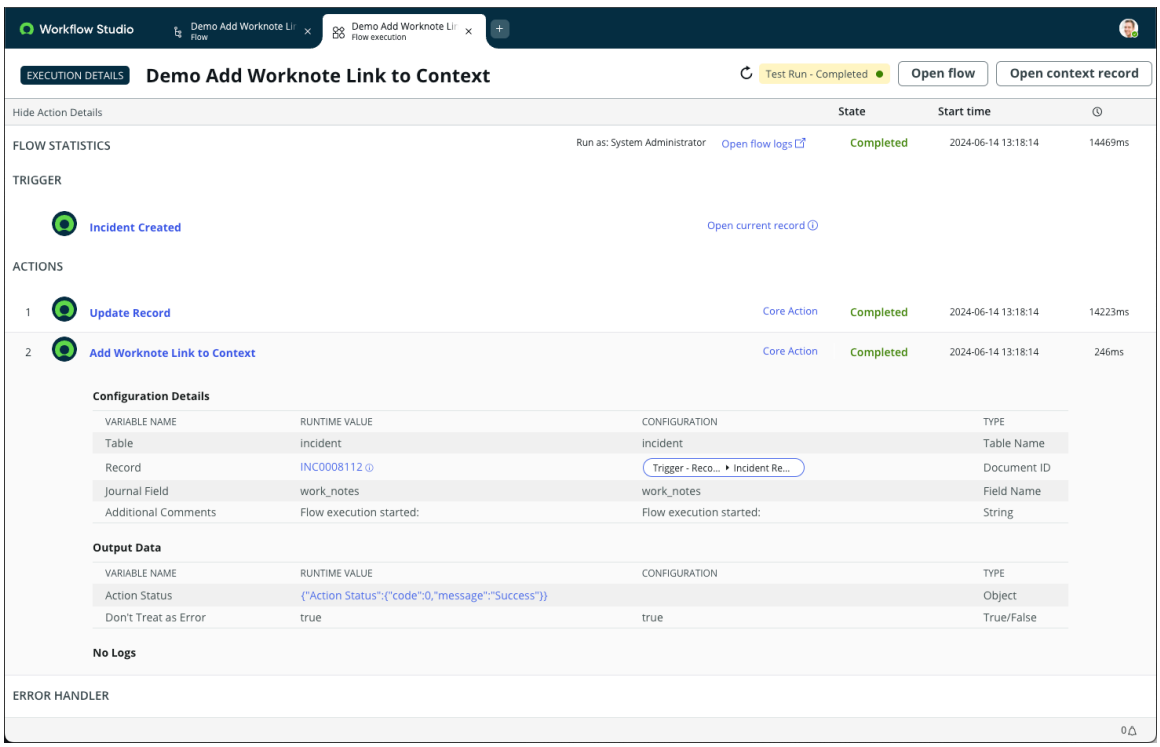
Outputs

This action has no outputs. Instead it updates the selected record to add a direct link to the flow execution details for the flow that ran this action.

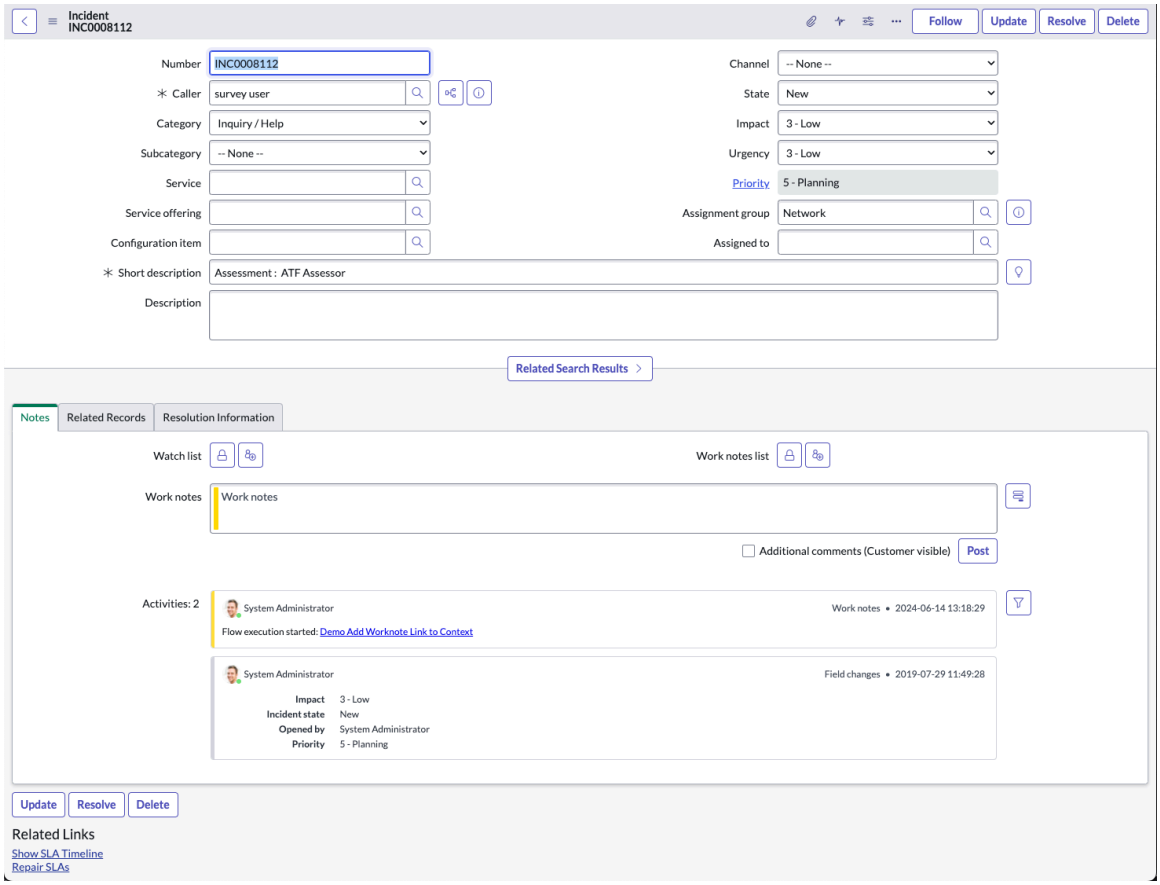
Example: Add a link to an incident record



In this example, a flow runs when an priority 1 incident is created in the Network category. The Update Record action assigns the incident to the Network assignment group. The Add Worknote Link to Context action updates the trigger incident record. The action adds the link to Work notes journal field.



The execution details show that the trigger incident record INC008112 was updated to add a link to the work notes field.



The incident record work notes field contains a text message and a link to the flow execution details for the flow.

Ask for Approval action

Request approval for a record with an approval field. You can configure a rule set for an approval, rejection, or cancellation. If a due date is added to an approval, the approval is automatically approved, rejected, or canceled if the approvers have not responded by the designated time.

Classic approvals is a platform feature that enables users or groups to approve or reject a task.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Record

Data type: *Record*

Reference to the record to approve. If the record contains an approval field, Workflow Studio automatically sets the Approval Field input.

Table

Data type: *Table Name*

Table name of the record associated with the approval request. The table that you select must support approvals by having an approval state field. For example, the Task table and its extensions contain approval fields.

Approval Field

Data type: *Field Name*

Field containing the results of approval requests.

Journal Field

Data type: *Field Name*

Field to store history and comments associated with the approval request.

Rules

Data type: *Approval Rules*

Approval and rejection rules to determine which users can approve or reject requests, and what happens after approval or rejection.

Approval or rejection rules include:

- Anyone approves
- All users approve
- All responded and anyone approves
- % of users approve
- # of users approve

In the field beside the approval rule, add the desired approvers. To add approvers:

- Select individual users or groups.
- Drag or select a field from a record.
- Select Manual approvers to allow a manual approver to process an approval or rejection. A manual approver is a user manually added to the Approvers related list who can then approve the request. For example, you can manually add a subject matter expert to a task to approve the request. To learn more about adding manual approvers, see Generate approvals using the approvers related list.

Note:
By default, Ask for Approval generates approval records for inactive users and groups. This behavior allows a flow or action to continue working even when a specific user or group is later made inactive. If you want to change the behavior of generating approvals for inactive entities, set the com.glide.hub.flow.approval.allow_inactive_entity system property. See Workflow Studio flow system properties.

Define rejection rules by adding another OR rule set. When defining approvals, include rejection rules that run when there are no matching approvals. Such rejection rules prevent the flow from remaining in a waiting state. For example, if an approval can be approved by anyone, create a time-based rejection rule in case no one approves it.

Note:
If you set an approval rule with no rejection rule (or vice versa) and the expected approval state is not met, the runtime value will be **canceled**.

For information about how to use inline script to specify approval rules, see the Scripted Approvals in Flow Designer with Flow Variables blog post on the ServiceNow Community.

Due Date

Data type: *Schedule Date/Time*

Due date for an approval state to prevent the flow from endlessly waiting for approval.

Output

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Approval State

Data type: *Choice*

Completion state of the approval request. The flow execution details page displays one of these values.

- Not Yet Requested [not requested]
- Requested [requested]
- Approved [approved]
- Rejected [rejected]
- Cancelled [cancelled]
- No Longer Required [not_required]
- Skipped [skipped]

Example

In this example flow, the flow asks for approval once an incident is created or updated. The Ask for Approval action waits one day for approval using an 8-5 weekday schedule. After one business day has passed, if no one has approved or rejected the request, the request is approved. The action sends an approval request to the manager of the person listed in the Assigned to field. That manager can approve or reject the request within one business day.

General guidelines

Follow these guidelines when asking for approvals.

Do not duplicate ask for approval actions in Do the following in parallel flow logic

Workflow Studio does not support making multiple approval requests to the same record using Do the following in parallel flow logic. Asking for approval on the same record creates a dependency between branches, which can produce unexpected results since there is no way to know which branch will complete first.

Associate Record to Email action

Associate a record with an Email [sys_email] record so that you can track which record is affected by the email.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Fields

Field	Description
Email Record	Email [sys_email] record that the Target Record associates with.
Target Record	Record to associate to the email record

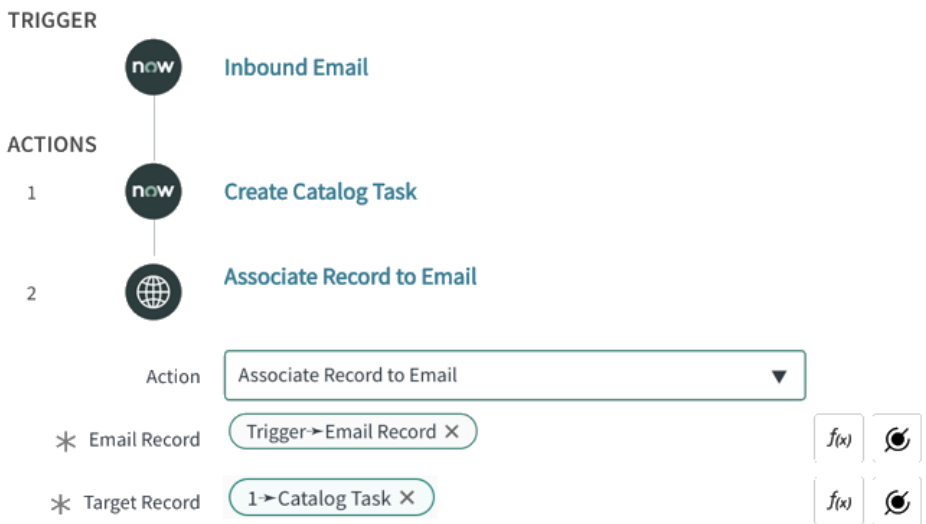
Output

This action updates the **Target** field on the Email [sys_email] record.

Example

In the following example, a process owner adds the Associate Record to Email action under an inbound email trigger. The user has also added a Create Catalog Task action in the flow. In the **Email Record** field, the user selects to associate a record to the email that triggered the flow. In the **Target Record** field, the user selects to associate the Catalog Task [sc_task] record that is created in the Create Catalog Task action.

Using the Associate to Email action



Create Catalog Task action

Creates a record in the Catalog Task [sc_task] table associated to a requested item in the Requested Items [sc_req_item] table. Adds the catalog task record as data to be used in the flow.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Table

Data type: *Table Name*

Extension of the Task table in which to create a record. For example, Catalog Task [sc_task] or Incident Task [incident_task].

Requested Item [Requested Item]

Data type: *Record*

The requested item record from the Requested Item [sc_req_item] table that this catalog task fulfills.

Short Description

Data type: *String*

The short description for the catalog task.

Fields

Data type: *Template Value*

The field values that you want to set in the catalog task. If adding the action to a subflow, you can allow flow designers to dynamically set field values. See Create a template value input.

Wait

Data type: *True/False*

Flag indicating whether to pause the flow until the Task record is no longer active. You can add a wait condition by dragging-and-dropping a True/False data pill into this input. The flow only waits for the Task record to complete when the condition field is true.

Template Catalog Item [Catalog Item]

Data type: *Record*

The Catalog Item [sc_cat_item] record you want to use to populate the catalog variables slush bucket. This input does not support any data pill values.

Catalog Variables

Data type: *Slush Bucket*

The list of catalog variables that you want to show on the catalog task. These variables provide more information to the catalog task fulfiller.

Outputs

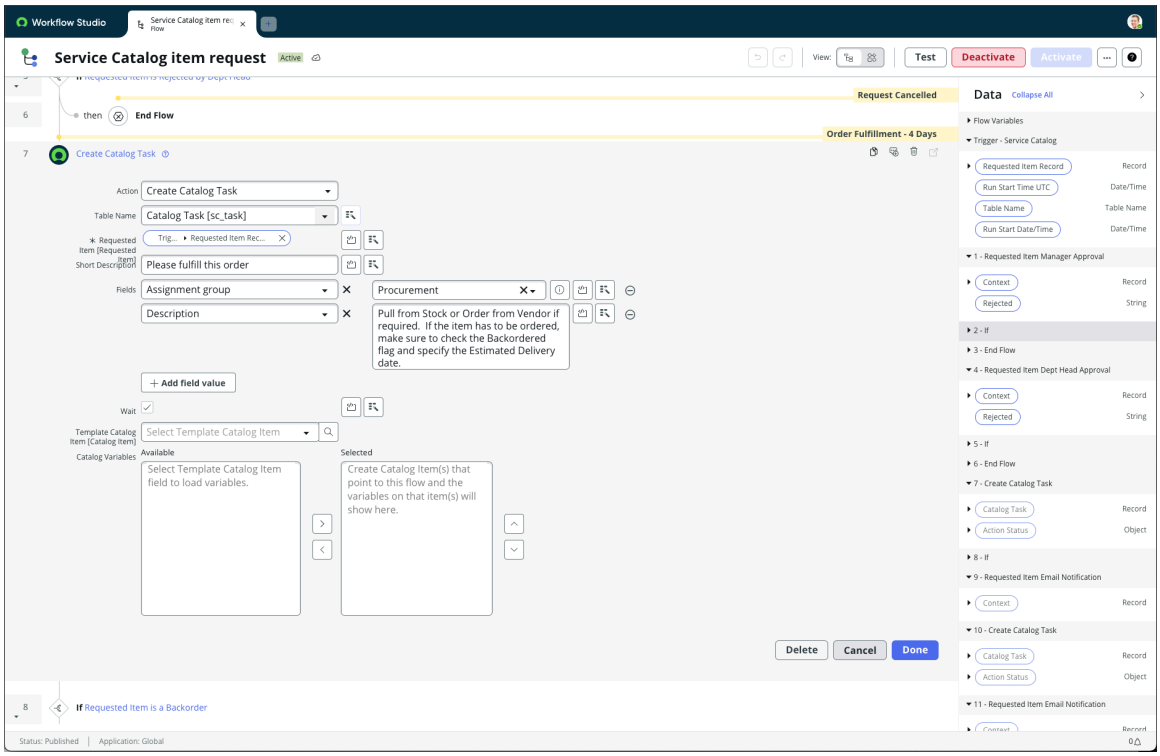
These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Catalog Task

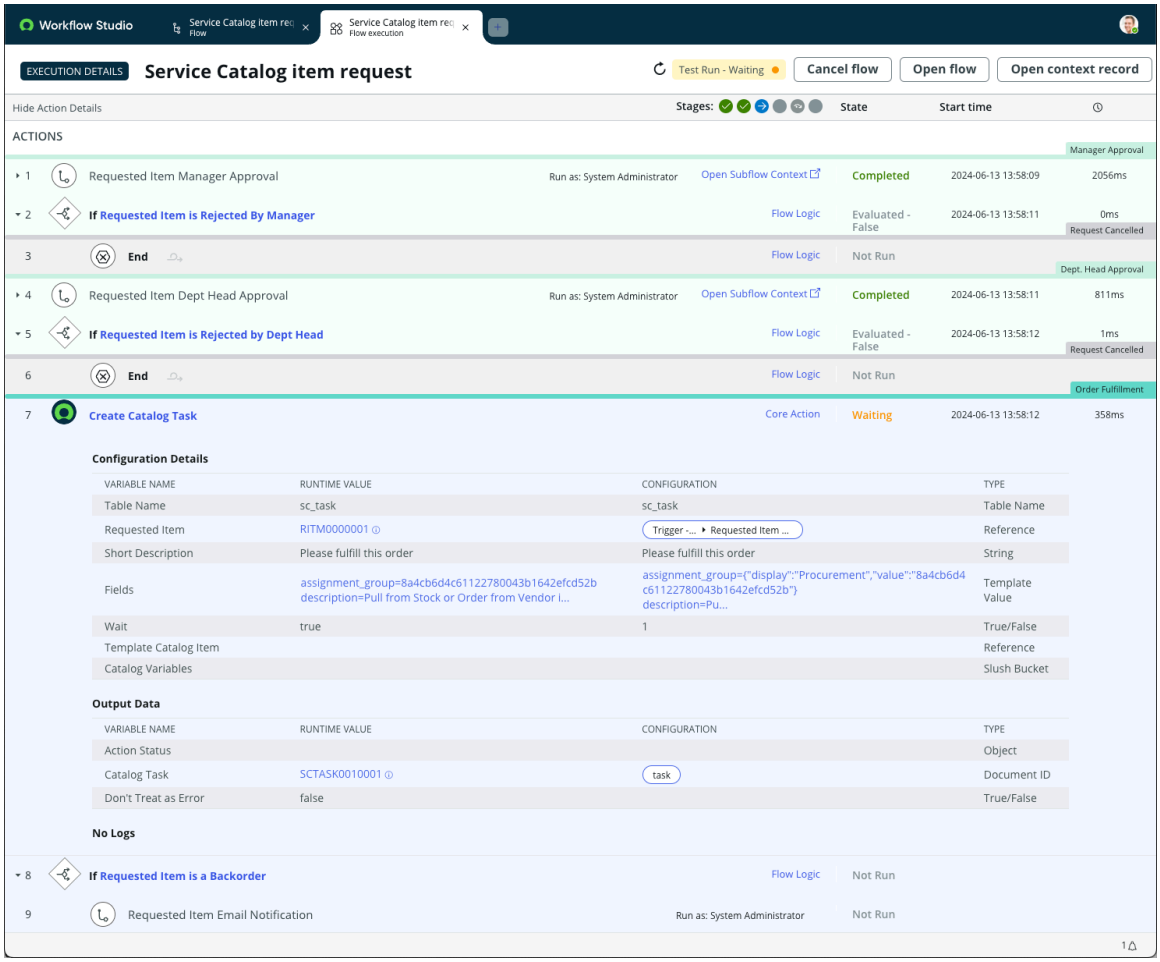
Data type: *Record*

Reference to Catalog Task record created.

Example: Use a flow to process a catalog item request



In this example, the Service Catalog Item request flow first requests manager and department head approval. Only approved requests ever run the Create Catalog Task action. The Requested Item input uses the requested item that triggered the flow to run. The Fields input sets the value of the task short description, assign group, and description. Since the wait input is selected, the flow will pause while the catalog task is being fulfilled. Also since there is no template catalog item selected, there are no catalog variables available to this task.



In this example, the flow execution details show the requested item record number and the catalog task record number. The action state is listed as waiting since the Wait input was selected. The flow waits until the catalog task is closed before continuing.

Create Flow Data action

Collect data from agents interacting with a Workspace playbook. Use this data to create reusable activities for process owners using Playbooks.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

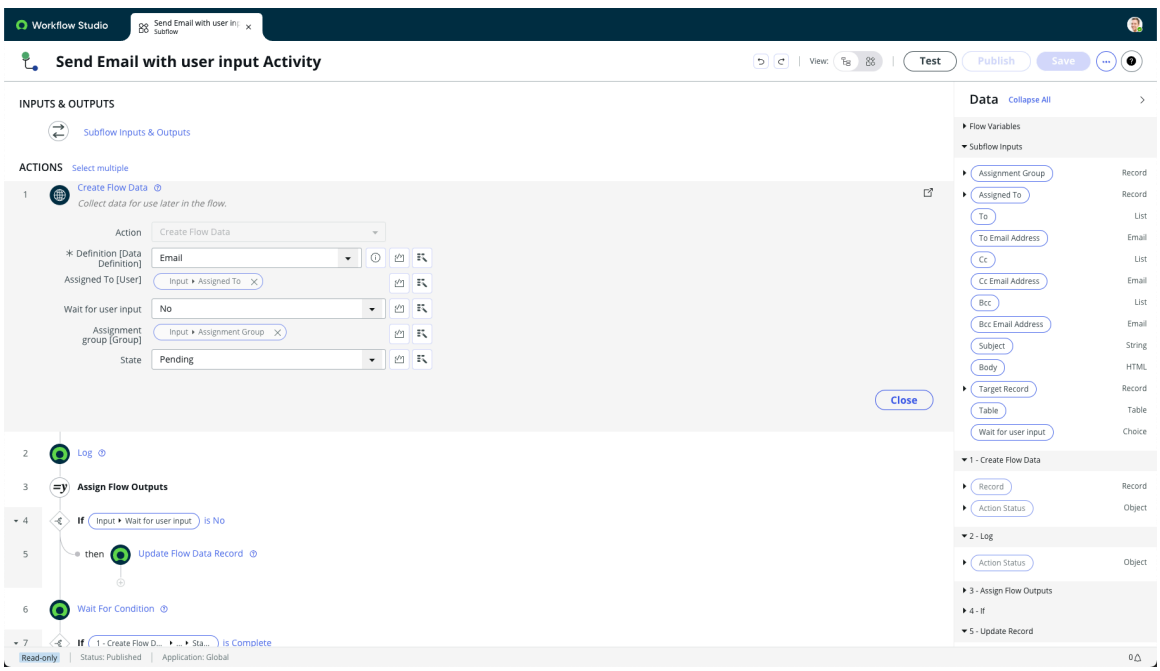
Input	Type	Description
Definition	Reference.Data Definition[sys_flow_data_definition]	Reference to the Data Definition record used to collect data for the flow. Each Data Definition record has a set of Flow Data Variables to store data. For example, the Create Task Data Definition record collects values for Task records.

Input	Type	Description
Assigned To	Reference.User[sys_user]	User responsible for completing the task associated with the Playbooks activity.
Assignment Group	Reference.Group[sys_user_group]	User group responsible for completing the task associated with the Playbooks activity.
Wait for user input	Choice	Option to prompts Playbooks users to determine if the activity pauses for input in a playbook during runtime. Options include: • **Yes** - Pause the activity and prompt end users for input • **No** - Don't pause the activity for end user input

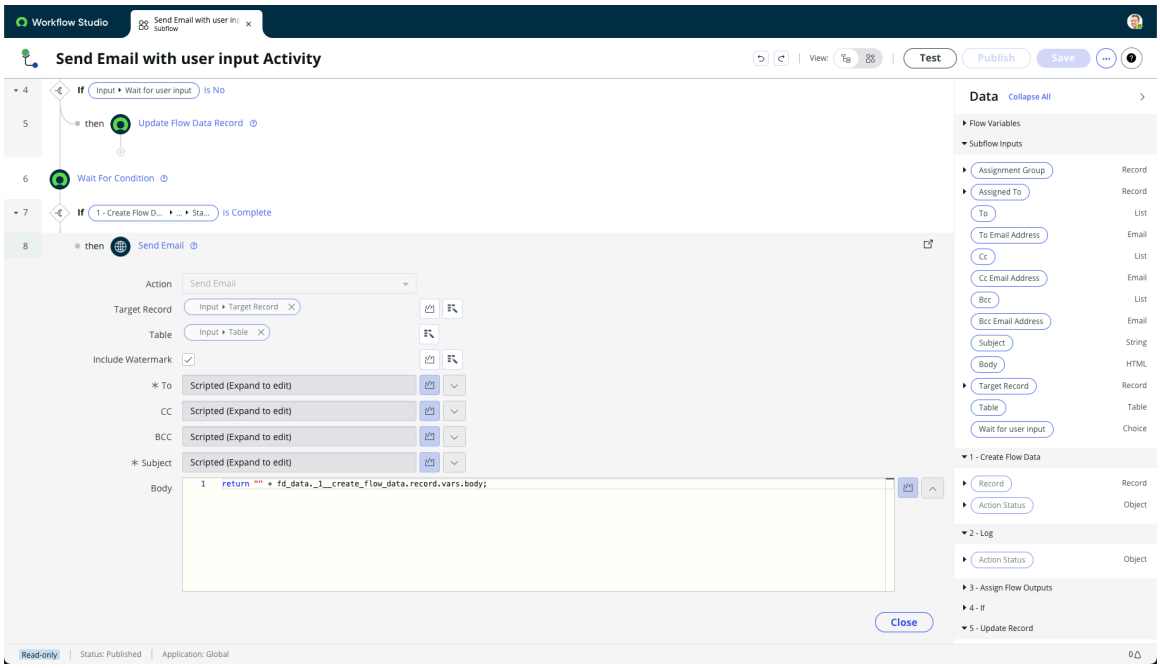
Outputs

Output	Type	Description
Record	Reference.Flow Data[sys_flow_data]	Reference to the Flow Data record created from end user input in a playbook activity during runtime.

Example: Send Email with user input Activity



Send Email with user input Activity is a subflow that gathers the information needed to create and send an email. This subflow provides data for the corresponding Playbooks activity. The subflow uses the Create Flow Data action to gather information from the person running the activity. This example configures the Create Flow Data action to use the Email Data Definition record, which includes variables for an email subject and body. The action is also configured to use the subflow inputs for Assigned To and Assignment Group to determine who can enter flow data.



The subflow waits until the Create Flow Data record is in the Complete state before sending an email. The Send Email action is configured to use the body variable from the Create Flow Data action as the body of the email.

Create Record action

Creates a record on any table. You can dynamically add and configure fields for the record.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Table

Data type: *Table Name*

Table in which to create record.

Fields

Data type: *Template Values*

Field values to set for the record. For example, to set the short description to a certain value, select **Short description** and set the desired value.

Important:
The system does not support updating multiple journal fields such as the additional comments or work notes of a task record.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Table

Data type: *Table Name*

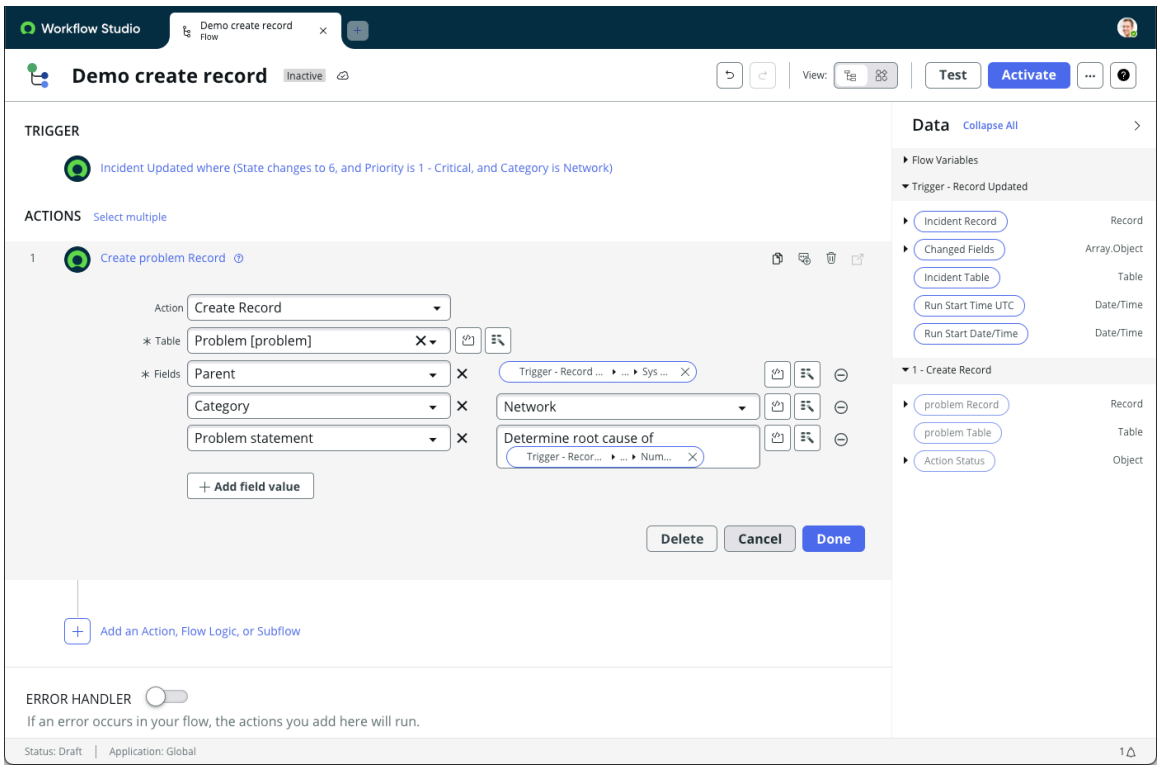
Table where record was created.

Record

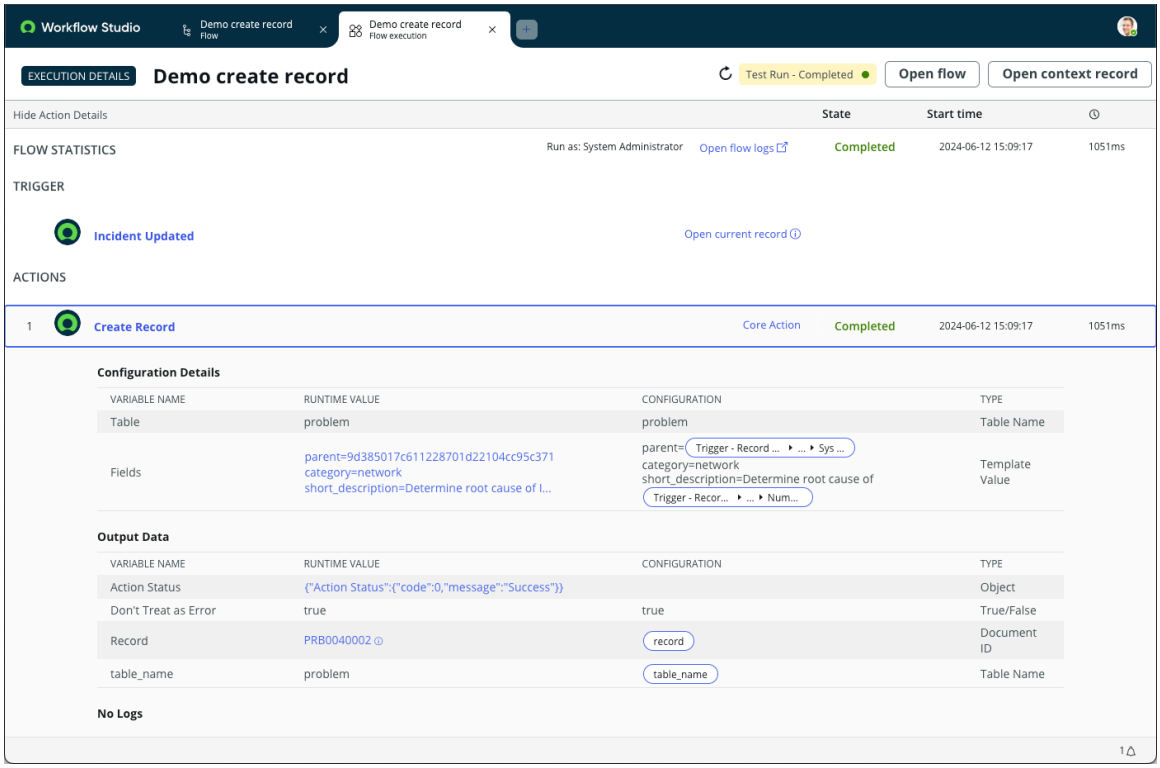
Data type: *Record*

Reference to record created.

Example: Create a problem record from an incident record



In this example, the flow runs when a priority 1 network incident changes to the resolved state. The Create Record action creates a problem record where the Parent is set to the trigger incident's Sys ID, the Category is set to Network, and Problem statement is set to Determine the cause of the trigger incident's number.



In this example, the action creates problem record PRB00400002, which was created from INC0000002. You can use the Record runtime value to open the new record.

Create or Update Record action

Create or update a record in a ServiceNow table using a single operation. Update a record that exists, or create a record using the values provided.

Identification of existing records

The Create or Update Record action identifies existing records by searching for matching values in the fields that you select as unique identifiers. For example, you can specify that the short description and priority fields uniquely identify an incident. When the action finds an incident with a matching short description and priority, it updates the matching record rather than creating a new record.

Note:

- If no field is selected as a unique identifier, the action creates a record with the field values provided.
- If more than one record matches the value of the unique identifiers, the action doesn't update any records and displays an error message in the flow execution details.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Table

Data type: *Table Name*

Table in which to create or update a record.

Fields

Data type: *Template Values*

Field values to set or update for the record. For example, to set the short description to a certain value, select **Short description** and set the desired value.

To learn about creating template value input, see Create a template value input.

> **Important:**
> The system does not support updating multiple journal fields such as the additional comments or work notes of a task record.

If adding the action to a subflow, you can Create a template value input. Dynamically set field values can trigger server-side validation rules but cannot trigger UI policies.

Determines uniqueness

Data type: *True/False*

Option for selecting the field as a unique identifier, which determines when to update or create a record. A record is updated when the incoming field value matches an existing record field value. A record is created when the incoming field value does not match an existing record field value. This option appears when the required table name and fields are selected.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Record

Data type: *Record*

Reference to record created or updated.

Table

Data type: *Table Name*

Table where record was created or updated.

Status

Data type: *Choice*

Completion status of the action. The flow execution details page displays one of these values.

- Created [created]: The action created a record.
- Updated [updated]: The action updated a record.
- Error [error]: The action produced an error.

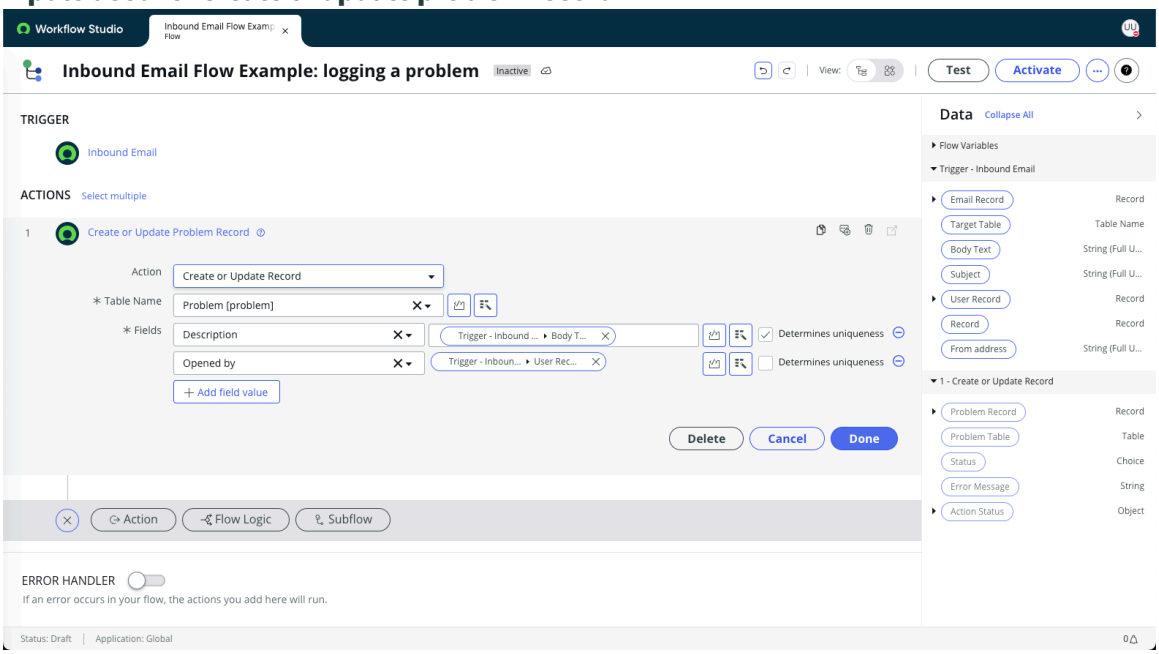
Error Message

Data type: *String*

Error message produced when the record operation fails.

Example: Update problem record from inbound email

Inputs used for create or update problem record



Create Task action

Create a task record in an extension of the Task table. After you choose the task table, you can dynamically select the fields to configure the action. Defining the Parent field associates the task to a parent record.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Table

Data type: *Table Name*

Extension of the Task table in which to create a record. For example, Catalog Task [sc_task] or Incident Task [incident_task].

Field Values

Data type: *Template Values*

Field values to set for the record. For example, to set the short description to a certain value, select **Short description** and set the desired value.

Wait

Data type: *True/False*

Flag indicating whether to pause the flow until the Task record is no longer active. You can add a wait condition by dragging-and-dropping a True/False data pill into this input. The flow only waits for the Task record to complete when the condition field is true.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Table

Data type: *Table Name*

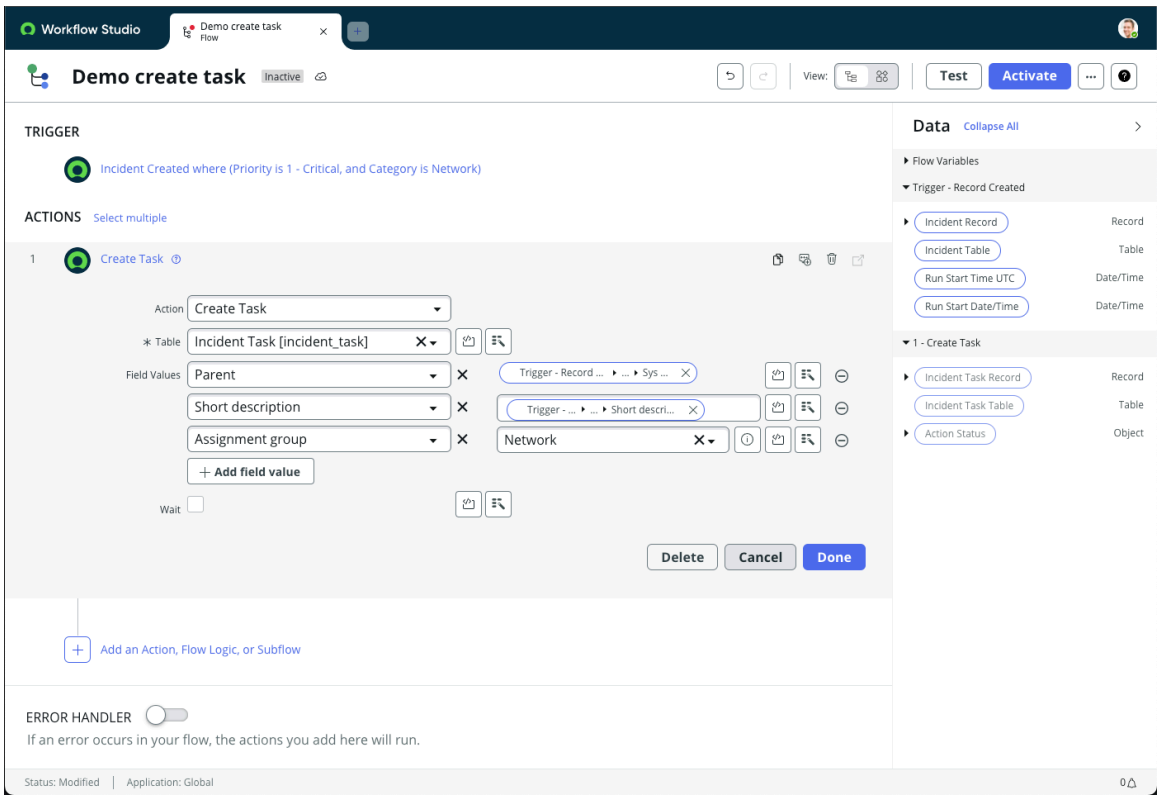
Table where the Task record was created.

Task

Data type: *Record*

Reference to the Task record created.

Example: Create an incident task from an incident



In this example, a flow starts when a high priority network incident is created. The flow creates an Incident Task record where the Parent field is set to the trigger incident, the Short Description inherits the incident short description, and the Assignment group is set to Network. Since there are no more actions in the flow, there's no need to select the Wait option to pause the flow until the Incident Task is complete.

Copy Attachment action

Copies an attachment from the Attachments [sys_attachment] table to a target record.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Source Attachment Record [Attachment]

Data type: *Record*

Attachment record containing the attachment you want to copy. You can either manually select a record from the Attachment [sys_attachment] table or look up the attachment record. You can use the Look Up Attachment action to provide one or more Sys ID values of attachments with a given file name in a specific source record. If you use the Look Up Attachment action, you must also use a Look Up Record action to convert the Sys ID value returned by the Look Up Attachment action into a usable record data pill. See the example section for an illustration of converting the Sys ID output of the Look Up Attachment action into a record data pill.

Target Record

Data type: *Record*

Record where you want to add a copy of the attachment. You can use the Look Up Record action to find an appropriate target record.

Table

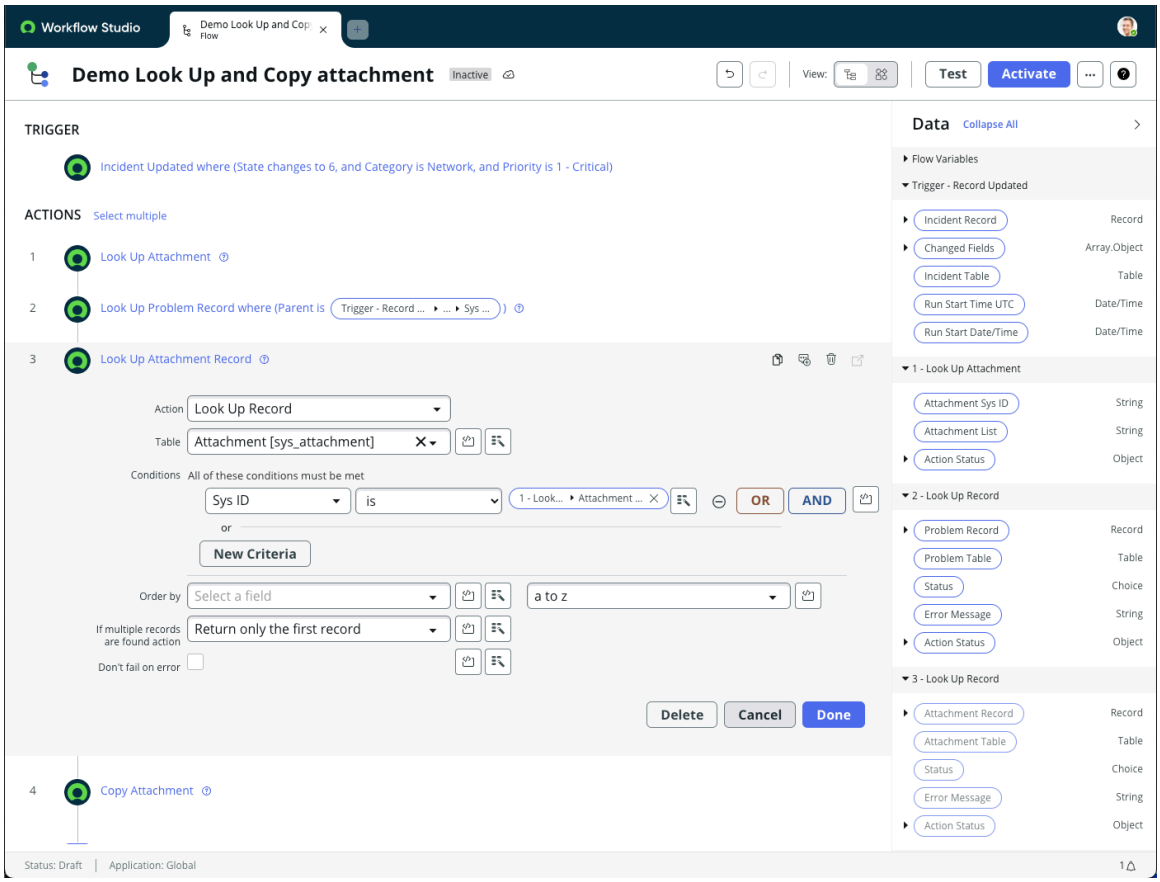
Data type: *Table Name*

Table containing the target record. This system sets this input value for you if you select a data pill for the target record.

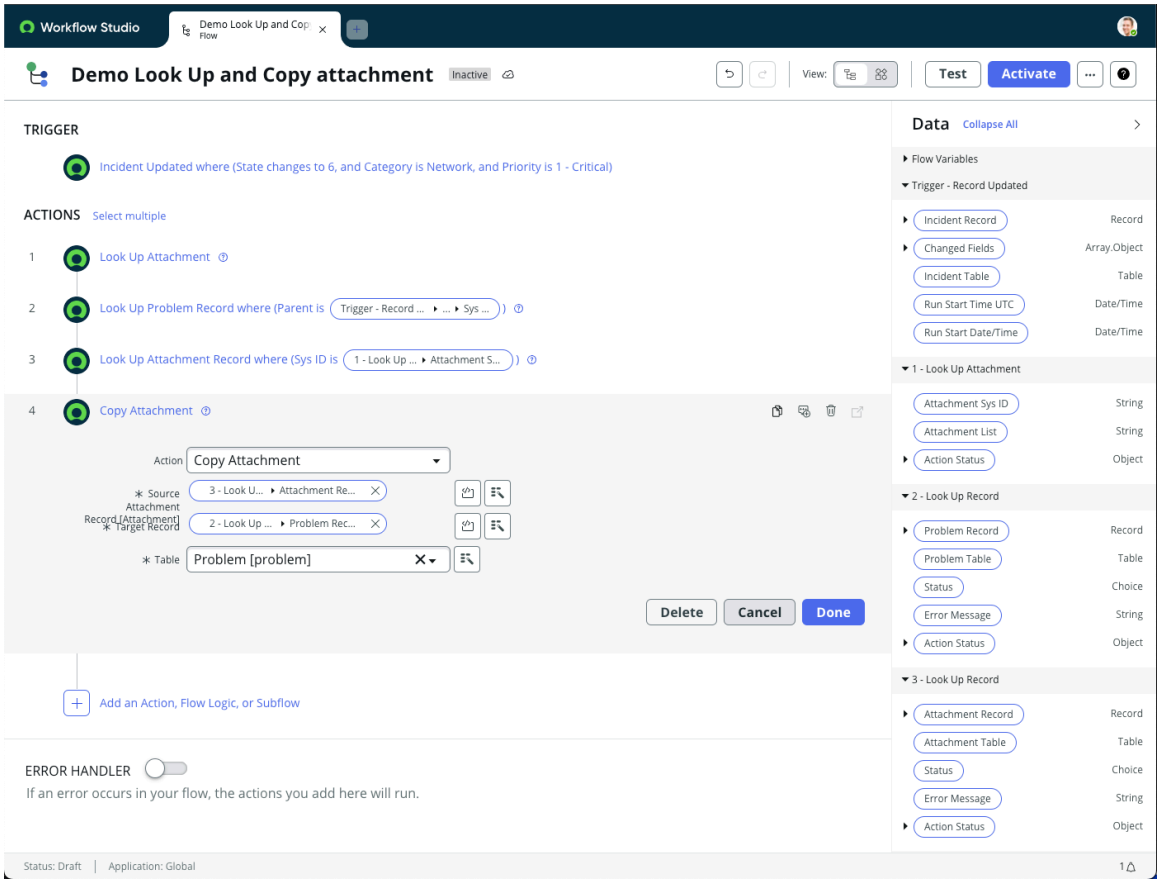
Outputs

This action has no outputs.

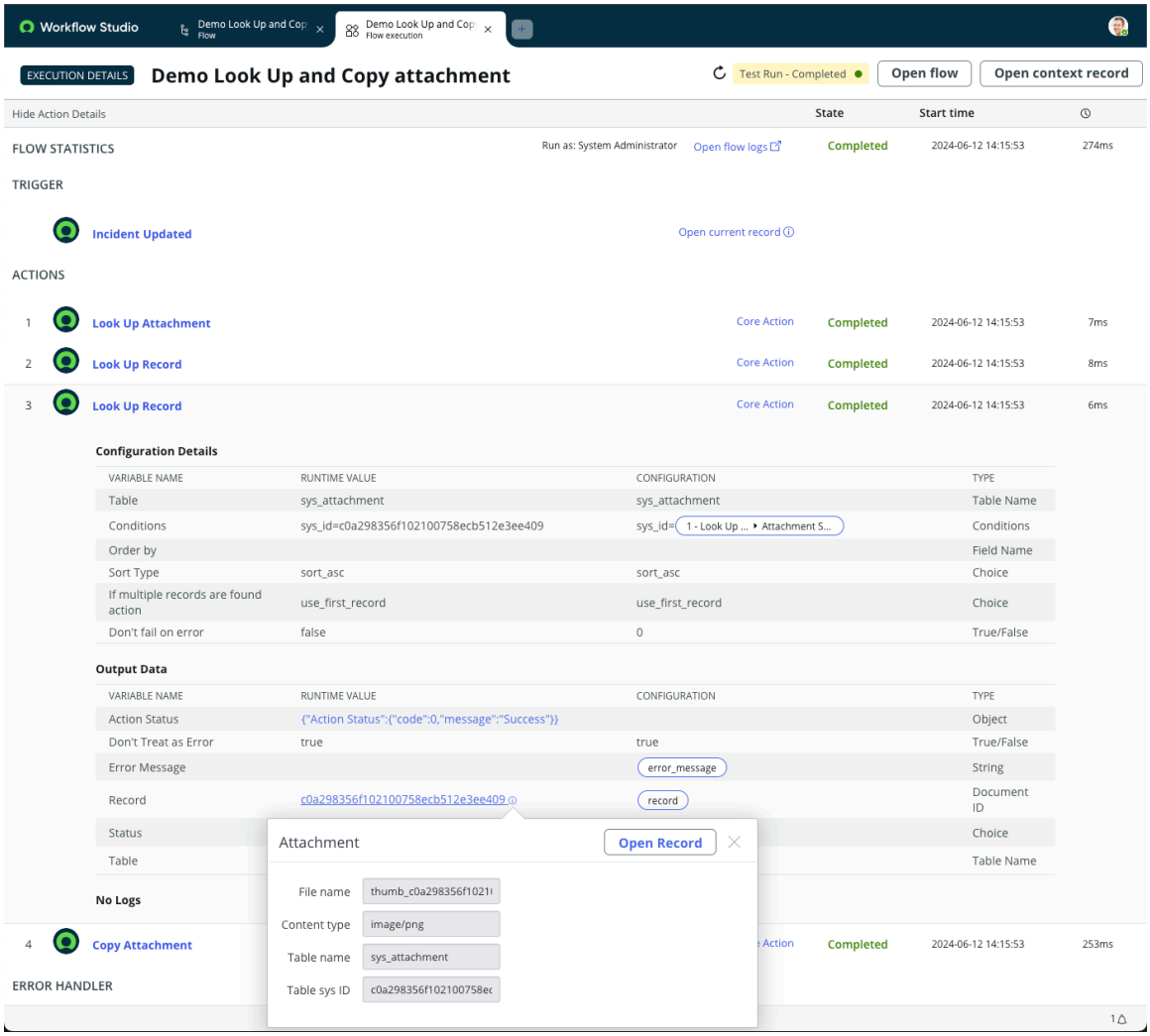
Example: Copy attachments from incident records to a problem records



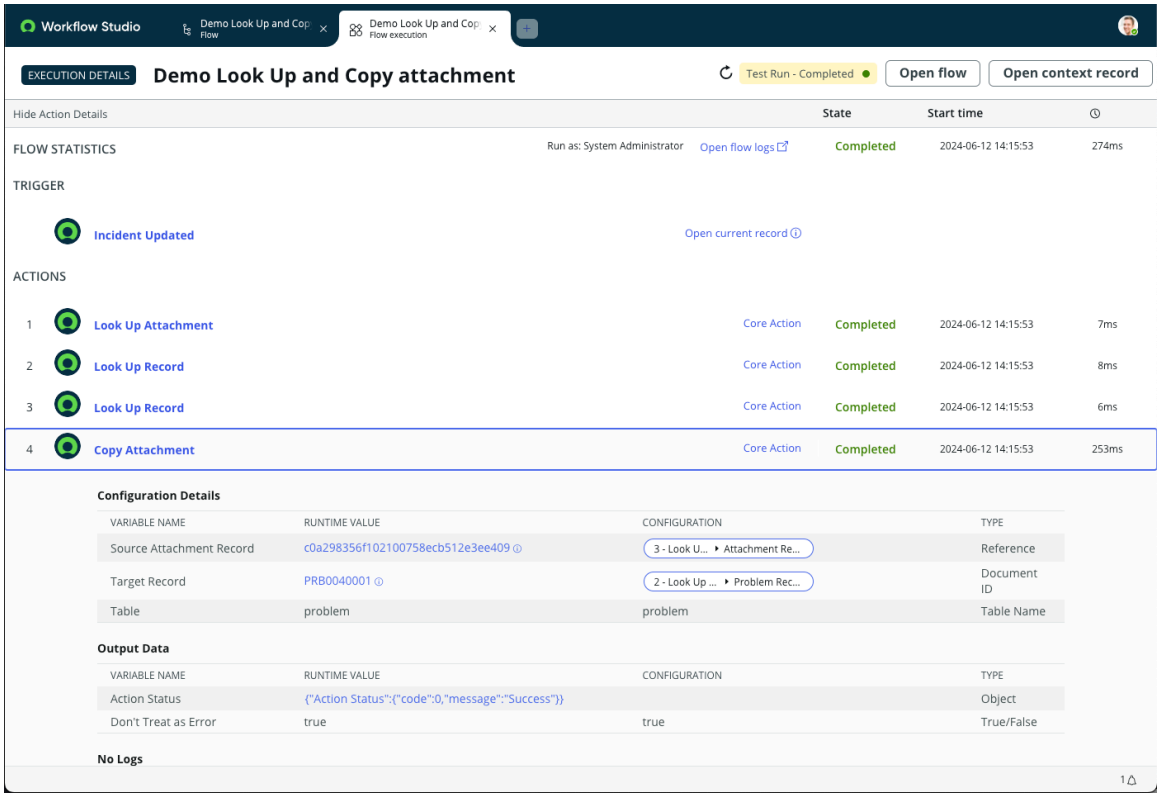
This example copies attachments from the trigger incident record into an associated problem record. This example assumes that there is only one source attachment to copy. If you need to copy multiple attachments, add a For Each flow logic. The flow first uses the Look Up Attachment action to find the attachment within the trigger incident. The Sys ID value of produced by this action is used as an input in the Look Up Record action in step 3. The flow next looks up a problem record associated with the incident record.



In this example, the Source Attachment Record input gets its value from the Look Up Record action for the Attachment [sys_attachment] table, because you can't use the string output of the Look Up Attachment action directly. The Target Record input gets its value from the Look Up Record action for the Problem [problem] table. The Table input inherits its value from the Target Record data pill.



The execution details for flow step 3 show that the Look Up Record action has found a specific attachment record. This record is used as the input for flow step 4 Copy Attachment.



The execution details for flow step 4 show that both input values use the output data pills of Look Up Record actions. The Source Attachment Record input uses the data pill from flow step 3, and its runtime value is a specific Attachment record. The Target Record input uses the data pill from flow step 2, and its runtime value is a specific problem record.

Delete Attachment action

Removes one or all attachments associated with a record and deletes the attachment record from the Attachments [sys_attachment] table.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Fields

Field	Description
Source Record	Drag a Record data pill from the data panel to delete one or all attachment records from.
Table	Automatically populates with the **Source Record** table.
Attachment File Name	Enter the name of the attachment file to delete a single attachment associated with the selected record. **Note:** If a record has multiple attachments with same name, all matching attachments are deleted.

Field	Description
Delete All Attachments?	Select to delete all attachments associated with the selected record.

Delete Record action

Deletes a record on any table.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Fields

Field	Description
Record	The record to be deleted. Drag-and-drop a record data pill or use the data pill picker to select a record.

Fire Event action

Create a system event record in the Event [sysevent] table to be processed by a scheduled event handler. Pass event parameters using flow data.

Roles and availability

Available as a ServiceNow Core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

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.

Event [Event Registration]

Data type: *Reference.Event Registration[sysevent_register]*

Event Registration record that defines the type of system event that you want to create.

Record

Data type: *Record*

Sys ID of the record that triggered the event.

Table

Data type: *Table Name*

Table containing the record that triggered the event.

Parameter 1

Data type: *String*

Text input to specify the first event parameter.

Parameter 2

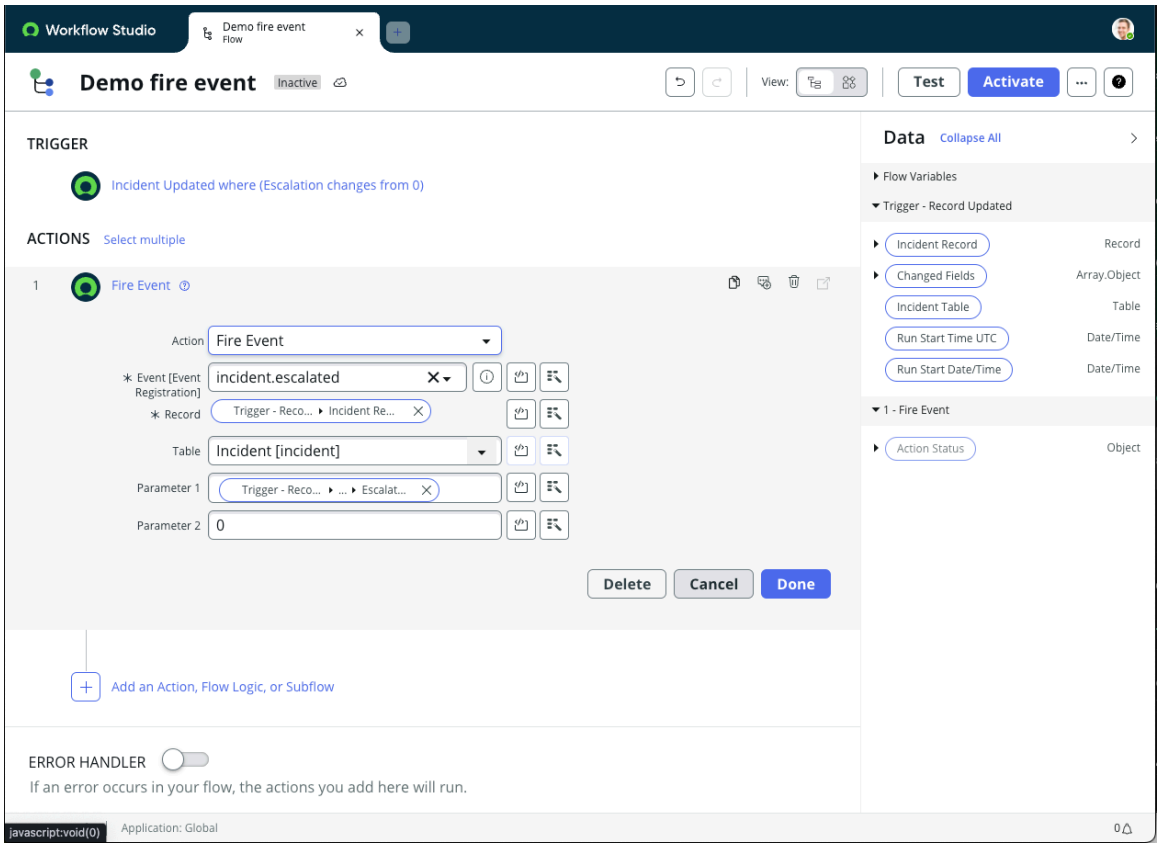
Data type: *String*

Text input to specify the second event parameter.

Outputs

This action has no outputs.

Example: Fire an incident escalated event



This example creates an incident.escalated event when an incident record is updated to change the value of the Escalation field. The incident.escalated event expects the value of parameter 1 to be the current value of the Escalation field and parameter 2 to be the previous value of the Escalation field.

Get Attachments on Record action

Access the list and count of the attachments associated with the provided source record as data pills in a flow. Use flow logic or scripting to process each attachment in the list of the attachments that the action returns.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

File name

Data type: *String*

Search string containing a partial or entire file name of an attachment file. You can use this input as a query filter to find one or more attachments that have the listed file name. If you leave this input empty, the action returns all attachments associated with the source record.

Source Record

Data type: *Document ID*

Record that you want to get attachments for. Typically this input uses a data pill from the data panel.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Attachment List

Data type: *Records*

Sys ID values of Attachment records found.

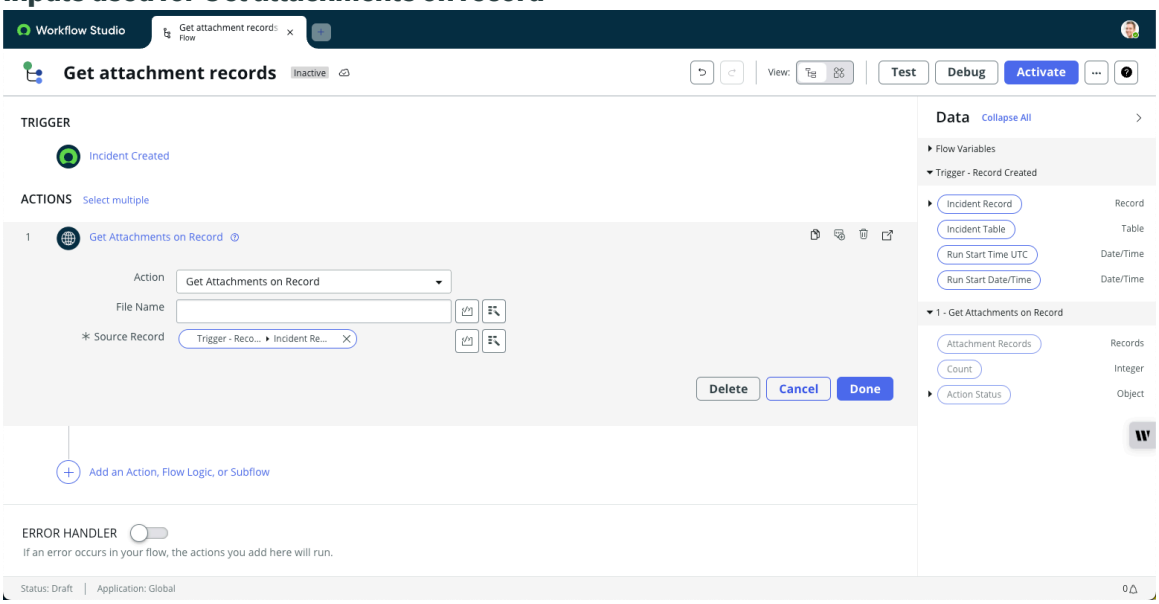
Count

Data type: *Integer*

Number of attachment records.

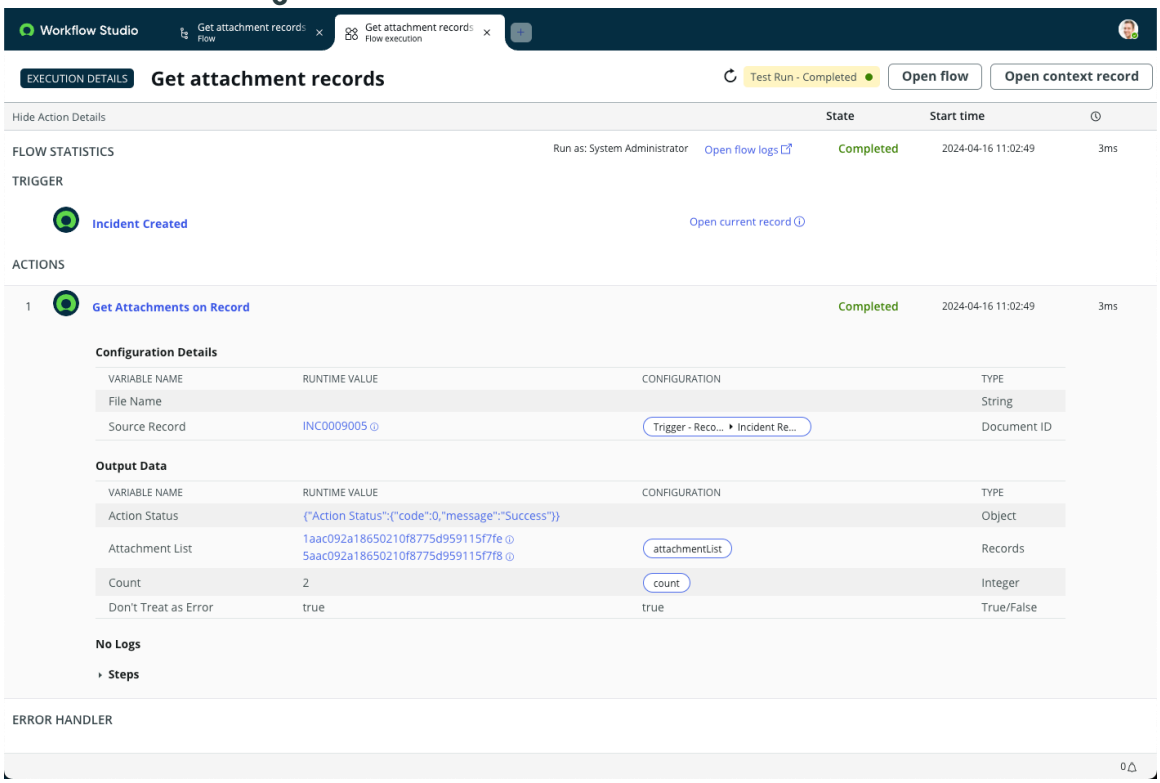
Example: Get attachments from an incident record

Inputs used for Get attachments on record



This example illustrates getting all attachments from an incident record. Leaving the File name input empty returns all attachment records.

Execution details for get attachment on record



In this example, the incident record contained two attachments.

Get Catalog Variables action

Select variables from multiple template catalog items and variable sets using the Get Catalog Variables action.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Note:
This action only returns the value of catalog variables at the time it was run. If you expect the action to recognize when catalog variable values change, you must build that logic into your flow. For example, use a Wait For Condition action with a changes operator to get catalog variables when a field value changes. If the field value changes, you must add the action in the flow again.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Submitted Request

Data type: *Record*

The requested item record from the Requested Item [sc_req_item] table for which you want to get catalog variable values.

Template Catalog Items and Variable Sets [Catalog Items and Variable Sets]

Data type: *Record*

The Catalog Item [sc_cat_item] record you want to use to populate the catalog variables slush bucket. This input does not support any data pill values.

Catalog Variables

Data type: *Slush Bucket*

The list of catalog variables whose values you want to get from the requested item record. Each catalog variable that you select is displayed as a data pill in the data pane. You can define flow-specific variables that are displayed in the Available list. To define flow-specific variables, see Create flow Service Catalog variables.

Outputs

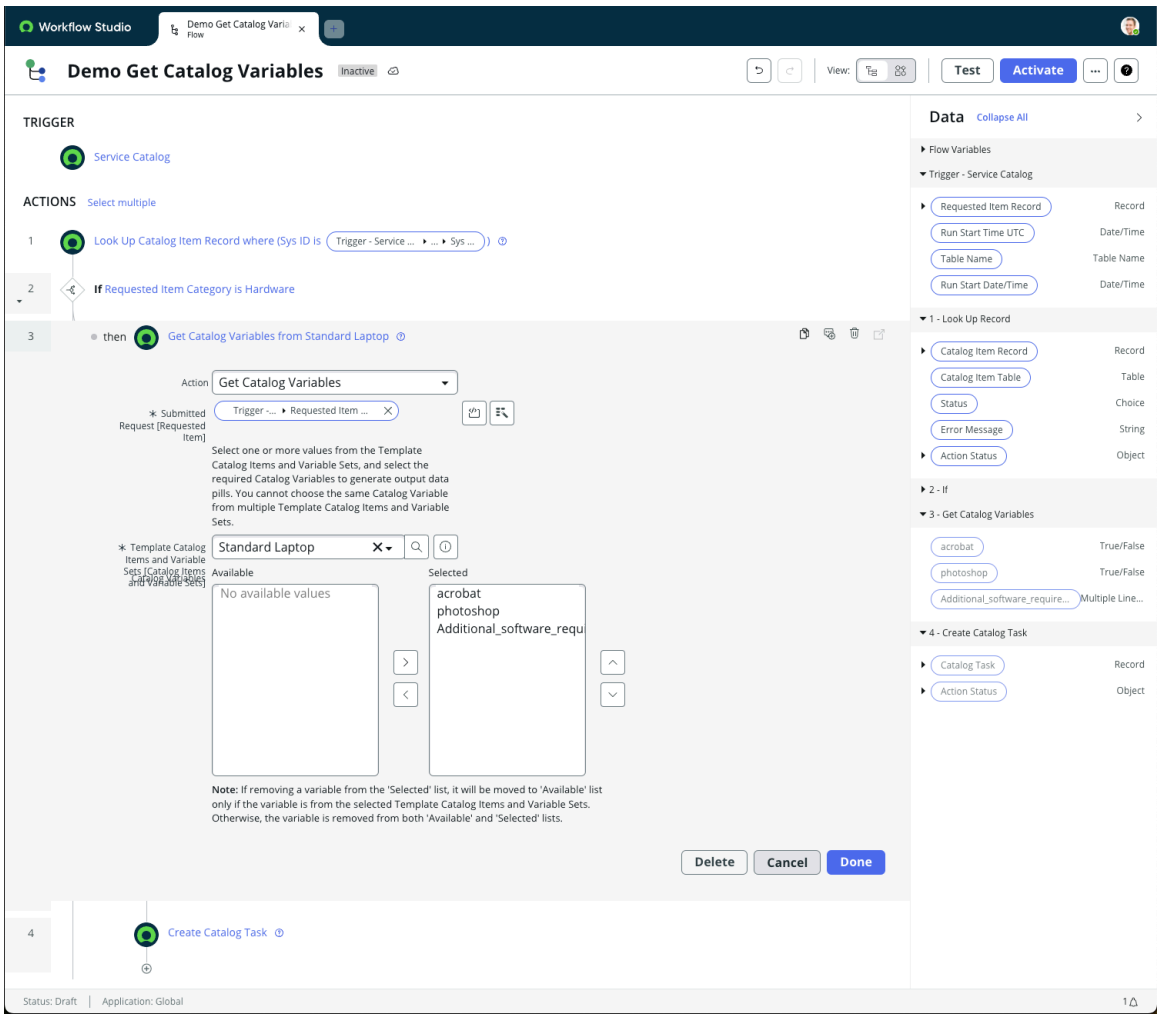
These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Catalog Variables

Data type: *Varies by catalog variable type*

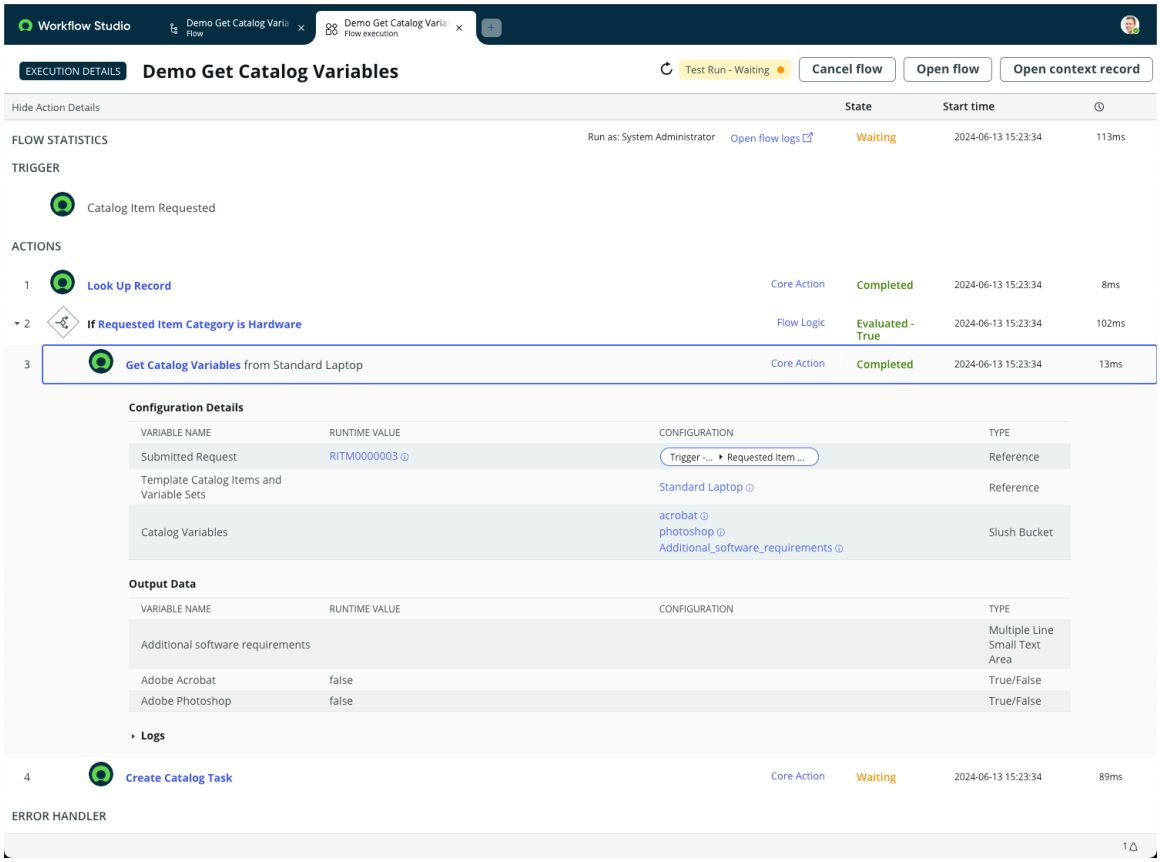
The data pane displays a separate data pill for each catalog variable selected from the Catalog Variables input. For a list of supported Service Catalog types, see Supported Service Catalog variable types.

Example: Get catalog variables for hardware items



In this example, a flow runs when someone requests an item from the Service Catalog. The first flow step looks up the Catalog item record of the requested item. The If flow logic uses the

category value (Sys ID) of the catalog item to determine if request is for an item in the Hardware category. The Get Catalog Variables action uses the trigger record as the Submitted Request input. The Template Catalog Items and Variable Sets input uses the variables provided in the Standard Laptop catalog item. All three available catalog variables are selected and displayed as data pills in the data pane.



The flow execution details show that the Submitted Request links to a request for a standard laptop. Of the three catalog variables, the string variable is empty and the two Boolean variables are false.

Get Email Header action

Access an email header value as a data pill in a flow.

Roles and availability

Available as a Workflow Studio core action. Process analysts use the flow_designer role to add an action to a flow and define configuration details.

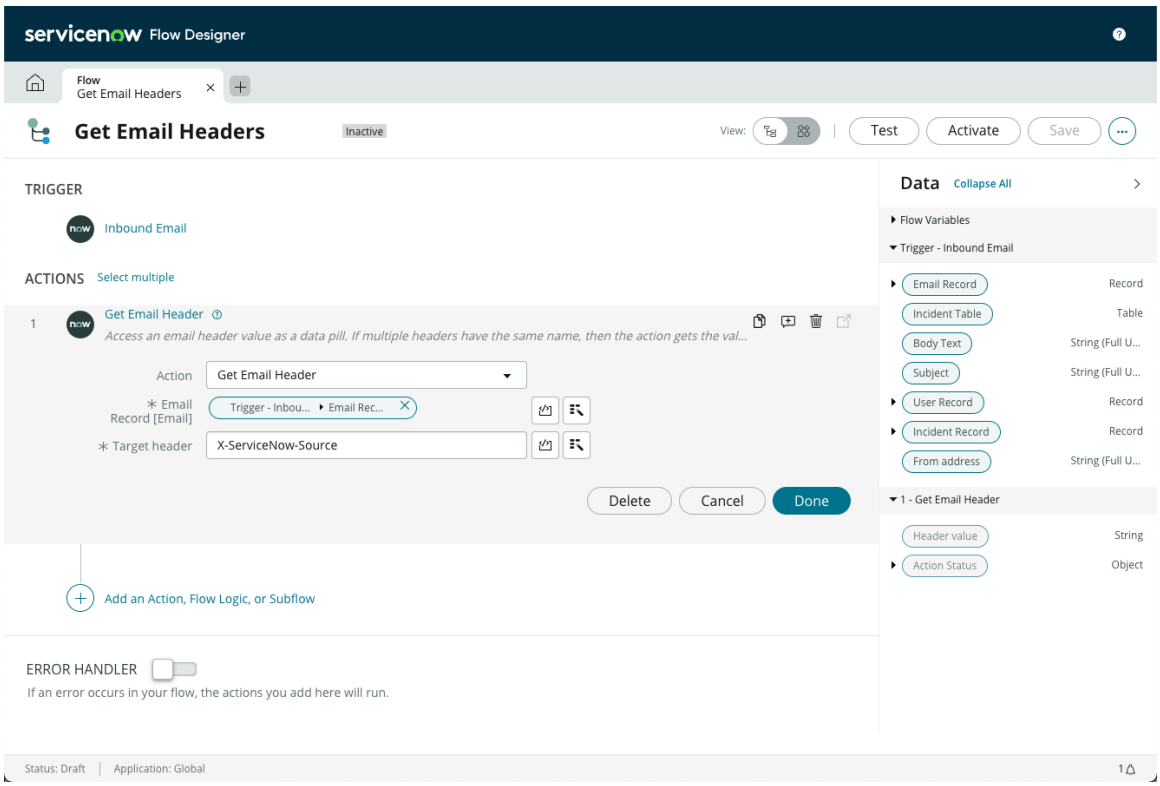
Fields

Field	Description
Email Record	Record from the Email [sys_email] table.
Target Header	Header from the email record. Upon completion of the action, the header value is added as a data pill in the flow. **Note:** If multiple headers have the same name, the action gets the value of the first header that appears.

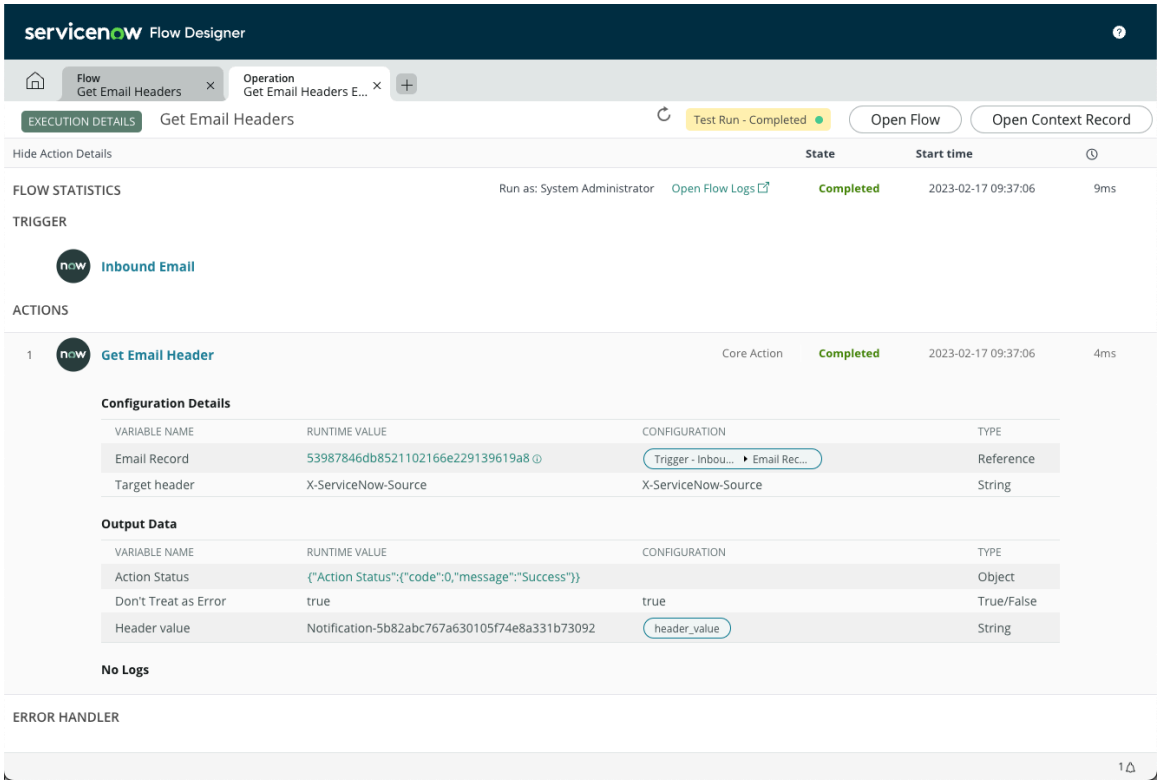
Field	Description
	This input requires a string value matching an email header name. ServiceNow provides several dedicated email headers. • X-ServiceNow-Source • X-ServiceNow-SysEmail-Version

Example

In the following example, a process owner adds the Get Email Header action under an inbound email trigger. In the **Email Record** field, the user selects to get an email header from the email that triggered the flow. In the **Target Header** field, the user selects to get the X-ServiceNow-Source header from the email.



Testing the flow with a sample email record produces the header value as a data pill.



Get Latest Response Text From Email action

Provide the most recent reply or forward message in an e-mail chain to other actions in your flow.

Roles and availability

- Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your action needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Input	Data type	Description
Email Record	Record	Email record whose most recent reply or forward message you want to provide to other actions in your flow. Select an Email [sys_email] record from the list, or add an Email [sys_email] record data pill from the Data panel.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Output	Data type	Description
Latest Response Text	String	Body text for the most recent reply or forward message in the Email [sys_email] record that you selected for the action's input. **Note:** If you select an Email [sys_email] record with a Type of New for this action's input, the Latest Response Text output will be the entire body text of the e-mail.

Log action

Logs a message in the Workflow Studio log table sys_flow_log.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

You can view the logs in the flow execution details. To learn more, see Flow execution details and Workflow Studio data.

Fields

Field	Description
Log level	Level of importance of the log message. • Error • Warn • Info
Log message	Message to display in the Flow log [sys_flow_log] table. Enter text or drag data pills into the field. **Note:** The Workflow Studio design environment only supports entering 255 characters of text for a log message. The length limitation only applies to text entered directly into the input. Data pill values can exceed 255 characters in length. You can log values greater than 255 characters long by using either a data pill value or calling the GlideSystem - log(String message, String source) method from a script.

General guidelines

Test your flow to ensure it generates useful logging information

Test your flow and review its execution details. Make sure that the log action stores useful data.

Use data pills to log dynamic data

Add one or more data pills to your Log action to store dynamically generated values. Make sure that the data pills come from actions and flow logic that have run before the Log action.

Look Up Attachment action

Looks up an attachment associated with a record and returns the Attachment Sys ID as a data pill.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

File Name

Data type: *String*

The name of the attachment that you want to look up. You can leave this input empty to look up all attachments for a record.

Source Record

Data type: *Record*

Record containing one or more attachments to look up. You can use the Look Up Record action to find an appropriate source record.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Attachment Sys ID

Data type: *String*

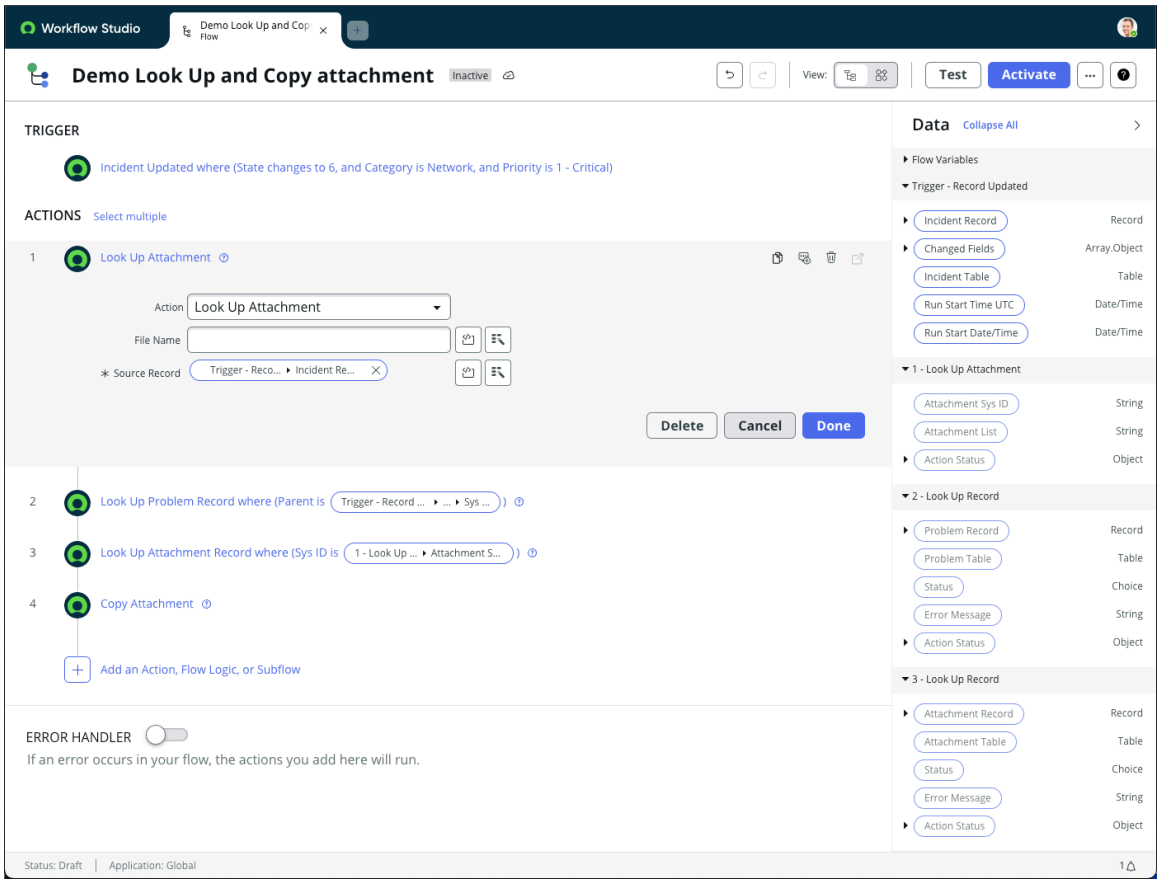
Sys ID of the first matching attachment. You can use the Look Up Record action to convert this string value into a record output.

Attachment List

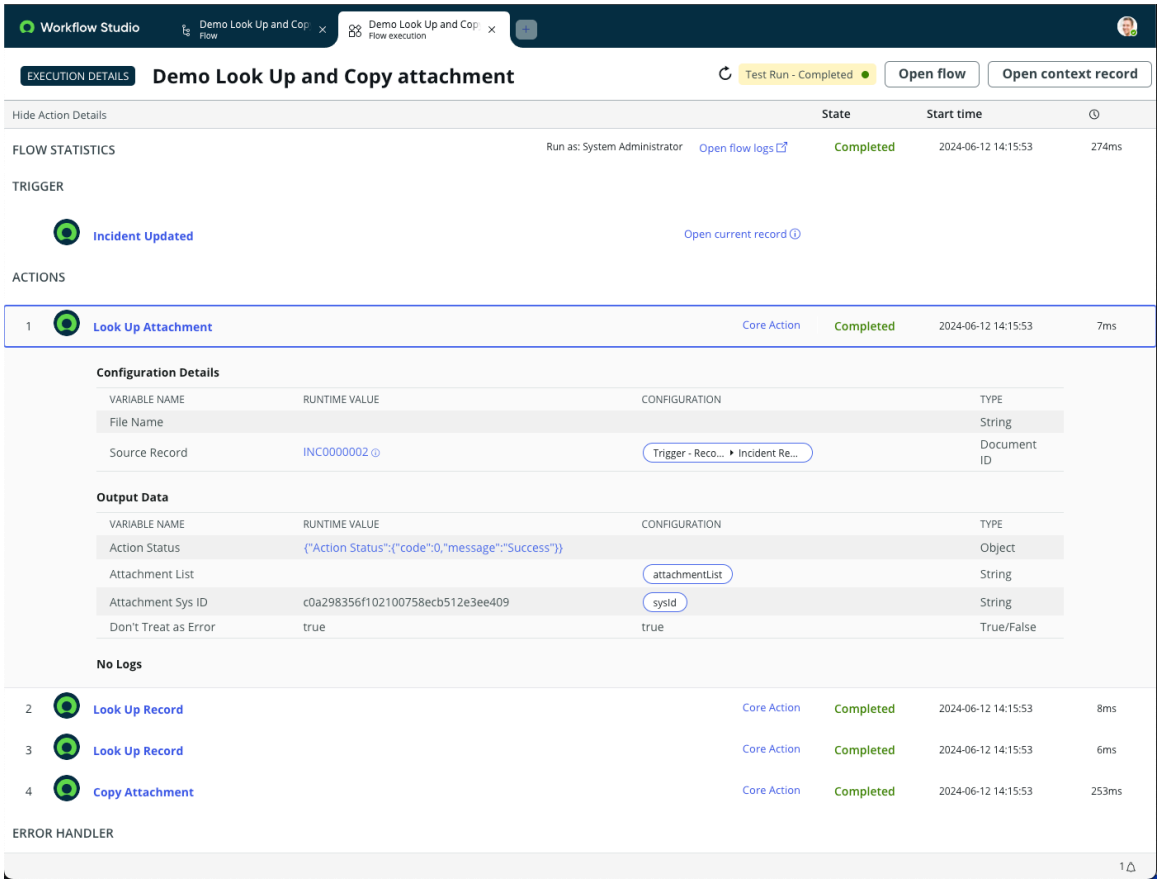
Data type: *String*

JSON formatted string containing all attachments that match the look up query. Each matching item is formatted as an object containing the Sys ID, the file name, and the file size in bytes of the attachment. You can use a custom action to parse this object as needed.

Example: Look up attachments from incident records



This example looks up attachment from incident records so that they can be copied to related problem records. The File Name input is empty to find all attachments in the source record. The Source Record input uses the incident trigger record data pill.



In this example, the flow execution details show that the trigger incident had only one attachment. The Attachment Sys ID output lists the Sys ID of the attachment record as a string value. The Attachment List output is empty because there was only one attachment associated with the source record.

When the triggering incident record has two or more attachments matching the look up query, the action populates the Attachment List output with a JSON string.

```
[
  {
    "sys_id": "a8008849db7e4210497c1a481396193c",
    "file_name": "Root Cause Analysis",
    "file_size_in_bytes": "48955"
  },
  {
    "sys_id": "c0a298356f102100758ecb512e3ee409",
    "file_name": "Pasted Image",
    "file_size_in_bytes": "97236"
  }
]
```

Look up email attachments action

Look up files that are attached to an email so that you can perform an action on the files.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Fields

Field	Description
Email record [Email]	Record from the Email [sys_email] table. For example, select the email record from the flow trigger. After you select an email record, the related Email Attachment [sys_email_attachment] record and its fields become available as data pills.

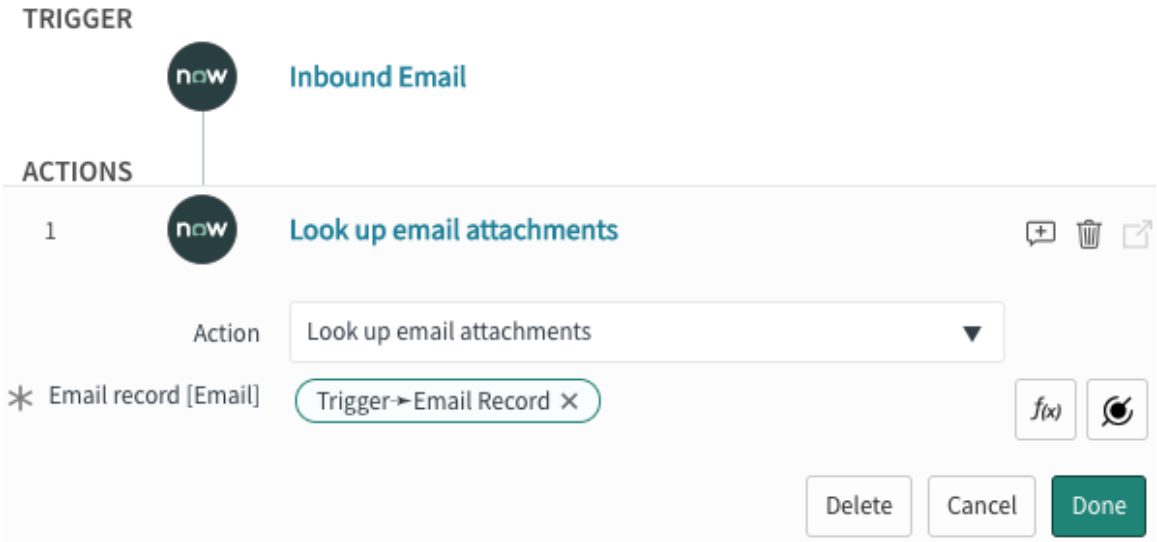
Output

This action generates a list of Email Attachment [sys_email_attachment] records, which list the attachments that are associated with a given email record. To perform an action on an attachment, add flow logic that runs for each Attachment pill under the Email Attachment Record pill. For more information, see For each flow logic.

Example

In the following example, a process owner adds the Look up email attachments action under an inbound email trigger. In the **Email record [Email]** field, the user selects to look up files that are attached to the email that triggered the flow.

Email attachments action in a flow



Look Up Record action

Look up a record from any table based on defined conditions.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your action needs. To add dynamic values, you can also drag pills from the Data panel or select them from the pill picker.

Table

Data type: *Table Name*

Table name containing the records you want to look up.

Conditions

Data type: *Conditions*

Field names and field values that you want to use to search for records. To use an inline script to specify conditions, consider using the GlideRecord and GlideQueryCondition classes to build your query. See GlideRecord - Global and GlideQueryCondition - Global.

Order by

Data type: *Field Name*

Field you want to use to sort results.

Sort Type

Data type: *Choice*

Option to sort alphabetically in ascending or descending order.

If multiple records are found

Data type: *Choice*

Option to determine what information to return when more than one record matches the defined conditions.

- Return only the first record
- Fail the step

Don't fail on error

Data type: *Boolean*

Option whether to fail the flow when the lookup can't find a record.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Record

Data type: *Record*

Record found based on the conditions you specified in the *Conditions* input.

Table

Data type: *Table*

Name of the table associated with the returned record.

Status

Data type: *Choice*

0 if a record was found successfully, and 1 if there was an error.

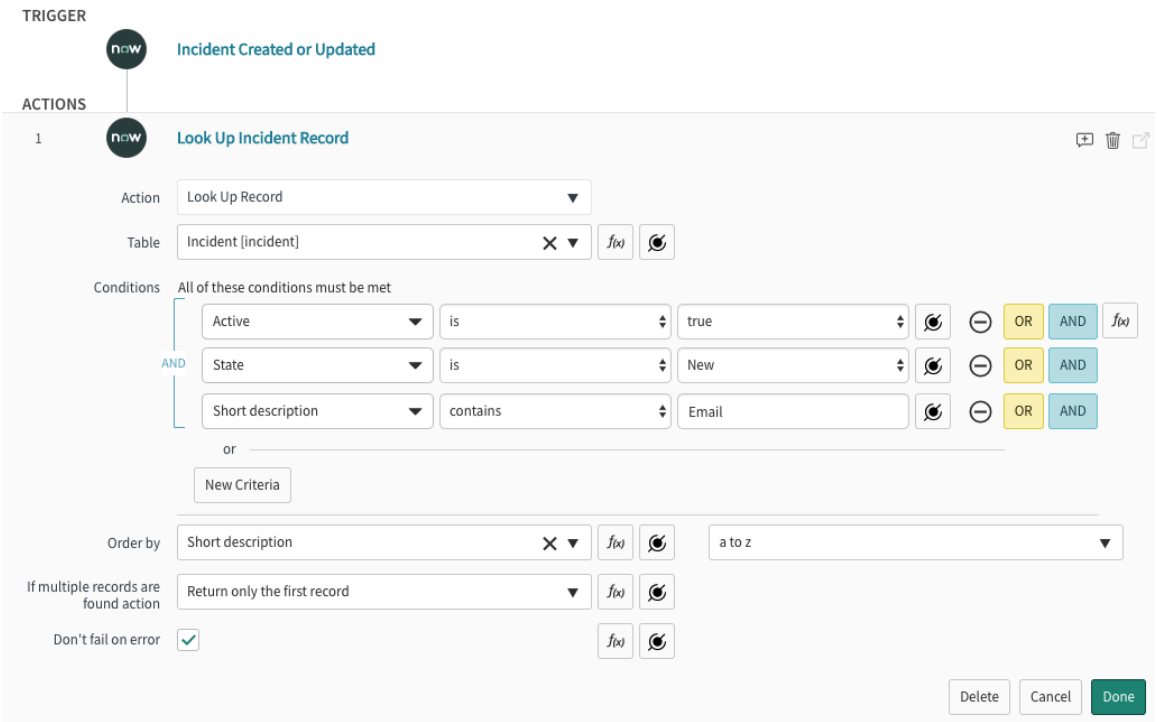
Error Message

Data type: *String*

Message containing details about why the record could not be found.

Note:
This output's value is only populated if the *Status* output's value is 1.

Example



Look Up Records action

Look up multiple records on any table using defined conditions.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag pills from the Data panel or select them from the pill picker.

Table

Data type: *`Table Name`*

Table name containing the records you want to look up.

Conditions

Data type: *`Conditions`*

Field names and field values that you want to use to search for records. To use an inline script to specify conditions, consider using the GlideRecord and GlideQueryCondition classes to build your query. See GlideRecord - Global and GlideQueryCondition - Global.

Order by

Data type: *`Field Name`*

Field you want to use to sort results.

Sort Type

Data type: *`Choice`*

Option to sort alphabetically in ascending or descending order.

Max Results

Data type: *Integer*

The maximum number of record results the action can return.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Records

Data type: *Records*

List of record Sys IDs found based on the lookup criteria that you provided. For more information, see Records.[Table] data type.

Table

Data type: *Table Name*

Table that contains the list of records.

Count

Data type: *Integer*

Number of records that the action returned.

General guidelines

Use these general guidelines when working with the Look Up Records action.

Process records with For Each flow logic

Use For Each flow logic to iterate through a list of records. For more information about using For Each flow logic, see For Each flow logic.

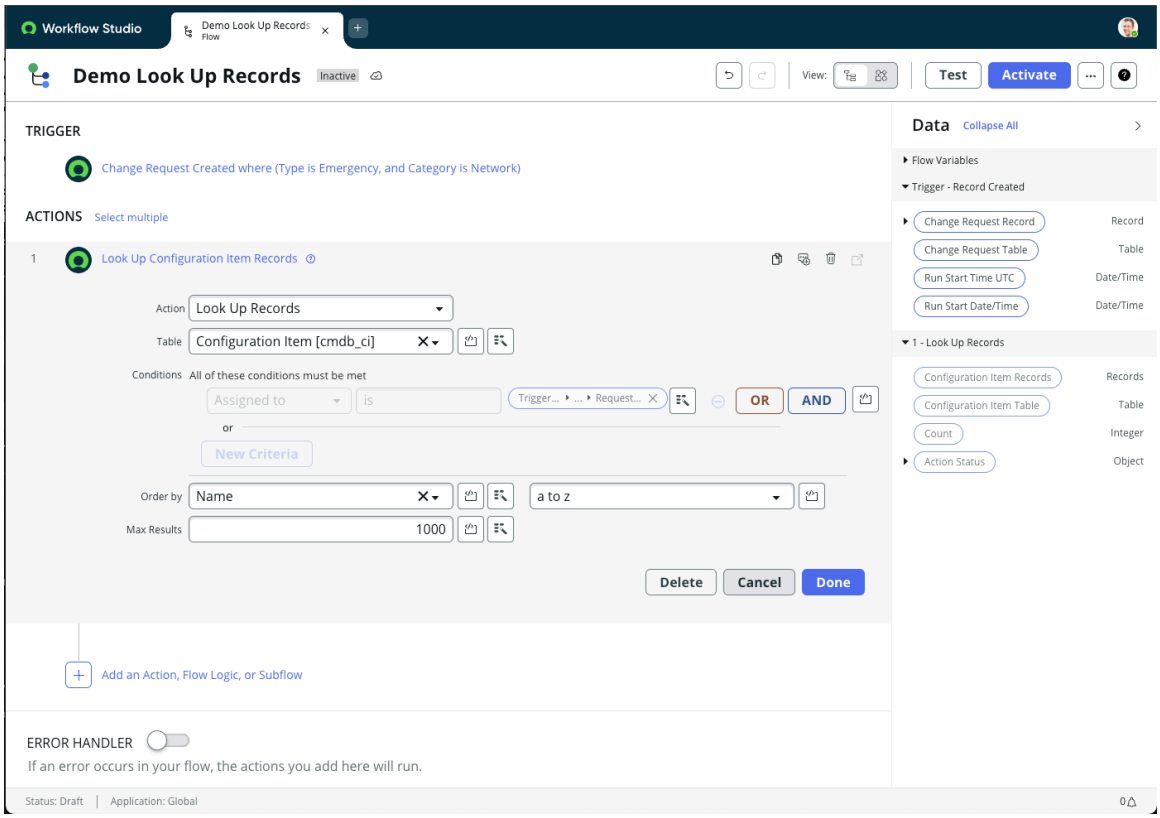
Set Max Results to improve performance

Set the Max Results input to 1000 records or lower to improve the performance of your flow. The more records that the system has to look up, the more system resources it takes to identify and process them.

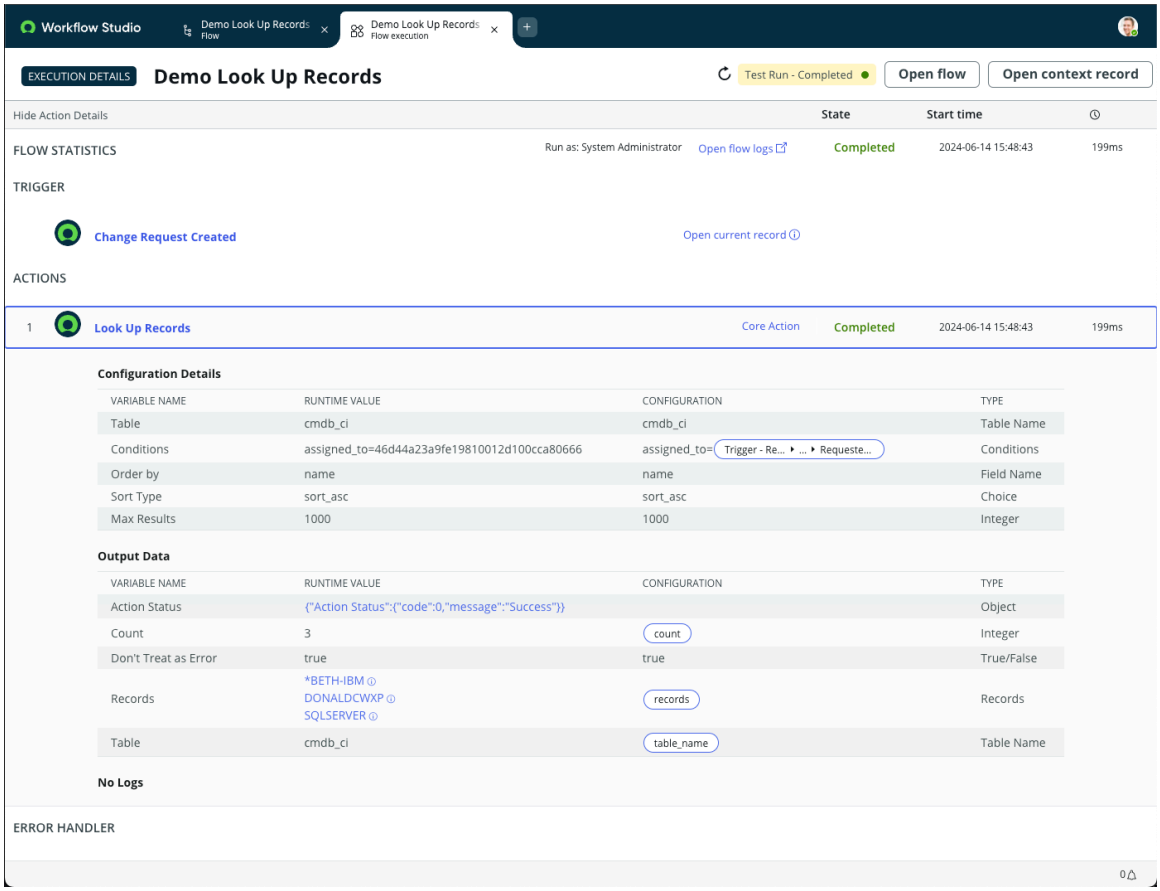
Use conditions to filter records

Use conditions to limit the number of records the action returns. The more specific conditions that you can provide, the better performance your flow has.

Example: Look up configuration items assigned to a change request user



In this example, the flow starts when an emergency change request is opened in the Network category. The Look Up Records action uses the Configuration Item [cmdb_ci] table as the Table input. The Conditions input looks for configuration items assigned to the requester of the change request. The Order by input uses the Name field to perform an ascending alphabetical type sort.



In the execution details, the Count output shows three configuration items that are assigned to the requester of the change request. The Records output shows the configuration items by name in the execution details page, but the data pill contains a series of Sys ID values. The Table output is the Configuration Item [cmdb_ci] table.

Move Attachment action

Associates a record from the Attachment [sys_attachment] table with a target record. Removes the attachment from any other associated records.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Note:
Server-side validation rules, such as data policies, business rules, and dictionary-defined mandatory fields are enforced. UI policies do not apply.

Fields

Field	Description
Source Attachment Record [Attachment]	Select an attachment record from the Attachments [sys_attachment] table.

Field	Description
Target Record	Drag a Record data pill from the data panel to attach the **Source Attachment Record** to.
Table	Automatically populates with the **Source Record** table.

Move Email Attachments to Record action

Move attachments from an email to a record so that the files are available to your users when they view the record.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Email Record

Data type: *Record*

Email [sys_mail] record containing one or more attachments that you want to move. You must enable email receiving to have valid email records. For more information about email services and setup, see Email Administration.

Target Record

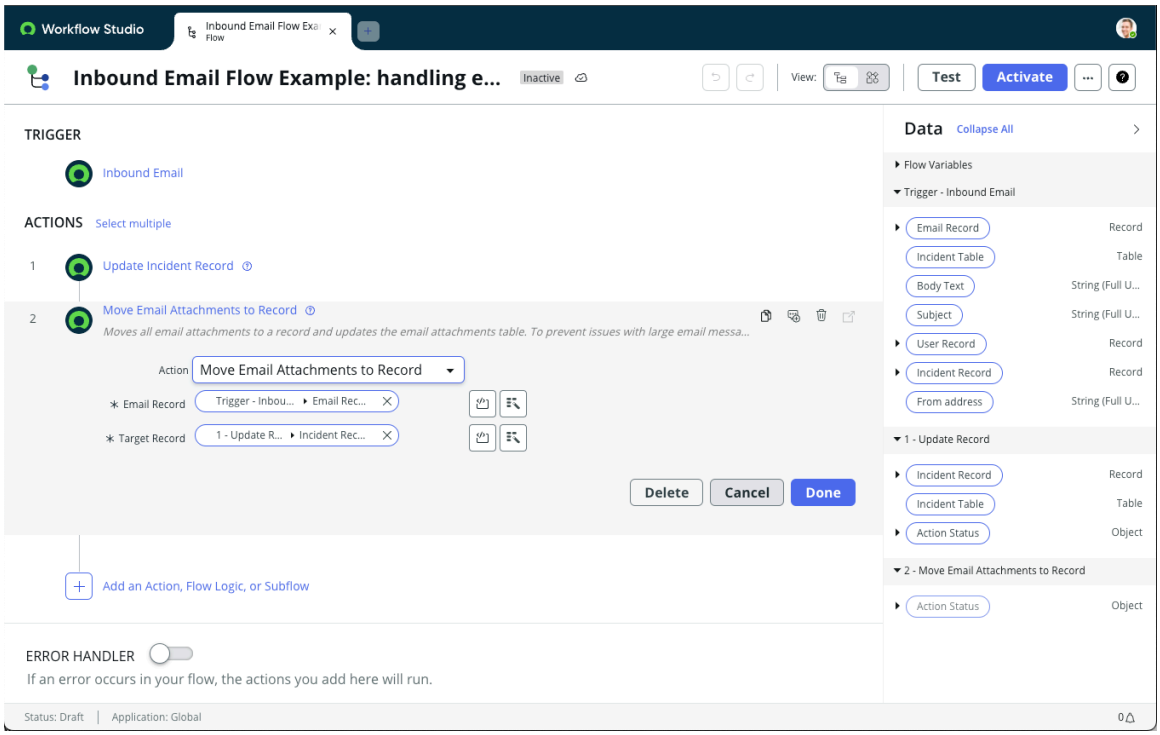
Data type: *Record*

Record to which you want to move one or more attachments. You can use the Look Up Record action to find an appropriate target record.

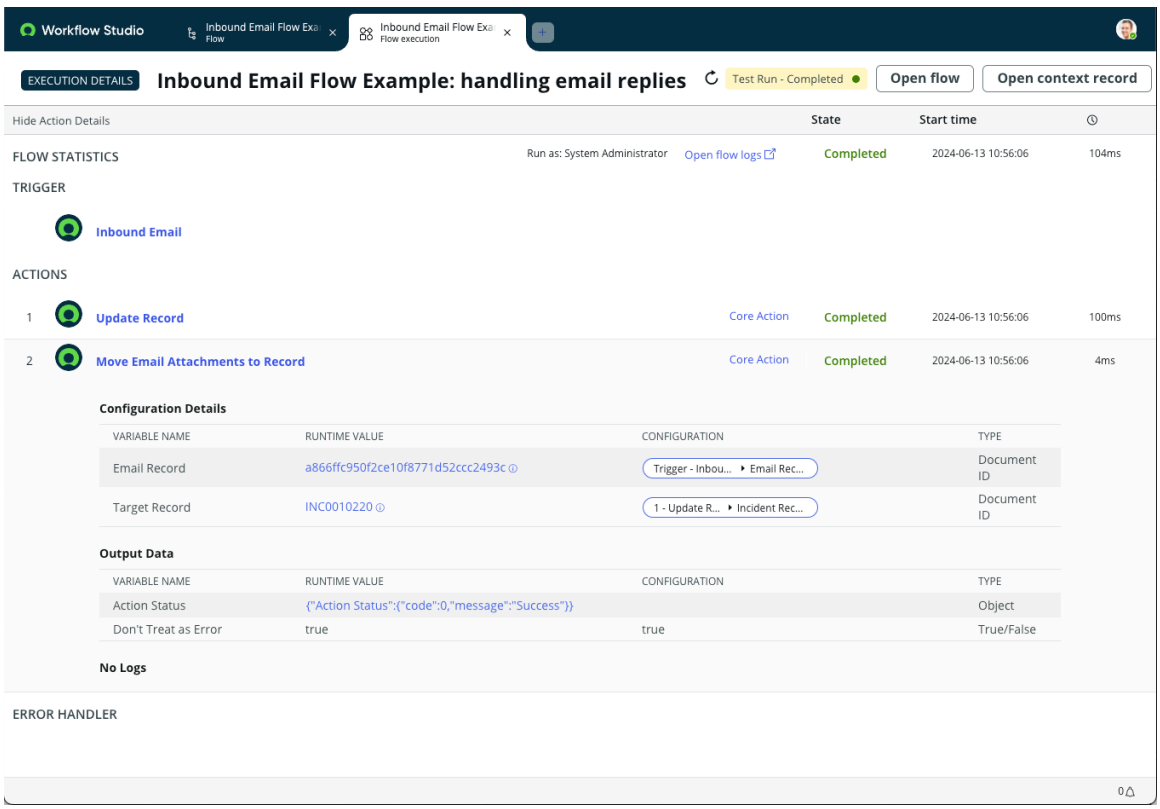
Output

This action has no outputs. Instead it updates the source Email Attachment [sys_email_attachment] record to move any attachments to the target record.

Example: Move email attachments to incident record



This example extends the sample flow for inbound email flow handling to add the Move Email Attachments to Record action. This examples assumes that you have previously set up your instance to receive emails, and that there are one or more email records with attachments. The Email Record input uses the data pill for the email that triggers the flow. The Target Record input uses the data pill for the incident record that was updated in flow step 1.



In this example, the Email Record was a received email that contained multiple attachments. The runtime value is the Sys ID reference to the received email record. The Target record was incident record INC0010220, which was updated by the email in flow step 1.

Record Producer action

Create a Task record from a Record Producer Catalog Item [sc_cat_item]. The Task record inherits values from the catalog item's variable values.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your action needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Input	Data type	Description
Catalog Item	Record	Reference to catalog item used to create a Task record.
Catalog Item Inputs	String	Catalog variables associated with the catalog item you choose. You can view associated catalog variables for Catalog Items from **Service Catalog** > **Catalog Definitions** > **Record Producers**.
Don't fail on error	True/False	Catalog variable indicating whether to fail on error.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Output	Data type	Description
Table	Table Name	Table where Task record was created.
Record	Record	Reference to Task record created.
Error Message	String	Error message produced when the record operation fails.
Status	Choice	Completion status of the action. The flow execution details page displays one of these numeric values.

Output	Data type	Description
		• Success [0]: The action succeeded. • Error [1]: The action produced an error.

Error messages

If an error occurs with this action, the following error messages appear in the execution details page.

Error message	Description
Record generation failed, check logs to get more information.	This error typically occurs when your flow or action contains logic to insert a record or records into other tables. When this logic exists, the Record Producer action aborts inserting the record into the table.

Design considerations

Follow these design considerations when creating flows that contain Record Producer actions.

Escape quotation marks from string data pills with the String Replace transform function

Complex string variables are converted into JSON format when stored in the system. To prevent any JSON formatting errors, you can use a Replace String transform function to escape the quotation marks present in any string data pills you use for catalog variables. See String transform functions.

Use transform functions to validate data pills

Whenever you use a data pill to provide data for an action input, you can use a transform function to validate the data. See Transform functions for a list of available transform functions.

Send Email action

Send an email to specified users or groups as an action in a flow.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Target Record

Data type: *Document ID*

Record that the email is associated to. When a user sends a reply to your email, the target record is updated with the reply email content.

Table

Data type: *Table Name*

Table containing the target record.

To

Data type: *String*

The main recipients of the email. Enter a list of user email addresses separated by commas or white spaces. You can also drag data pills that contain email addresses into the field, such as a User or Group record. For example, if you want to send an email to the group assigned to the incident, drag the **[Assignment group]** data pill from the data panel.

To send email to a group, you must provide a **Group email** address. To send email to group members, the group must have the **Include members** option enabled.

Note:
The number of email recipients must be equal to or less than the maximum number set by the glide.email.smtp.max_recipients system property. See Minimize SMTP Recipient Quantity [Updated in Security Center 1.3] for information about setting this value.

CC

Data type: *String*

Additional recipients copied on this email. Enter a list of user email addresses separated by commas or white spaces. You can also drag data pills that contain email addresses into the field.

BCC

Data type: *String*

Additional recipients of this email, who are visible only to the sender (blind copied). Enter a list of user email addresses separated by commas or white spaces. You can also drag data pills that contain email addresses into the field.

Subject

Data type: *String*

Subject of the email. You can enter text or drag data pills into the field.

Body

The content of the message body. You can enter text or drag data pills into the field. The editor formatting options add inline styles. There is no style sheet associated with an email body. You can add your own inline style sheet to this HTML input.

Note:
You can partially reuse content from the Email Template [sysevent_email_template] table with some manual editing. The Send Email action can't process Email template variables, which are strings that start with the dollar sign character and include content in curly braces. You must replace email variables with references to data pills. For example, replace the string `${number}` with a data pill reference to an incident record number.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your action or flow.

Email

Data type: *Record*

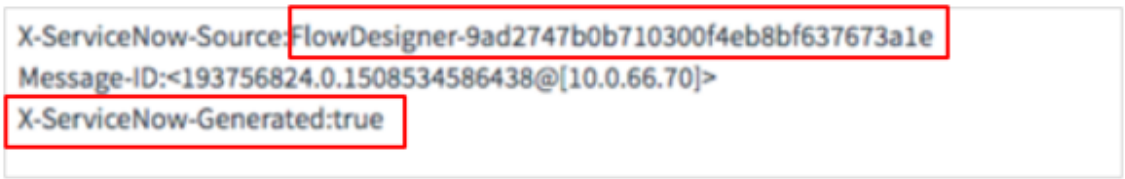
Email record created.

Configuring instance Email

For information about configuring your instance to send and receive email messages, see Configure email administration.

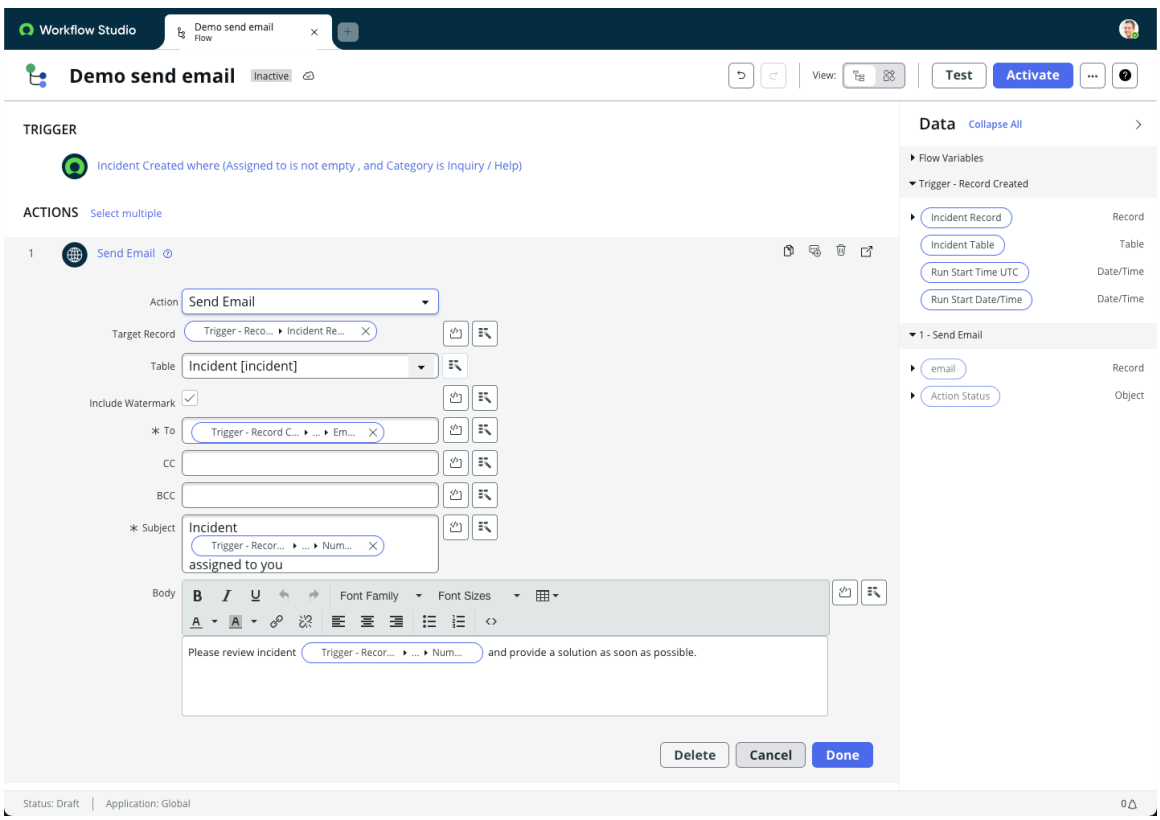
Testing the email action

To verify that the email was generated when testing the action, review the email record in the Email [sys_email] table. The **Headers** field indicates whether the email was successfully generated. For



example:
ACL restrictions apply to the Send Email action. If you configured your flow to run as the user who initiates the session, ensure that the user can access email. To test access controls for a Send Email action, impersonate a typical email sender and manually trigger the flow.

Example: Send email when incident created



In this example, the system sends an email whenever an incident record is created where the Assigned to filed is not empty and when the incident category is inquiry/help. The incident trigger record provides the values used by the send email action. For example, the email is sent

to the email address of the assigned to user, and the email subject and email body both refer to the incident number.

Send Notification action

Send an email notification to predefined recipients with predefined content. Select or create a Notification [sysevent_email_action] record to configure the email notification.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Record

Data type: *Record*

Reference to record to send notification [sysevent_email_action] about.

Table

Data type: *Table Name*

Table of the record to send a notification about. This input is automatically populated when you select a record.

Notification

Data type: *Record*

Reference to the Notification [sysevent_email_action] to send. If there is no associated Notification for the type of record you select, you will need to create one from **All** > **System Notification** > **Email** > **Notifications**.

Outputs

This action has no outputs, although it does send the notification specified by the input values.

General guidelines

Use these general guidelines when using the Send Notification action.

Select the Send Notification action to send an email notification

Select this action when you want to send an email notification to predefined recipients containing predefined content. For information about creating an email notification, see Create an email notification.

Use a Integration Hub spoke action for other types of notifications

Use an Integration Hub spoke action to send a notification through an external service such as Microsoft Teams. For a list of available integration spokes and their actions, see Integration Hub available spokes.

Send SMS action

Send an email-based SMS text message to specified users or groups using your instance email server. Recipients must have an SMS device configured to receive the message.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Deprecation of email-to-text gateway

Email-to-text service is now deprecated and no longer supported by service providers such as AT&T and Verizon. It is recommended to use Notify for sending SMS notifications.

Notify provides support for sending SMS notifications through integration of third-party services, such as Twilio, and requires configuration of supporting workflows. For more information, see How to setup a SMS email notification in ServiceNow [KB0712569] knowledge base article.

Fields

Field	Description
Recipients	Recipients of the SMS text message. Specify a user or group by dragging a pill from the data panel. **Note:** Hard-coded addresses are not supported.
Message	Content of the SMS text message. You can enter text directly or drag a pill from the data panel.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Recipients

Data type: *String*

Recipients of the SMS text message. Specify a user or group by dragging a pill from the data panel.

Message

Data type: *String*

Content of the SMS text message. You can enter text directly or drag a pill from the data panel.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Email

Data type: *Record*

The Email record created for the SMS message.

Error messages

Error	Description
`Recipients are required to send SMS`	Appears when no user or group is entered in the recipients field.
`Message is required to send SMS`	Appears when no message is entered in the message field.
`While parsing the recipients, found hardcoded addresses: %s which were ignored as they are not supported`	Appears when hard-coded address is entered in the recipients field.
`While parsing the recipients, no active users or groups were found for sys_id(s): %s`	Appears when one or more inactive users or groups are entered in the recipients field.
`No active SMS device was found for users(s) with sys_id(s): %s`	Appears when no SMS device is configured and found for one or more recipients.
`Email sending is currently disabled. Please enable it via the property to send SMS using email gateway`	Appears when email sending is disabled.

SLA Percentage Timer action

Identify when a task SLA record reaches a specific percentage value and perform other actions or flow logic that is based on the SLA percentage. For example, send a notification when an SLA percentage timer completes.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Fields

Field	Description
Percentage	The positive integer percentage of the total SLA duration used to compute an end time. For example, a 50% percentage results in the system computing an end date-time value that is 50% of the total SLA duration. If an SLA requires tasks to be completed within 24-hours, then 50% of that SLA would be 12 hours.

Scheduled End Date/Time

The **Scheduled End Date/Time** output data pill lists the computed time that the SLA percentage timer action is expected to end. The computed end date is determined by the input Task SLA record and the input Percentage. This date/time value is independent of any elapsed time field values in the Task SLA record.

- If the end date is in the future, the system creates a system event to continue running the action at that future date. While the system waits for the scheduled end date, it pauses the flow and action.
- If the end date is in the past, the system immediately sets the **Status** of the SLA Percentage Timer action.

Status

The **Status** data pill contains the result of the SLA percentage timer.

SLA Percentage Timer status descriptions

Status	Description
Completed	The timer action reached its scheduled end date/time. Flow designers can build specific flow logic for this action status.
Paused	The timer was paused before its scheduled end date/time. If the timer resumes running, Workflow Studio generates a new scheduled end date/time value. Flow designers can build specific flow logic for this action status.
Repair	The flow is running in repair mode, and the scheduled end date/time is in the past. Flow designers can build specific flow logic for this action status.
Skipped	The timer did not run because the scheduled end date/time is in the past. Flow designers can build specific flow logic for this action status.
Waiting	The timer is running and has yet to reach the scheduled end date/time.

Workflow Studio sets the action status when the SLA state matches an SLA condition or when certain UI actions are selected.

Action status set for SLA state

SLA state	Action status set	Flow run state
SLA attaches and the scheduled end date/time is in the future.	Set action status to **Waiting**.	The flow waits until the SLA timer completes, is cancelled, or is paused.
SLA attaches and the scheduled end date/time is in the past.	• If the flow was started in Repair mode, set action status to **Repair**. • Otherwise, set action status to **Skipped**.	The flow runs the next action or flow logic in the flow sequence.
SLA Cancels.	Set action status to a null value.	The flow stops with a state of **Cancelled**.
SLA Pauses.	Set action status to **Paused**.	The flow waits until the SLA Task flow is cancelled or is resumed.
SLA reaches Scheduled End Date/Time.	Set action status to **Completed**.	The flow runs the next action or flow logic in the flow sequence.

Action status set for SLA state (continued)

SLA state	Action status set	Flow run state
SLA Resumes.	Set action status to **Waiting**.	The flow waits until the SLA timer completes, is cancelled, or is paused.
SLA Stops.	Set action status to a null value.	The flow stops with a state of **Cancelled**.

Total Duration

The **Total Duration** data pill lists the total number of seconds that the action ran. The total duration is computed from the action start time and the time when the action reached the **Completed** status. Status values other than **Completed** produce a null value **Total Duration**.

General guidelines

Follow these general guidelines when creating flows that contain Service Level Agreement (SLA) Percentage Timer actions.

Add SLA Percentage Timer actions only to flows with an SLA Task trigger

An SLA Percentage Timer action can only run when the flow starts from an SLA Task trigger. You cannot activate a subflow containing an SLA Percentage Timer action.

Create conditional flow logic for expected Status values

Use the value of the **Status** field as a condition for flow logic. Build flow logic for expected **Status** values such as **Completed**, **Repair**, and **Skipped**. For example, add an **If** flow logic block to send a notification when the SLA Percentage Timer has a status of **Completed**.

Assign each SLA Percentage Timer action a unique cumulative Wait for percentage value

Each SLA Percentage Timer action computes its own Scheduled End Date/Time using its Wait for percentage value. If you create multiple SLA Percentage Timer actions, give each action its own unique cumulative Wait for percentage value. For example, create three separate actions with different percentage complete values such as 25%, 50%, and 75% complete. Setting all three actions to the same percentage complete value such as 25% causes the timers to complete at the same time.

Copy existing flows to make customizations

Reduce development time by copying the default SLA flows and customizing the copies with your own logic. Select a customized flow to run from the SLA definition. See Create an SLA definition .

Submit Catalog Item Request action

Create a requested item [sc_req_item] on a Service Catalog Request [sc_request].

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

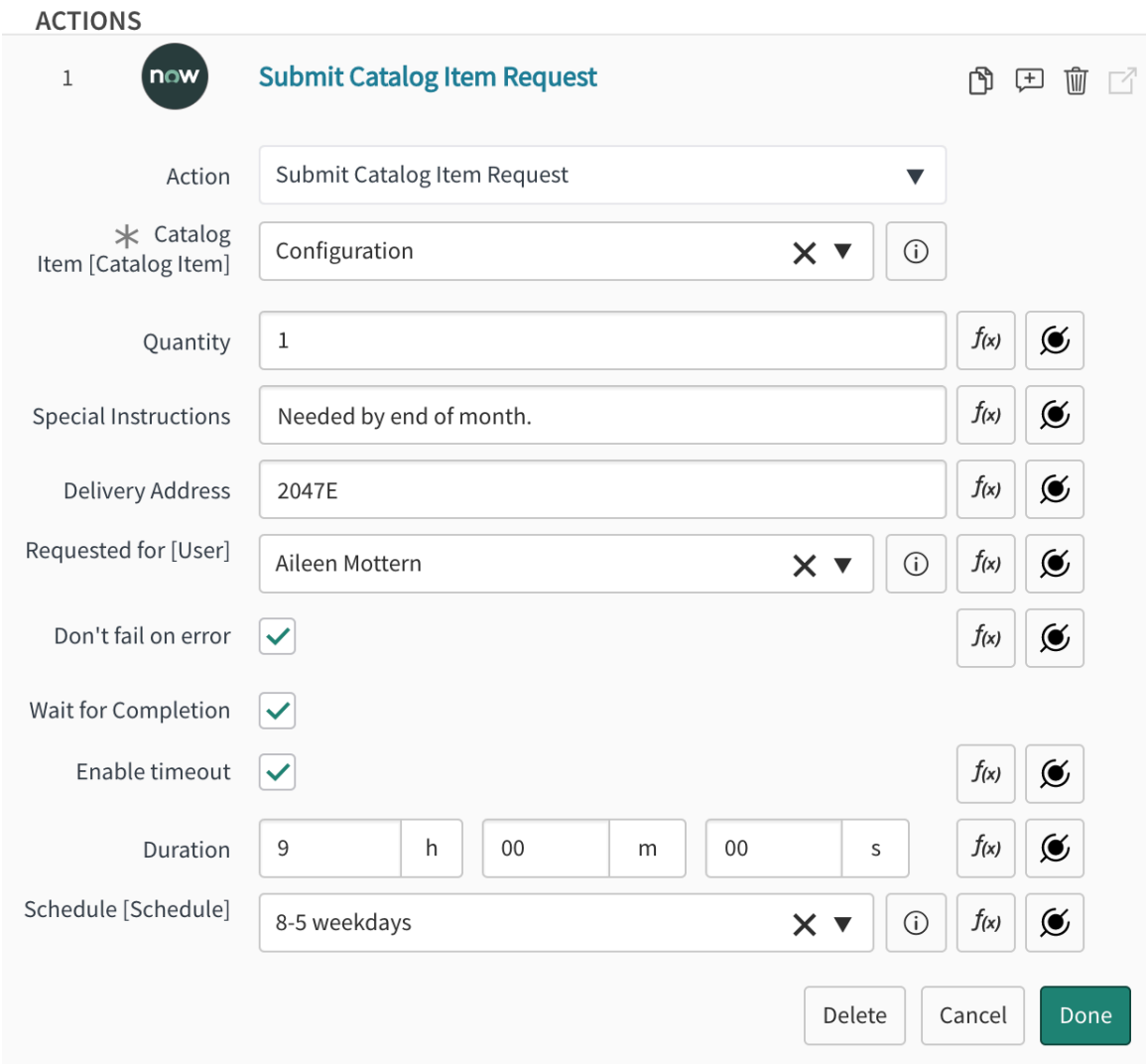
Input	Description
Catalog Item	Name of the requested catalog item. Extra inputs may be added dynamically, depending on which catalog item is selected. For example, when the requested catalog item is a new email account, a field for **Preferred Email address** appears. **Note:** The following Service Catalog variable types are not supported. • list collector • lookup multiple choice • lookup select box
Quantity	Number of items requested.
Special Instructions	Text describing any special instructions about the item request.
Delivery Address	Location where the requested item should be delivered.
Requested for	User that the item is requested for.
Don't fail on error	Option to determine whether to fail the flow if the action produces an error.
Wait for Completion	Option to force the flow to wait until the action has been completed before continuing.
Enable timeout	Option to limit the amount of time that the flow waits for the action to be completed before continuing. **Note:** Use the **Enable timeout** option to prevent this action from continuing to run. If the condition to continue is never met, a timeout value specifies when the system skips the Wait for Condition action and go to the next item in the flow. You must set a Duration value to enable a timeout. You can also select a Schedule if you want to compute the duration end date based on a specific work schedule. This field appears only when the **Wait for Completion** option is selected.
Duration	Amount of time that the flow waits before continuing when the **Enable timeout** option is selected. Enter the time to wait in hours, minutes, and seconds. If you leave this field empty, the flow does not wait. This field appears only when the **Wait for Completion** option is selected.
Schedule	Schedule used to compute the timeout duration when the **Enable timeout** option is selected. For example, waiting for 10 hours as part of an 8-5 weekdays

Input	Description
	schedule causes the flow to wait for one or more business days. If you leave this field empty, the timeout runs without a schedule. This field appears only when the **Wait for Completion** option is selected.

Output

Field	Description	Data Type
Error Message	Message that displays if the action produces an error.	String
Requested Item	Document ID for the requested item.	Document ID
Status	The completion status of the action as a numeric value. • 0 (success) • 1 (error) • 2 (timeout)	Choice

Example



Design considerations

Follow these design considerations when creating flows that contain Submit a Catalog Item Request actions.

Escape quotation marks from string data pills with the String Replace transform function

Complex string variables are converted into JSON format when stored in the system. To prevent any JSON formatting errors, you can use a Replace String transform function to escape the quotation marks present in any string data pills you use for catalog variables. See String transform functions.

Use transform functions to validate data pills

Whenever you use a data pill to provide data for an action input, you can use a transform function to validate the data. See Transform functions for a list of available transform functions.

Update Multiple Records action

Look up and update multiple records as a single action. Using this action removes the need to separately look up a list of records and then process the list with For Each flow logic. Set field values with a template or add and configure them using data pills.

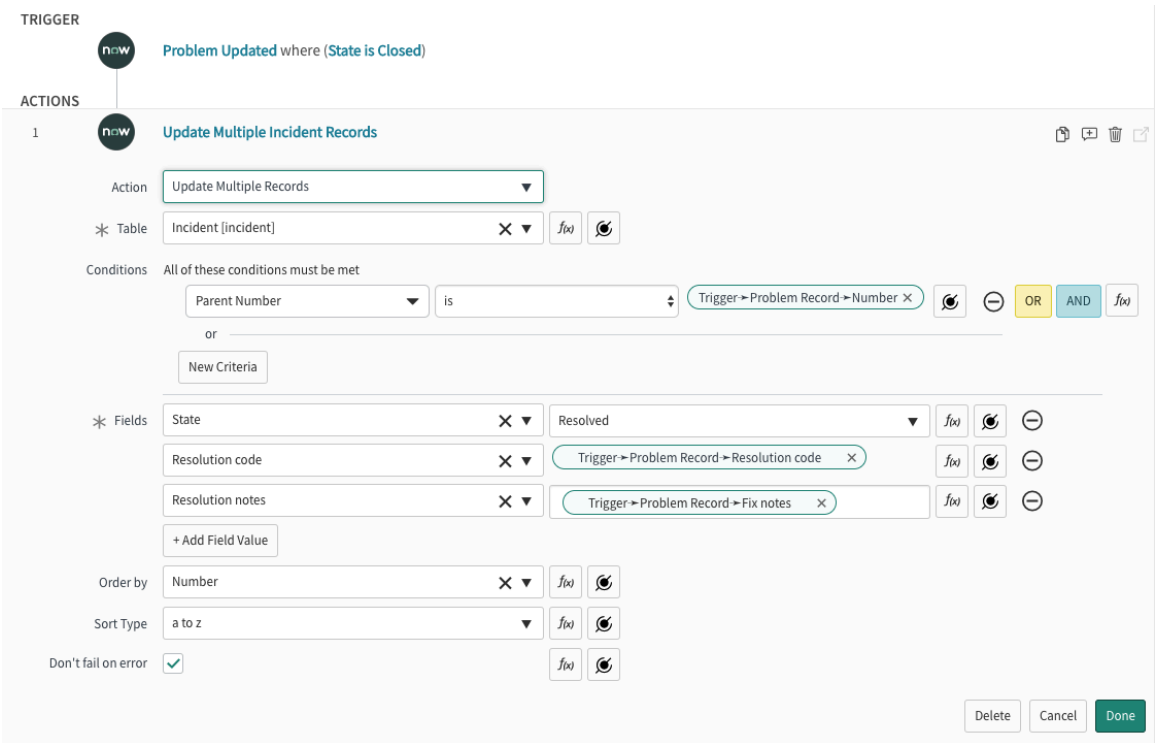
Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Fields

Field	Description
Table	Select the table containing the records to look up and update.
Conditions	Define the filter conditions used to look up records.
Field Values	Set static or dynamic values of fields in the record. For example, to set the short description to a static value, select **Short description** and set the desired value. To add dynamic values, see Create a template value input. **Important:** The system does not support updating multiple journal fields such as the additional comments or work notes of a task record.
Order by	Select the field you want to use to sort the records when more than one record matches the defined conditions.
Sort Type	Determine whether to sort the records alphabetically in ascending or descending order.
Don't fail on error	Specify whether to continue running the flow when there is an error.

Example



Outputs

Field	Description	Data Type
Count	Number of records updated. If no records are updated, the count is 0.	Integer
Error Message	Message that displays if the action produces an error.	String
Status	The completion status of the action as a numeric value. • 0 (success) • 1 (error)	Choice

Update Record action

Update an existing record in a table. You can dynamically add and configure fields for the record.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag pills from the Data panel or select them from the pill picker.

Record

Data type: *Record*

The record to be updated. Drag-and-drop a record data pill or use the data pill picker to select a record.

⚠ Warning:
When using script to select a record, always add a condition to check for a matching record. Some GlideRecord methods return a list of records whenever the method query produces no results. Without a condition check, the action may update all records in a table. For example, this script uses an If condition to verify that a record exists. If the record exists, it returns a Sys ID value. If no record exists, it returns a null result.

```
var configurationItem = new GlideRecord('cmdb_ci');
if
 (configurationItem.get(fd_data.trigger.cmdb_ci.sys_
id))
return configurationItem;
else
return null;
```

Table

Data type: *Table Name*

The table associated with the record to update.

Field Values

Data type: *Template Value*

The values of fields in the record to be updated. Select **Add field value** to display options to select a field and set a value. For example, to set the short description to a certain value, select **Short description** and set the desired value.

If adding the action to a subflow, you can Create a template value input.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Record

Data type: *Record*

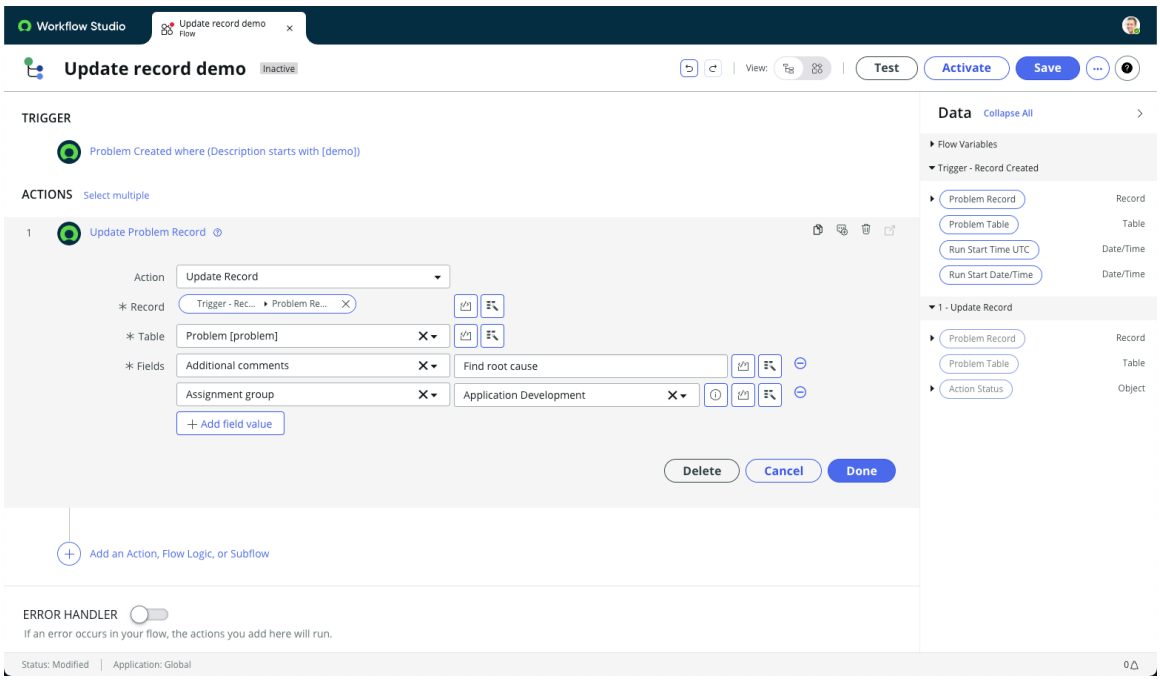
The record that was updated.

Table Name

Data type: *Table Name*

The table associated with the updated record.

Example: Update trigger record



In this example, the Update Record action updates values provided by the trigger problem record. The action updates both the Additional comments and Assignment group fields of a problem record.

Wait For Condition action

Pause a flow until record values match a specific set of conditions.

Roles and availability

Available as a Workflow Studio ServiceNow core action. Users with the flow_designer or admin role can add an action to a flow and define configuration details.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Record

Data type: *Record*

The record whose field values the action monitors for changes. This record can come from a flow trigger, a subflow input, or the output of a previous flow step.

> **Note:**
> If this record is deleted, the flow stops waiting and continues running.

Table

Data type: *Table Name*

The table associated with the record to monitor. This read-only value is set to the table associated with the record you selected. Confirm that the system supports Wait for Condition for your selected table. For a list of unsupported tables, see the Unsupported tables section.

Conditions

Data type: *Conditions*

The record values necessary to resume running the flow. For example, if the condition is **[State] [is] [Closed]**, the flow pauses until the condition is met. Once met, the flow moves on to the next flow step. Only select conditions that apply to field values in the table to which the record belongs. See Condition evaluation for more information about creating valid conditions.

> **Note:**
> For conditions that depend on a specific duration, consider using Wait for a duration flow logic instead.

Enable Timeout

Data type: *True/False*

Option to limit the amount of time that the flow waits for the action to be completed before continuing.

> **Note:**
> Use the **Enable timeout** option to prevent this action from continuing to run. If the condition to continue is never met, a timeout value specifies when the system skips the Wait for Condition action and goes to the next item in the flow. You must set a Duration value to enable a timeout. You can also select a Schedule if you want to compute the duration end date based on a specific work schedule.

Duration

Data type: *Duration*

Amount of time that the flow waits before continuing when the **Enable timeout** option is selected. Enter the time to wait in hours, minutes, and seconds. If you leave this field empty, the flow does not wait.

Schedule

Data type: *Choice*

Schedule used to compute the timeout duration when the **Enable timeout** option is selected. For example, waiting for 10 hours as part of an 8-5 weekdays schedule causes the flow to wait for one or more business days. If you leave this field empty, the timeout runs without a schedule.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

State

Data type: *Choice*

The completion status of the action as a numeric value.

- 0 (success)
- 1 (error)

Condition evaluation

The Wait for Condition action only evaluates the wait condition when there are changes to fields in the **Record** you select. A valid wait condition meets these criteria.

- Each condition evaluates a field from the table to which the record belongs.

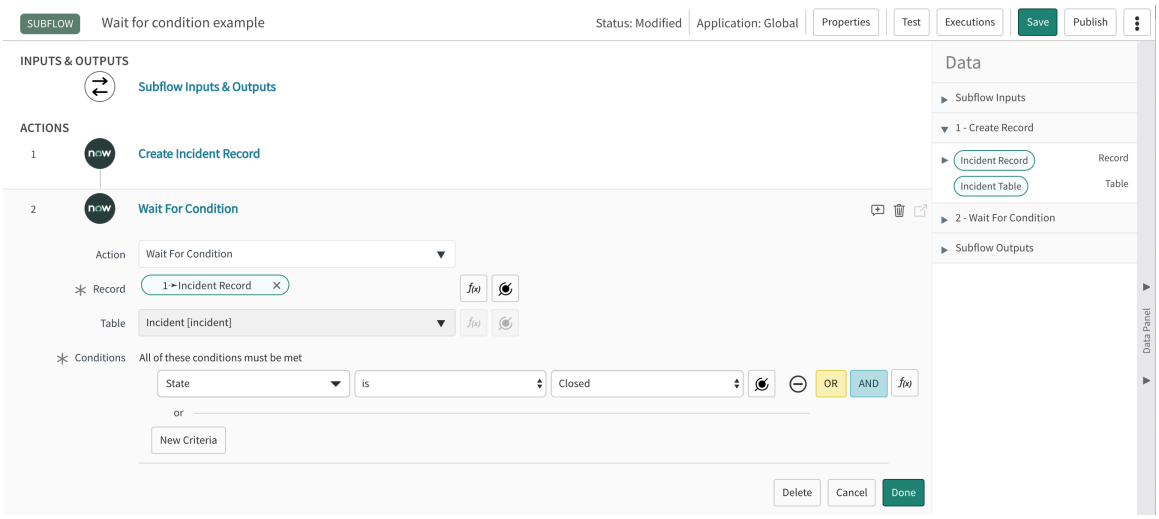
> **Note:**
> Don't create conditions that dot-walk to another table or depend upon catalog item variables from a related record.

- Each condition specifies a field value change rather than a relative time period.

> **Note:**
> For conditions that depend on a specific duration, consider using Wait for a duration flow logic instead.

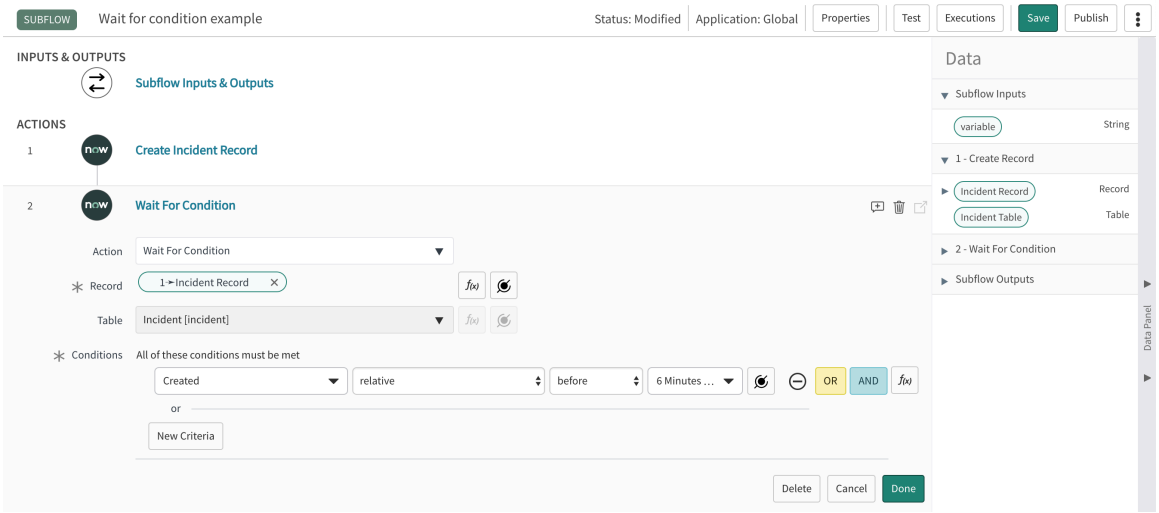
Valid wait condition

In this example, the wait condition is valid because the **State** field belongs to the Incident table and the condition is a field value change to **Closed**.

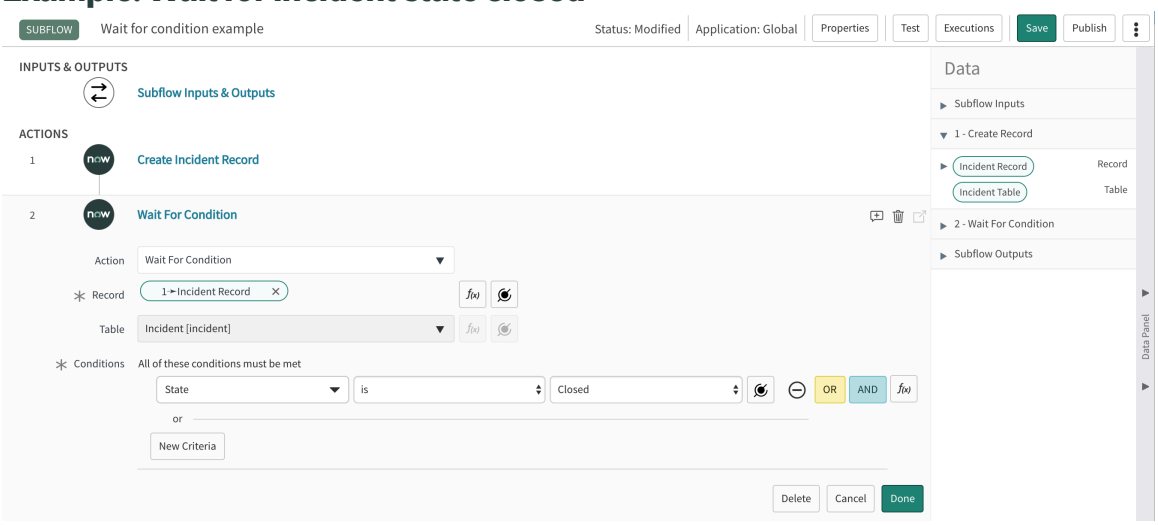


Invalid wait condition

In this example, the wait condition is invalid because it is a time relative to the **Created** date. Actions that have a condition that relies on a time interval will not be met, so the action is never performed.



Example: Wait for incident state closed



In this example, a subflow provides an incident record as an input to the Wait For Condition action. The condition is valid because it uses a field value from the Incident table.

General guidelines

Follow these general guidelines when creating flows that wait for a condition.

Add trigger conditions instead of wait conditions to start flows

If you only want a flow to run when certain record conditions are met, create a flow with a record trigger instead of starting and pausing a flow. A waiting flow consumes more system resources than a flow trigger.

Cancel flows whose resume conditions can never occur

Prevent your flows from waiting indefinitely by specifying flow stop conditions with End Flow flow logic. To free up system resources, you can also cancel any flow whose resume conditions can never be met. For example, cancel flows waiting for incident record updates where the related incident is closed.

Provide at least one condition to resume a flow

The Wait For Condition action requires at least one condition to resume running a flow. If you want to pause a flow for a specific amount of time, use the Wait for a duration of time flow logic instead.

Restrict wait conditions to fields present on the current table

The Wait For Condition action can only monitor changes to the fields of the table to which the record belongs. Don't create conditions that dot-walk to fields in other tables. The action can't detect changes to fields in related records or catalog variables. For example, if an action waits for changes to an Incident record, then it cannot detect changes to a related record such as a catalog item or change task record. Instead of building wait conditions that dot-walk to another record, look up the related record whose value you want to monitor and use that record as the input of your Wait for Condition action. Avoid building wait conditions that rely on catalog variables.

Use a conditions data pill to specify dynamic conditions

To enable flow designers to dynamically apply conditions, define an input of type Conditions and drag-and-drop the input data pill into the **Conditions** field.

Unsupported tables

The system does not support Wait for Condition for the following tables.

Table Category	Table Names
Audit	Sys Audit [sys_audit], Audit Deleted Record [sys_audit_delete], Audit Relationship Change [sys_audit_relation], Audit Roles [sys_audit_role], Audit Relationship Change [sys_audit_relation], Audit Deleted Record [sys_audit_delete]
Email	Email [sys_email], Email Account [sys_email_account], Email Log [sys_email_log]
Events	Event [sysevent], Notification [sysevent_email_action], Stationery [sysevent_email_style], Email Template [sysevent_email_template], Inbound Email Actions [sysevent_in_email_action], Slow Event [sysevent_pattern], Event Registration [sysevent_registration], Script Action [sysevent_script_action]
Import Sets	Import Set [sys_import_set], Import Set Row [sys_import_set_row], Import Set Row Error [sys_import_set_row_error], Transform History [sys_import_set_run], Computer [imp_computer], Notification [imp_notification], Location [imp_location], User [imp_user]
JRobin	JRobin Database [jrobin_database], JRobin Shard [jrobin_shard], Graph Line [jrobin_graph_line], JRobin Shard Fragments [jrobin_shard_location], Member [jrobin_graph_set_member], Round Robin

Table Category	Table Names
	Archive [jrobin_archive], Round Robin Data Source [jrobin_datasource], Round Robin Definition [jrobin_definition], Round Robin Graph [jrobin_graph], Round Robin Graph Set [jrobin_graph_set]
Logs	Log Entry [syslog], Service Portal Log Entry [sp_log]
MID Server	MID Server Property [ecc_agent_property], Mid Server Log [ecc_agent_log], Queue [ecc_queue], Configuration [ecc_queue_config], ECC Queue Statistics (by ECC Agent) [ecc_queue_stats_by_ecc_agent]
Performance Analytics	Job Log [pa_job_logs]
Record Watcher	Responders [sys_rw_action], Channel Responders [sys_rw_amb_action]
Reporting	Summary Set [sys_report_summary], Report Summary Line [sys_report_summary_line]
Scheduled Jobs	Schedule Item [sys_trigger], Broadcast Message [sys_broadcast_message], Broadcast Message Relationships [sys_broadcast_message_m2m], Progress Worker [sys_progress_worker], Progress Worker Domain [sys_progress_worker_domain]
SSO	SSO Properties [sso_properties], Digest Token Properties [digest_properties], SAML Update 1 Properties [saml2_update1_properties], SSO Federation [sso_federation]
System Cache	Cache Flush [sys_cache_flush], Cache Entry [sys_db_cache]
System Clone	ServiceNow Instance [instance], Clone Security Token [clone_token], Preserved Data [clone_preserved_data]
System Dictionary	Dictionary Entry Override [sys_dictionary_override]
System Events	Event Processor [sys_event_processor]
System Fields	Field Class [sys_glide_object]
System Performance	Component Status [sys_status], Cluster Message [sys_cluster_message], Node State [sys_cluster_state]
Text Index	Ts Attachment [ts_attachment], Text Index Attribute Map [ts_attribute_map], Ts Chain [ts_chain], Chain Summary [ts_chain_summary], Text Index Column Attribute Map [ts_column_attribute_map], Text Index Configuration [ts_configuration], Text Index Configuration Attribute [ts_configuration_attribute], Ts Delete

Table Category	Table Names
	Doc [ts_deleted_doc], Ts Document [ts_document], Ts Field [ts_field], Text Search Groups [ts_group], Japanese User Token [ts_japanese_token_dictionary], Ts Phrase [ts_phrase], Global Searches [ts_query], Knowledge Searches [ts_query_kb], Text Search Stat [ts_search_stats], Text Search Summaries [ts_search_summary], Stop Word [ts_stop], Synonym Dictionary [ts_synonym_dictionary], Synonym Set [ts_synonym_set], Text Search Table [ts_table], Text Index Table Attribute Map [ts_table_attribute_map], Service Catalog Searches [sc_ts_query], Ts Word [ts_word], Ts Word Roots [ts_word_roots]
Update Sets	Update Set [sys_update_set], Update Version [sys_update_version], Customer Update [sys_update_xml], Update Set Log [sys_update_set_log]
Upgrades	System Upgrades [sys_upgrade_history], Upgrade Details [sys_upgrade_history_log], System Upgrade Metric [sys_upgrade_metric], Upgrade Blame Log [sys_upgrade_blame], Upgrade Manifest [sys_upgrade_manifest], Upgrade State [sys_upgrade_state]
Usage Analytics	Usage Data for Applications [ua_app_usage], UsageAnalytics Count Configurations [usageanalytics_count_cfg], Application Metadata [ua_app_metadata], UsageAnalytics Count for Tables [usageanalytics_count], Subscription [license_details], Role for Subscription [role_has_license]
Users	User Session [sys_user_session], User Token [sys_user_token], User Preference [sys_user_preference], Navigator History [sys_ui_navigator_history]
Workflow	Workflow Execution [wf_workflow_execution], Workflow History [wf_history], Workflow Executing Activity [wf_executing], Workflow Queued Command [wf_command], Workflow Context [wf_context], Workflow Transition History [wf_transition_history]

Workflow Studio flow logic

Enable flows and subflows to specify conditional or repeated actions. Combine the elements of flow logic to create workflows in a graphical interface with little or no scripting.

The system provides these flow logic options.

Flow logic option	Description
Assign subflow outputs	Specify the data the subflow returns when it completes running. Use subflow output as data for a parent flow or as input for another process.
Call a Workflow	Run a published and active workflow from your flow. You can use the flow data as a workflow input. For example, you can specify the current record as a workflow input.
Do the following until	Apply one or more actions repeatedly until an end condition is met. You can use the flow data to specify the end conditions.
Do the following in parallel	Run actions and subflows in separate paths within an isolated flow logic block.
Dynamic Flow	Identify and run a flow or subflow dynamically by using runtime data. Build templates to provide expected inputs for dynamically called flows or subflows.
End Flow	Stop running the current flow. Use End Flow within a branch of the flow to specify an exit condition. For example, end the flow when it reaches a specific If flow logic block.
For each	Apply one or more actions to each record in a list of records.
Get Flow Outputs	Use this flow logic to access flow contexts and derive the runtime values for dynamic flow outputs.
If	Selectively apply one or more actions only when a list of conditions is met.
Make a decision	You can use the decision table branching logic in situations where multiple conditional paths are required, as an alternative to nested If, Else If, or Else flow logic. For example, if you want to determine what kind of car insurance you need, you can add inputs such as your age, accident history, and car model to the decision table to determine a level of insurance coverage. This logic can save you time and present a more readable format than nested if conditions or switch case statements.
Set Flow Variables	Assign a value to one or more flow variables, which store flow data as data pills. Access flow variable values by referring to their data pill.
Try	Allow a flow to continue running when an error occurs within a flow logic block. Run a sequence of actions in response to errors within the flow logic block.

Flow logic option	Description
Wait for a duration of time	Use this flow logic to give your users time to act during automated processes or to wait for a specific date and time to complete actions

Flow logic inputs

Each flow logic option displays one or more fields that are used to determine its behavior. For example, the **Call Workflow** flow logic has an input where you are able to select a workflow to run. Use these inputs to define the behavior of the flow and enable optional functionality depending on your needs. See flow logic option documentation for a list of the available inputs and how they control the function of that element.

Flow logic outputs

Flow logic options may also have outputs. These represent information that is returned by the flow logic. For example, the **Make a decision** flow logic has an output that contains a decision answer record representing the decision reached by the flow. See the documentation for a flow logic option to see definitions for its outputs.

Assign subflow outputs flow logic

Specify the data the subflow returns when it completes running. Use subflow output as data for a parent flow or as input for another process.

Important:
This flow logic sets values for flow outputs that have already been created. For instructions on creating flow outputs, see Building subflows.

Inputs

Field	Description
Name	Name of the output. Select from the list of outputs available for the flow.
Data	Value for the output. Enter a string value, input a script, or use a data pill. Output values can reference any data pill from earlier in the flow, including other outputs. If you set outputs values by reference to other data pills, you must maintain the order of the output assignments. The referenced value must always come before the output that uses the referenced value. Changing the order may produce null values. To assign an empty value, leave this field empty. **Note:** Flow output values are set in the order in which they are assigned from top to bottom. If you set the value of the same output multiple times, the flow only uses the last value set.

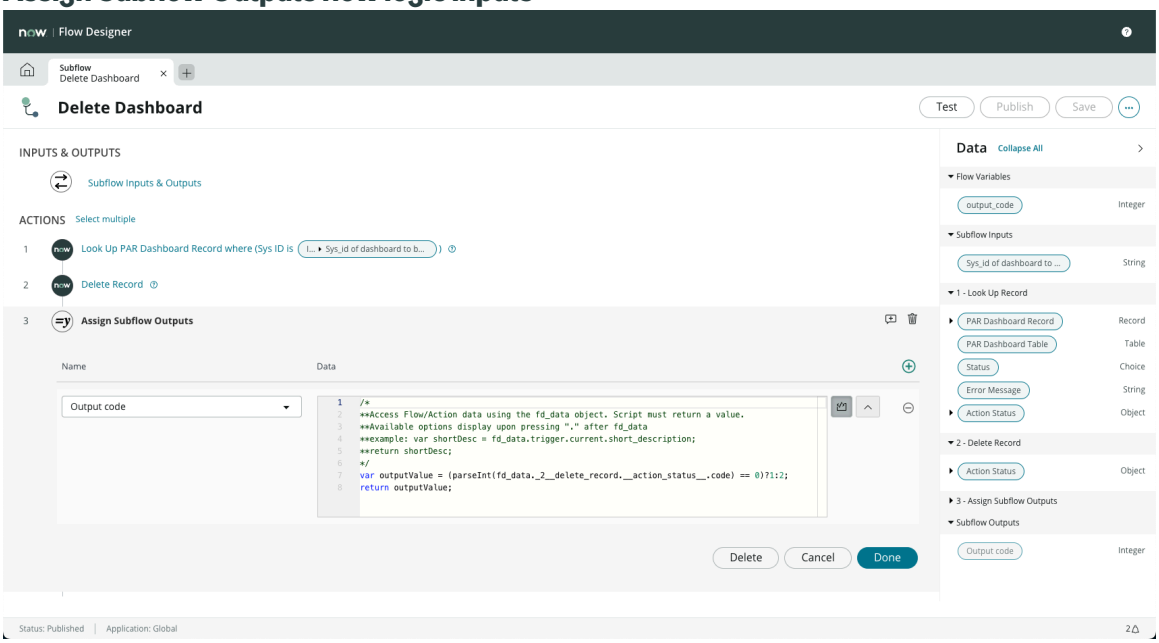
Outputs

This flow logic produces no outputs of its own, but it does set values in the **Subflow Outputs** section of the Data pane.

Example: Set the output code of a Delete Record action

In this example, the flow uses the Sys ID of a dashboard to look up a record, delete the record, and then return the action status code of the delete operation. The subflow assigns the output value of the Output code flow variable.

Assign Subflow Outputs flow logic inputs



Design considerations

Follow these design considerations when assign output values from a subflow.

Do not assign subflow output values within loops

Subflow outputs are intended to be static values generated at the completion of the subflow. Loops do not have access to subflow output values while the subflow is running. Assigning subflow output values within a loop can produce unexpected results such as the loop only receiving the last value set. If you need to generate dynamic values that change within a For each or Do until loop, use flow variables instead.

Call a workflow flow logic

Run a published and active workflow from your flow. You can use the flow data as a workflow input. For example, you can specify the current record as a workflow input.

Inputs

Input	Description
Select a Workflow	Published and active workflow that you can select to run. The workflow that you select determines the records that are associated with it. If the workflow has inputs, Workflow Studio displays them as additional flow inputs.

Input	Description
	Note: • To prevent the workflow from running outside of Workflow Studio, modify it to remove its start conditions. • You cannot select a workflow that runs on the Requested Item table. Instead, create a new flow with a Service Catalog trigger.
Wait?	Workflow that you set to true so that the flow waits for workflow completion before continuing. Only workflows that wait for completion can return certain output values to the flow. Set to false to continue running the flow separately from the workflow. **Note:** If the workflow is canceled or its context record is deleted prior to the workflow finishing, the flow stops waiting and instead continues running.
Current	Current record that the workflow processes. Select a data pill that contains a record from the associated workflow table.

Outputs

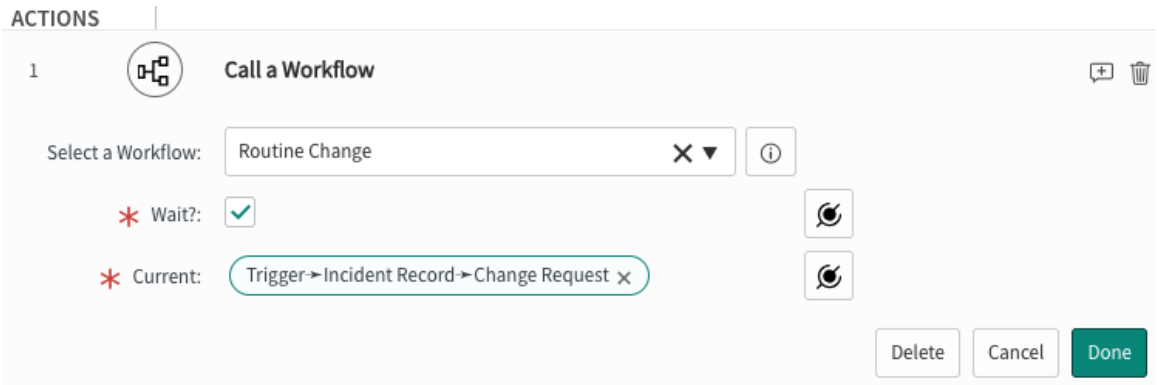
The flow execution details only display workflow output values that are generated while the flow is running. If you configure the flow to wait for the workflow to finish, the flow execution details can display all workflow output values. If the flow does not wait, the flow execution details only display the workflow output values that were generated before the **Call a Workflow** flow logic completes. If **Call a Workflow** completes before the workflow finishes, the workflow output values stop updating and only display the last known value.

Output	Description
State	State of the workflow. This value comes from the workflow context record. The state will be **Complete** if the workflow executes successfully. If the workflow is canceled, the workflow state is set to **Canceled**. If the context record is deleted prior to the workflow finishing, the workflow state is set to **Invalid**.
Context	Reference to the workflow context record.
Result	String that contains the result from the workflow. This value comes from the workflow context record.

Output	Description
Return value	String that contains the return_value from the workflow. This value comes from the workflow context record.

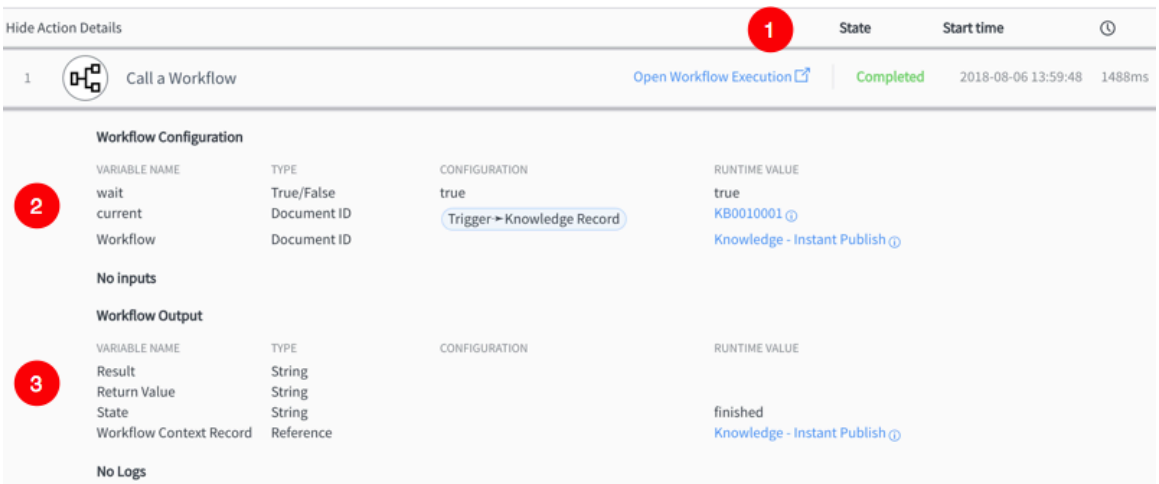
Example: Calling the Routine Change workflow

In this example, the flow calls the **Routine Change** workflow. The **Wait?** option is checked, so the flow pauses until this workflow completes. The **Current** field is filled using a data pill representing the record that triggered this flow.



Execution details

Call Workflow execution details



1. The header displays a link so that you can view the workflow progress in the Workflow Editor, the flow logic state, the start time, and the runtime duration.
2. The **Workflow Configuration** section displays how the flow logic was configured for this flow and the runtime values that were generated.
3. The **Workflow Output** section displays the output that is generated by the workflow while the flow is running.

Note:
If you cancel the workflow or delete the context record prior to the workflow finishing, the flow logic state is set to **Complete**.

Do the following until flow logic

Apply one or more actions repeatedly until an end condition is met. You can use the flow data to specify the end conditions.

You can use **Do the following until** flow logic to create a loop that repeatedly applies one or more actions. This flow logic requires a condition specifying when to stop the loop.

Note:
When you set a data pill value from inside a Do the following branch of flow logic, the data pill value is only available to other actions in the same branch. Referencing a data pill value that was set inside a Do the following branch from outside of the flow logic branch produces a null value.

Inputs

Condition label

Data type: *String*

Text description of the condition that you want to display in the flow.

Conditions

Data type: *Conditions*

Conditions under which the loop terminates. You could, for example, end a loop when the state of an incident changes. If the end condition is true when the flow starts, the loop runs once and then stops.

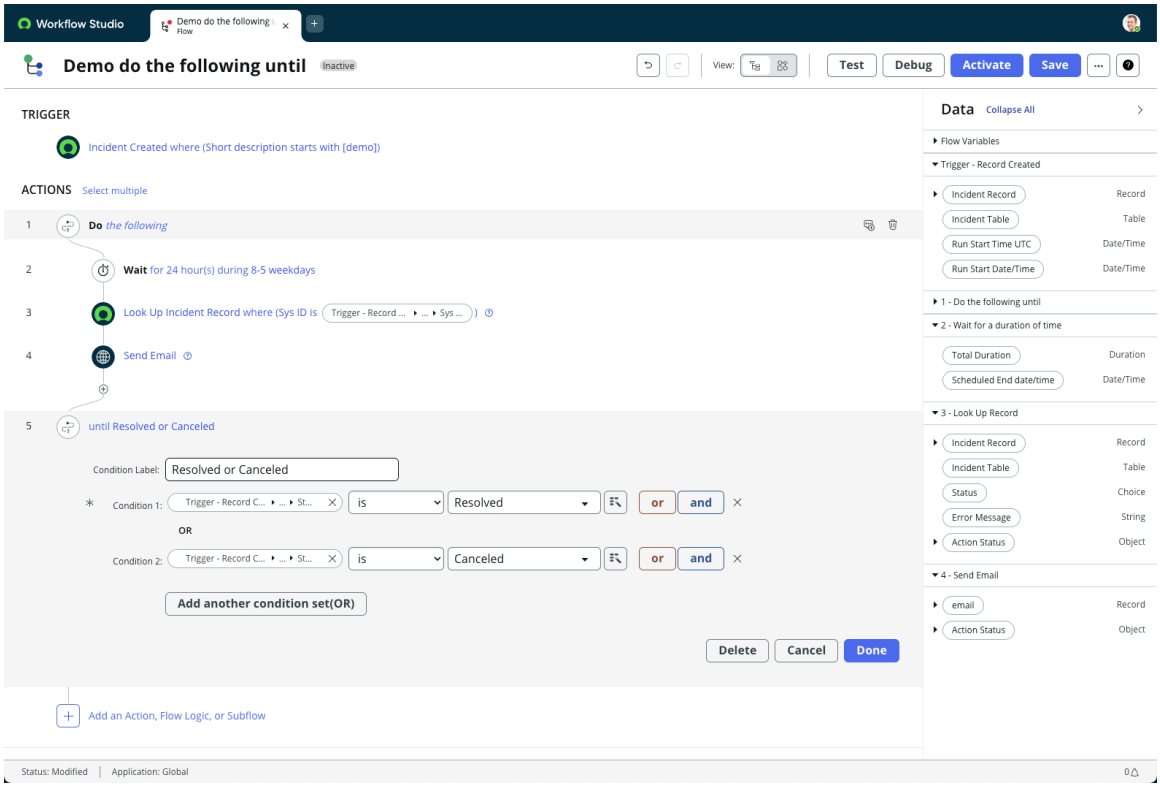
Tip:
You can use flow variables to check for custom conditions such as the number of times a loop has run.

Outputs

This flow logic produces no outputs.

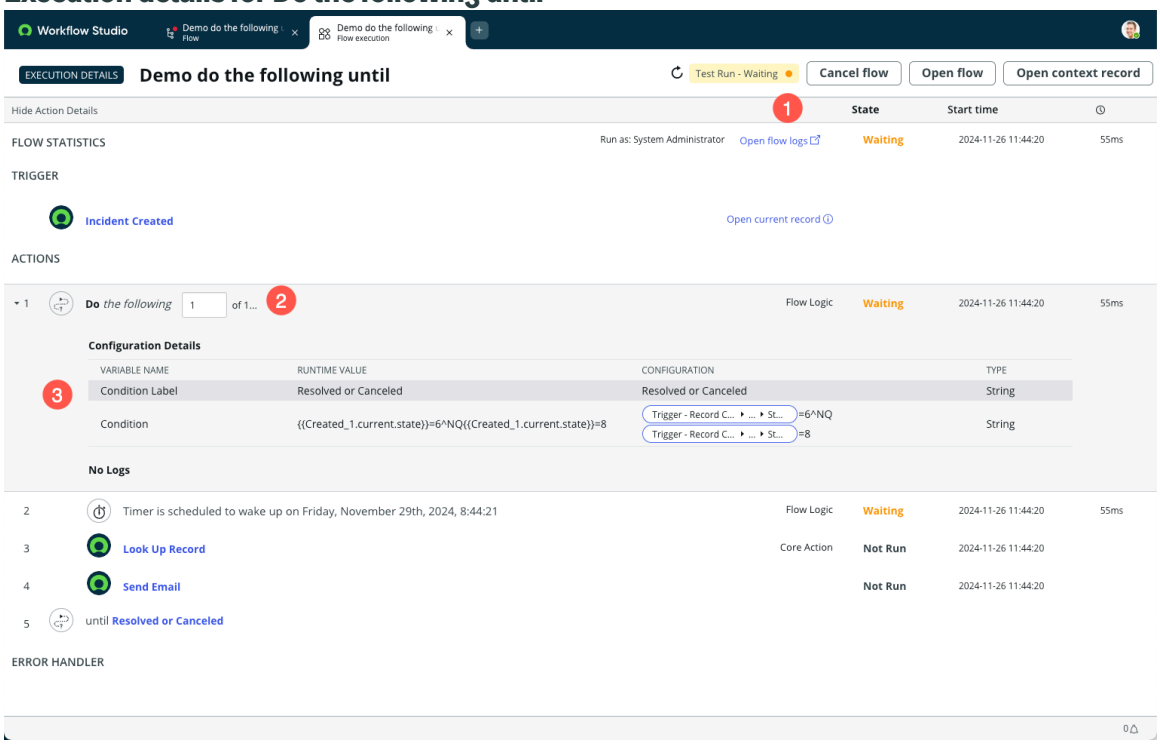
Example: Send a daily email until an incident is resolved

In this example, the flow sends a daily email about the incident, until the incident is in a closed or canceled state. Inside the **Do the following** branch, there is a step for looking up the incident record.



Execution details

Execution details for Do the following until



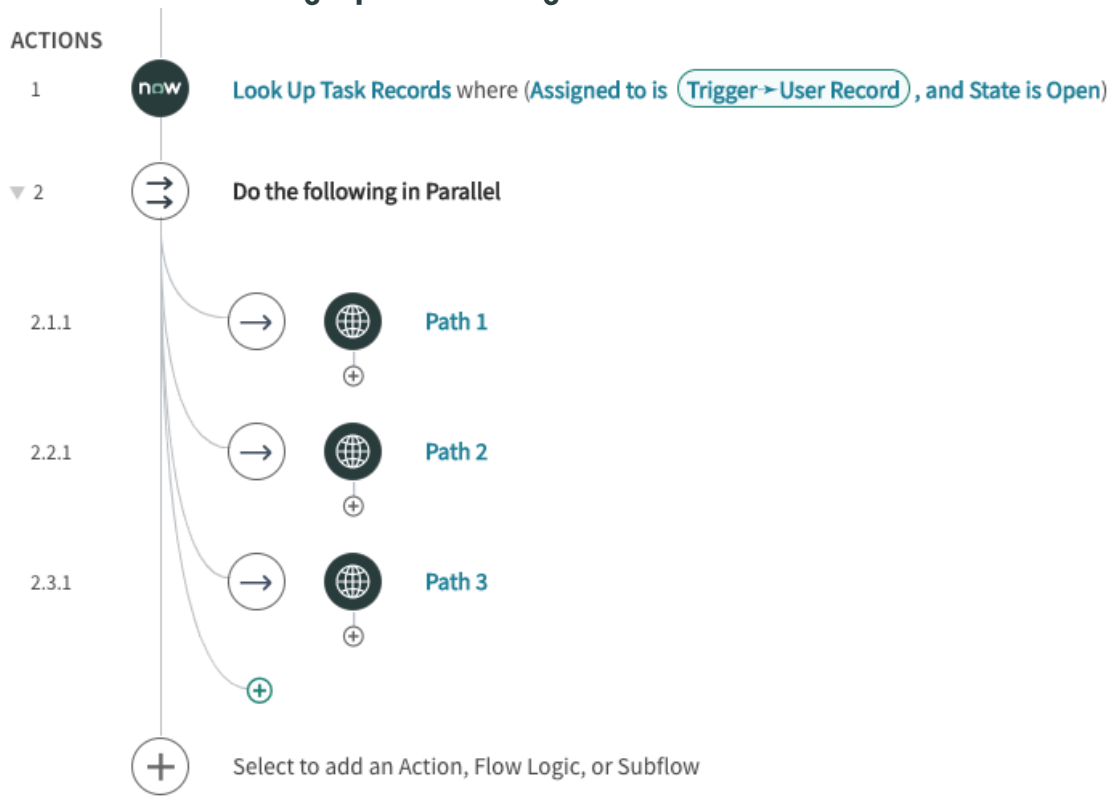
1. The header shows the state, start time, and runtime for the flow logic.
2. This flow logic can run actions or subflows multiple times until it's condition is met. Use the arrow icons to select an iteration and its values.
3. The Actions section shows details on the actions, flows, or subflows that are run during this loop iteration.

Do the following in parallel flow logic

Run actions and subflows in separate paths within an isolated flow logic block.

With this flow logic, you can run actions and subflows in separate paths. If any action within the Do the following in parallel flow logic block must wait, other actions run until all paths within the block finish processing.

Paths in a Do the following in parallel flow logic block



Note:
Paths in a Do the following in parallel flow logic block do not run in multiple threads, since a flow execution context runs in a single thread. However, there may be times when you want to run flows within separate contexts even though this may consume more of your instance's resources. To run subflows in separate flow contexts within the same flow, see Dynamic flows.

Inputs

Do the following in parallel flow logic does not have field inputs. Instead, it displays a plus (⊕) icon that enables you to create a path with actions or subflows.

The actions and subflows in each path run until all tasks within the flow logic block have completed.

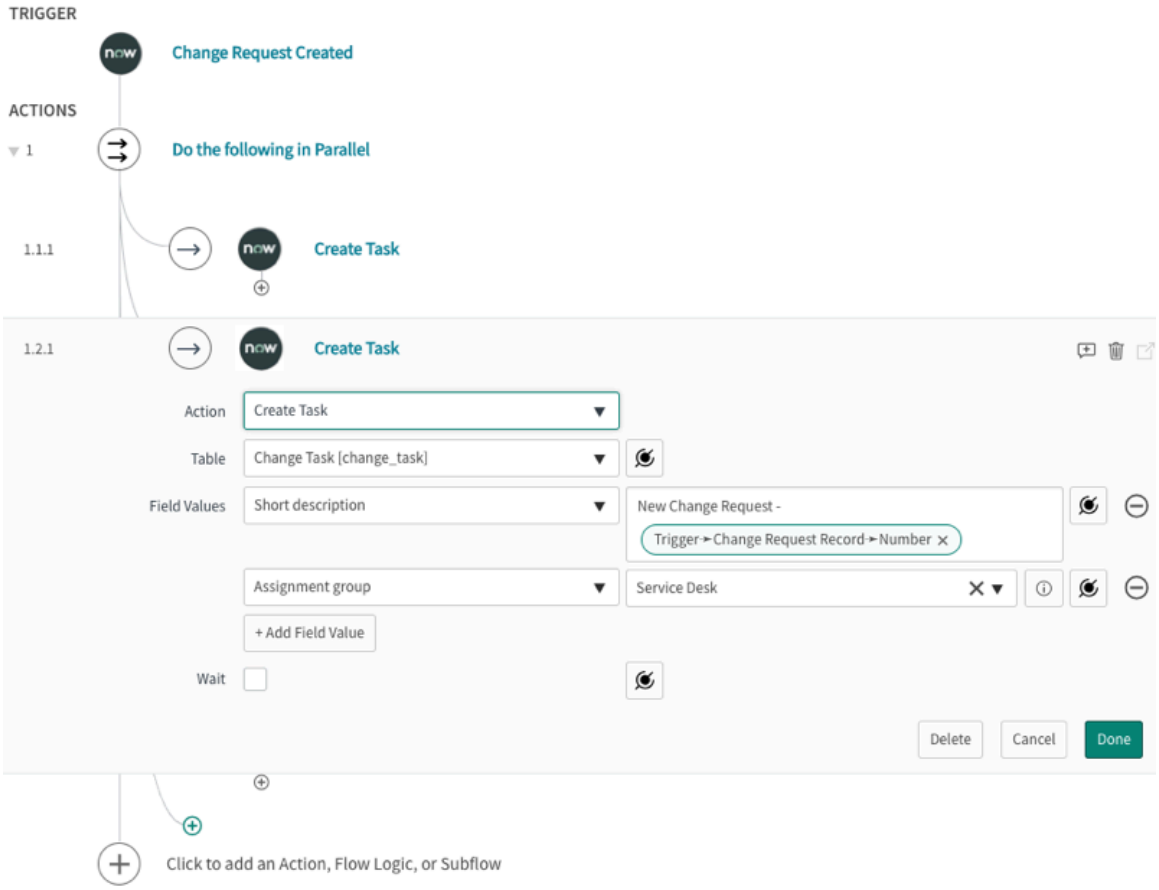
Outputs

This flow logic has no outputs, but actions and subflows in each path may have outputs. While the flow is running, outputs from a path are only accessible to other actions in the same path. After the Do the following in parallel flow logic completes, its final outputs are accessible to the rest of the flow.

Example: Create two tasks in parallel when a change request is created

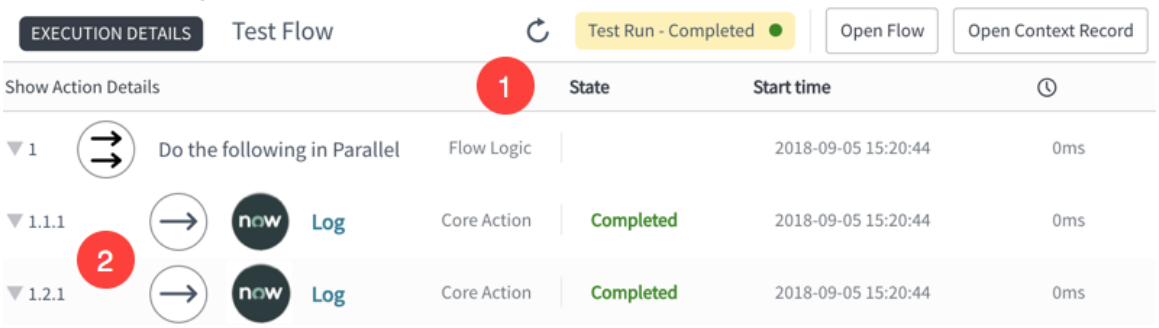
In this example, a flow triggers when a new change request is created. Using **Do the following in Parallel**, two tasks are created in separate paths and are assigned to different groups. The flow uses the **Number** field data pill from the triggering change request to display the number in the short description for the task record.

Do the following in parallel flow logic inputs



Execution details

Do the following in parallel execution details



1. The header shows the state, start time, and runtime for the flow logic.
2. The Configuration Details section shows the state, start time, and runtime for each path in the flow logic block.

General guidelines

Avoid creating data dependencies between paths

Since a flow can run paths in any order, avoid creating data dependencies between separate paths. For example, do not have one path that creates a record and another path that updates the same record. The update record path may run before the create record path.

Do not share data between paths

Workflow Studio prevents you from dragging data pills between paths because the system cannot determine which path will finish first to supply the output value.

Dynamic flows flow logic

Identify and run a flow or subflow dynamically by using runtime data. Build templates to provide expected inputs for dynamically called flows or subflows.

The Dynamic Flow flow logic calls a flow or subflow during runtime by using the data that you specify during the flow design. You can use Dynamic Flow to select which flow to run when multiple flows have similar names or purposes.

To use Dynamic Flow, the flow designer does the following:

1. Creates and publishes a flow or subflow to use as a template.
2. Adds Dynamic Flow to a parent flow.
3. Selects the flow template for Dynamic Flow.
4. Enters the flow name for Dynamic Flow.
5. Enters the required inputs specified by the flow template.

Dynamic flow templates

The template for Dynamic Flow can be a flow or a subflow. The template's inputs must match the inputs of any flow or subflow that you call dynamically. An input matches when it has the same **Label** and **Name** field values in each flow or subflow.

After you build a template, consider copying it to create similarly named and similarly performing flows or subflows. Make sure that you name each flow or subflow with a standard naming convention that can be generated by Dynamic Flow. For more information on creating a template for Dynamic Flow, see Getting started with Dynamic Flow and Get Flow Outputs.

Inputs

The following inputs always appear when you add the Dynamic Flow flow logic to a flow.

Input	Description
Flow Template	Template whose inputs the dynamic flow copies and displays. The inputs for the template must match the inputs of the subflow that you want to run.
Flow	Name or Sys ID of the flow or subflow that you want to run. Flow names must be the display name not the flow internal name. For example, enter the name `My dynamic flow not my_dynamic_flow`. Generate a flow or subflow name dynamically by entering a string and concatenating it with data pill values.

Input	Description
	You can call a flow or subflow that is associated with another application scope by entering a scope-name.flow-name format. The user running the flow or subflow must have access to the application scopes that you specify. If you do not specify an application scope, the dynamic flow runs in the parent flow's scope. **Note:** If the system can't find the flow or subflow, it skips the **Dynamic Flow** flow logic step and logs an error message.
Wait for completion	Option to force the parent flow to wait until the dynamic flow finishes running before proceeding with the next action.

Note:
You see more inherited inputs after you select a **Flow Template**.

Get flow outputs

Dynamic Flow outputs appear in the data panel as Record type data pills that are named as Context. You can use these flow contexts to derive the runtime values for dynamic flow outputs with the Get Flow Outputs flow logic. To use Get Flow Outputs, the flow designer does the following:

1. Adds a Dynamic Flow to a flow.
2. Adds Get Flow Outputs after the dynamic flow.
3. Specifies the value for the **Context** input by clicking the data pill picker ( ) and selecting **Dynamic Flow** > **Context**.

For more information on using Get Flow Outputs, see Getting started with Dynamic Flow and Get Flow Outputs.

General guidelines

Use dynamic flows if you have multiple subflows with similar functionality

Dynamic flows let you compartmentalize your processes by applying a template to handle the inputs of multiple similar subflows. Compartmentalization lets you distinguish between subflows that perform similar functions, such as subflows for IntegrationHub spokes.

Ensure dynamically called subflow inputs match template flow inputs

The system throws an error and the main flow can't run properly when the inputs of a dynamic flow and flow template don't match.

Use the correct context when getting flow outputs

A context record uniquely identifies the flow run. If you run a dynamic flow multiple times, there are multiple context records to choose from. When you use dynamic flow multiple times within a flow, make sure to pick the right context record from the right run each time you get flow outputs.

End Flow flow logic

Stop running the current flow. Use End Flow within a branch of the flow to specify an exit condition. For example, end the flow when it reaches a specific If flow logic block.

End Flow logic can be contained within a conditional flow logic block, such as an **If**, **Else If**, or **Else** flow logic block. Use this flow logic to stop a flow when certain conditions are met. You cannot add actions or flow logic after you use the **End Flow** flow logic. All branches of a flow are ended when a flow reaches the **End Flow** flow logic, including the **Wait for a duration** logic. Any branches that run in parallel also stop their progress when you use the **End Flow** flow logic.

Inputs

This flow logic has no inputs.

Outputs

This flow logic has no outputs. When this flow logic ends a flow or subflow, it sets the state to Completed. Subflows that end can only return an output value to a parent flow if there was an Assign subflow outputs flow logic prior to the End subflow flow logic.

General guidelines

Use the general guidelines when adding an End flow or End subflow flow logic.

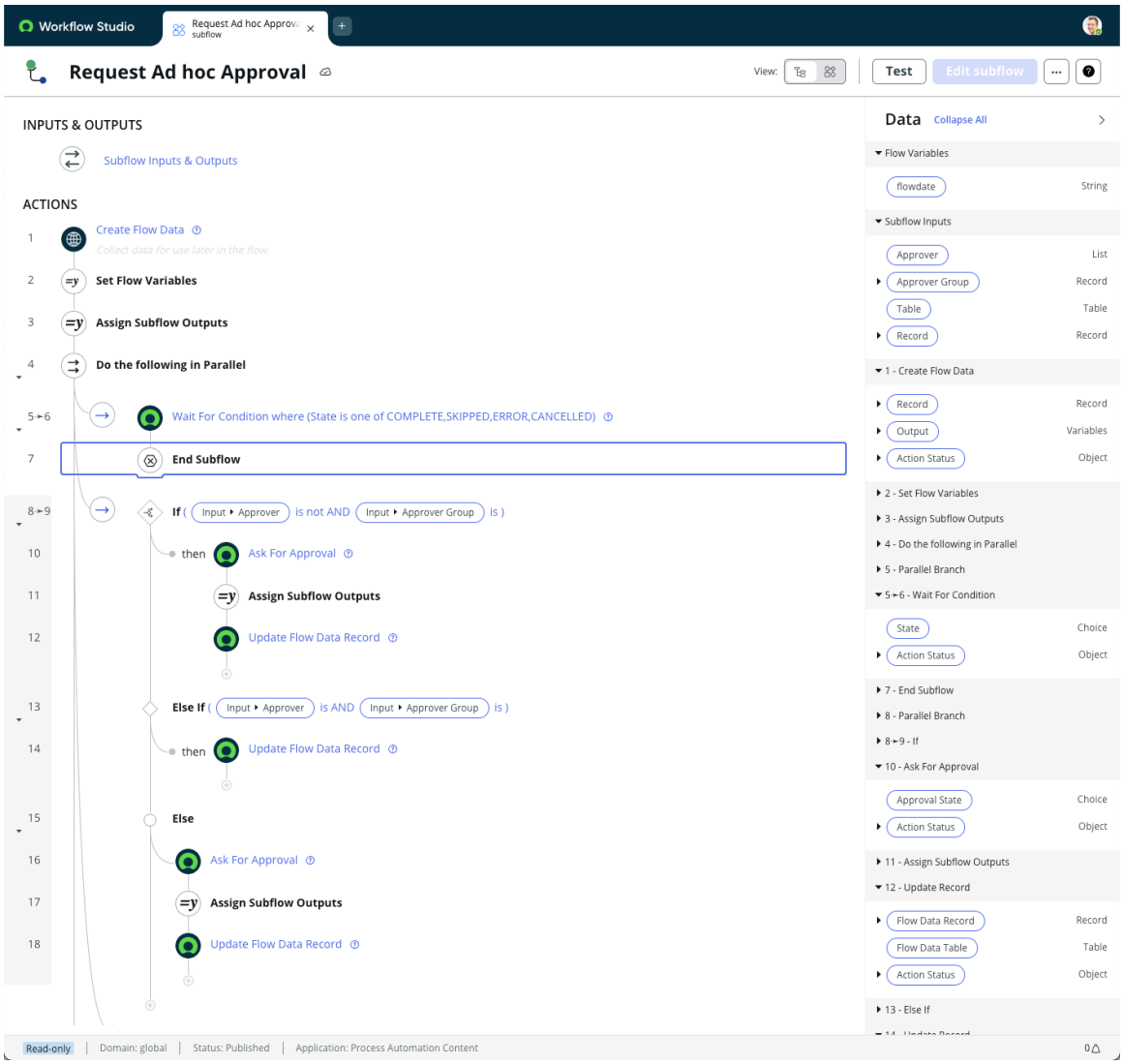
Assign subflow outputs prior to End subflow flow logic

If a subflow always needs to return an output value to a parent flow, use an Assign Subflow Outputs flow logic prior to the End Subflow flow logic. By default, an ended subflow does not return any output values.

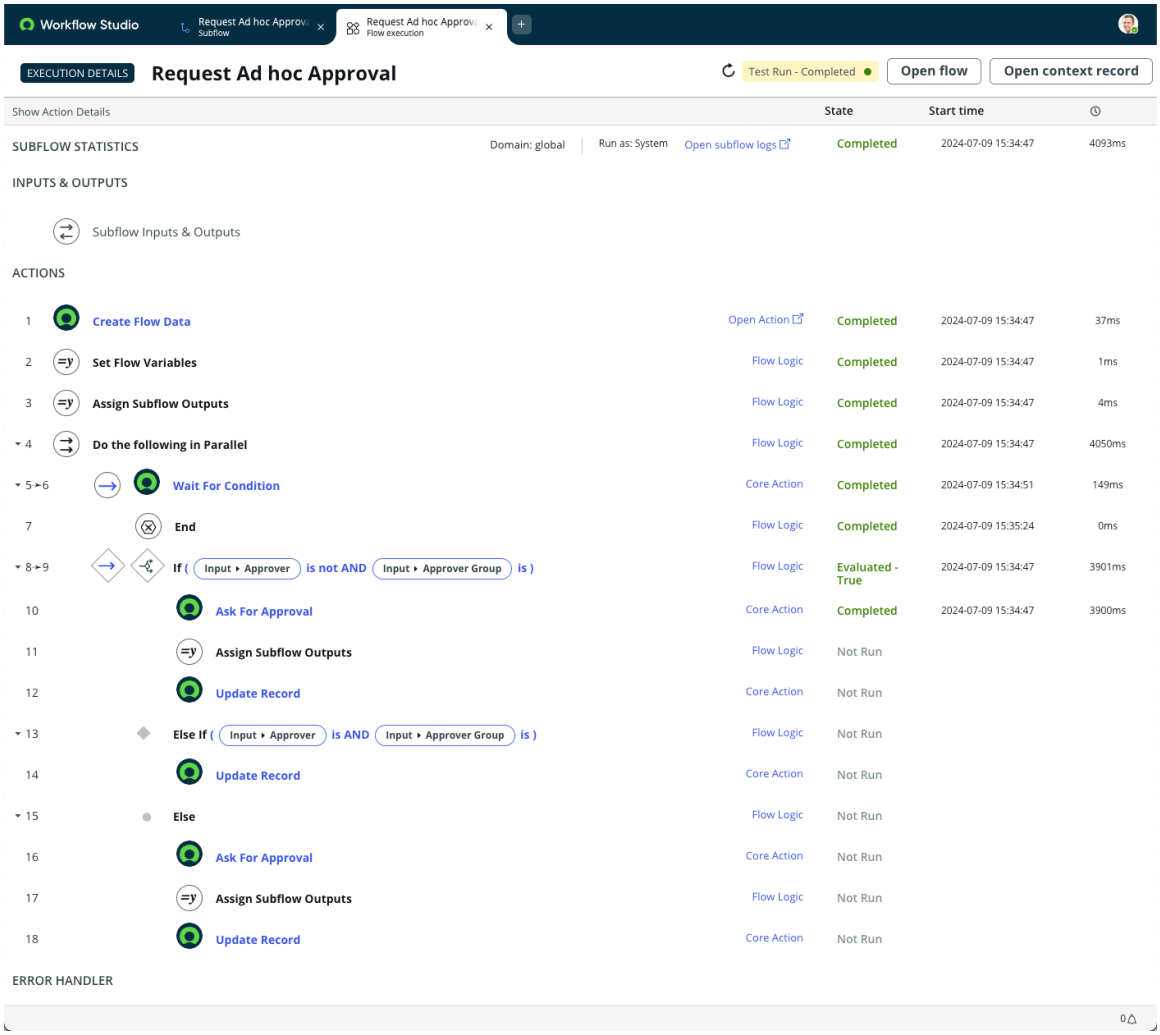
Design parent flows to handle an ended subflow

If you call a subflow that has an end branch, make sure that the calling flow can handle not receiving subflow output. It is up to the flow and subflow author to pass data between flows.

Example: Request Ad hoc Approval subflow



This example subflow supports adding arbitrary approvals from a Playbooks activity. The wait condition in steps 5 to 6 checks the flow data record for an end state such as cancelled, complete, error, or skipped. When the flow data record enters one of these states, the subflow ends without assigning any outputs.



When the subflow reaches this branch, it stops the subflow and sets the state to Completed. The actions in other branches are not run.

Exit Loop flow logic

Exit from a flow logic loop when the conditions of an If flow logic are met. Continue running the flow from the next step after the flow logic loop. This flow logic is also known as break.

Valid Exit Loop placement

You can only add Exit Loop flow logic within certain portions of a flow. The Exit Loop flow logic must be within a branch of a parent For Each or Do the following until flow logic block.

- Then branch of an If flow logic block
- Then branch of an Else If flow logic block
- Then branch of an Else flow logic block

Inputs

This flow logic has no inputs.

Outputs

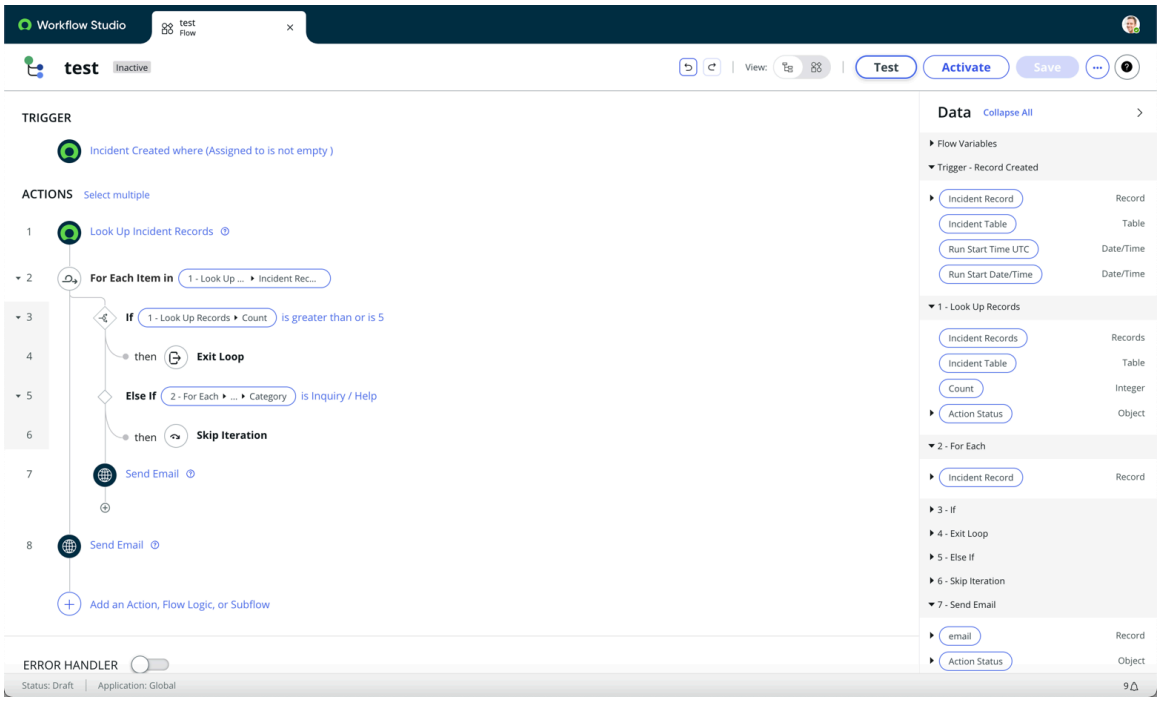
This flow logic has no outputs.

Execution Details

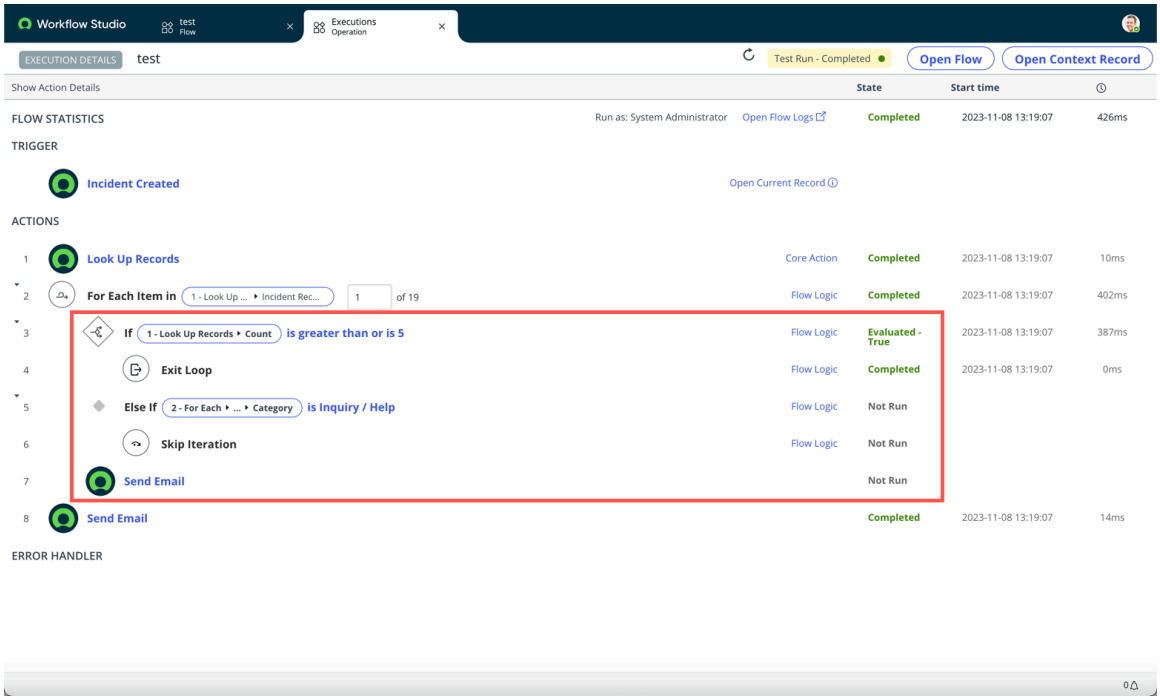
When a flow exits a loop, the state of the Exit Loop flow logic block becomes Completed. Any remaining steps in the For Each or Do the following until flow logic block aren't run.

Example: Exit a loop based on incident count

In this example, a flow generates a list of incidents assigned to a user. For each incident that is assigned to the user, the flow sends an email. If the list of incidents is greater than or equal to 5, then the flow exits the For Each flow logic loop and doesn't send an email.



In this example, there are 19 incidents assigned to the user, which meets the exit conditions. The first item of the For Each flow logic counter shows the Exit Loop flow logic having a state of Completed.



For Each flow logic

Apply one or more actions to each record in a list of records.

The **For Each** flow logic applies one or more actions to a list of records. The flow applies the actions contained within the flow logic to each record in the list.

> **Note:**
> When you set a data pill value from inside a For each item branch of flow logic, the data pill value is only available to other actions in the same branch. Referencing a data pill value that was set inside a For each branch from outside of the flow logic branch produces a null value.

Iterating over a large number of records can be resource intensive, especially when the For Each logic block includes complex actions for each iteration. To avoid performance issues, turn off reporting using the *com.snc.process_flow.reporting.level* system property. For more information, see Workflow Studio flow system properties.

Inputs

Items

Data type: *Records*

List of Sys ID values or Records data pill specifying the records to process in sequence. You can use a Look Up Records action to generate a list of records. For more information, see Look Up Records action.

> **Note:**
> If you want to process items in a particular order, you must first sort the items in this input in advance. For example, use the Order by option to sort the results of a Look Up Records action.

Outputs

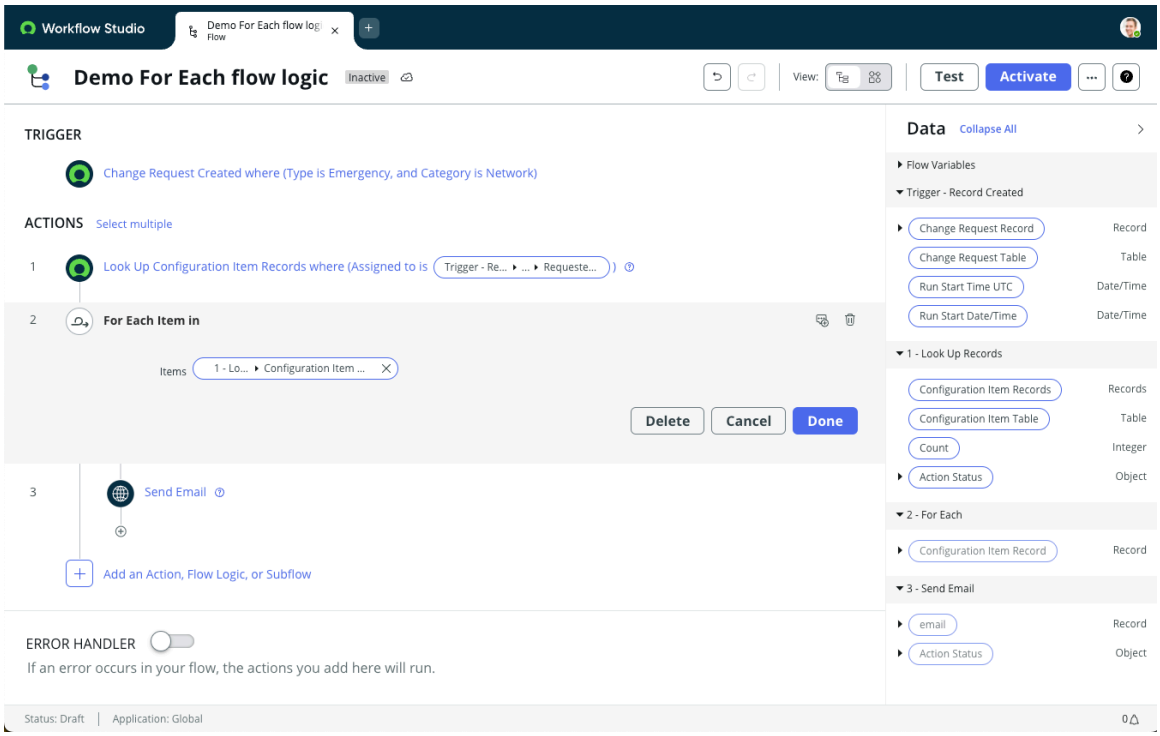
[Table name] Record

Data type: *Record*

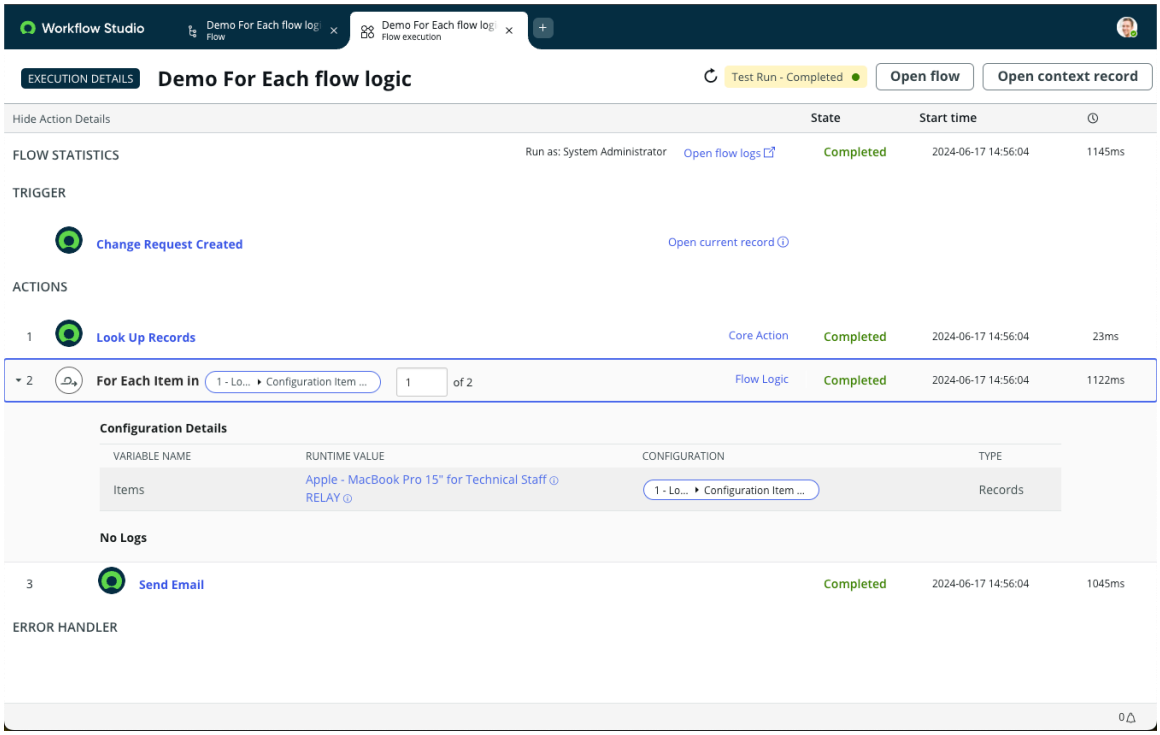
Current record in the loop.

Note:
By default, all flow loops only store execution details for the first and last iterations of a loop. To report on all iterations of a loop, create a flow execution setting record for each flow that you want to collect loop execution details. For more information about flow execution settings, see Flow execution settings.

Example: Send an email for each configuration item potentially affected by a change



This example flow starts when a change request record is created. The flow uses a Look Up Records action to find Configuration Item records assigned to the requester of the change request. The flow uses For Each flow logic to send an email about each configuration that might be affected by the change request. The output of the Look Up Records action contains the list of records to process.



The flow execution details show the configuration item record used for each iteration of the loop.

General guidelines

Use these general guidelines with a For Each flow logic.

Avoid adding more than 1000 items

Avoid iterating over lists with more than 1000 records. Keep your list of records smaller to optimize flow performance. To iterate over lists with more than 1000 records, divide the list into smaller sections and use multiple flows.

Avoid defining stages that depend on a For Each flow logic

Flow Designer prevents you from adding stages within a **For Each** block. You can only add stages before or after a **For Each** block.

Avoid nested For Each loops

Avoid nested For Each loops that process many records. Nested loops may cause the flow to run until it is stopped by the flow transaction quota rule, which prevents flows from running longer than an hour. For more information about transaction quotas, see Transaction quotas.

Get Flow Outputs flow logic

Retrieve output values generated by dynamic flows or subflows.

Roles and availability

Available as part of ServiceNow Core.

Role requirements

This flow logic 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 flow logic needs. To add dynamic values, you can also select data pills using the pill picker.

Flow Template

Data type: *Reference*

Flow or subflow that was used as a template for a dynamic flow. A flow template provides the inputs for a dynamic flow.

Context

Data type: *Document ID*

Context record generated by a Dynamic flow flow logic.

Outputs

You can use these outputs as inputs for other items.

Response

Data type: *JSON*

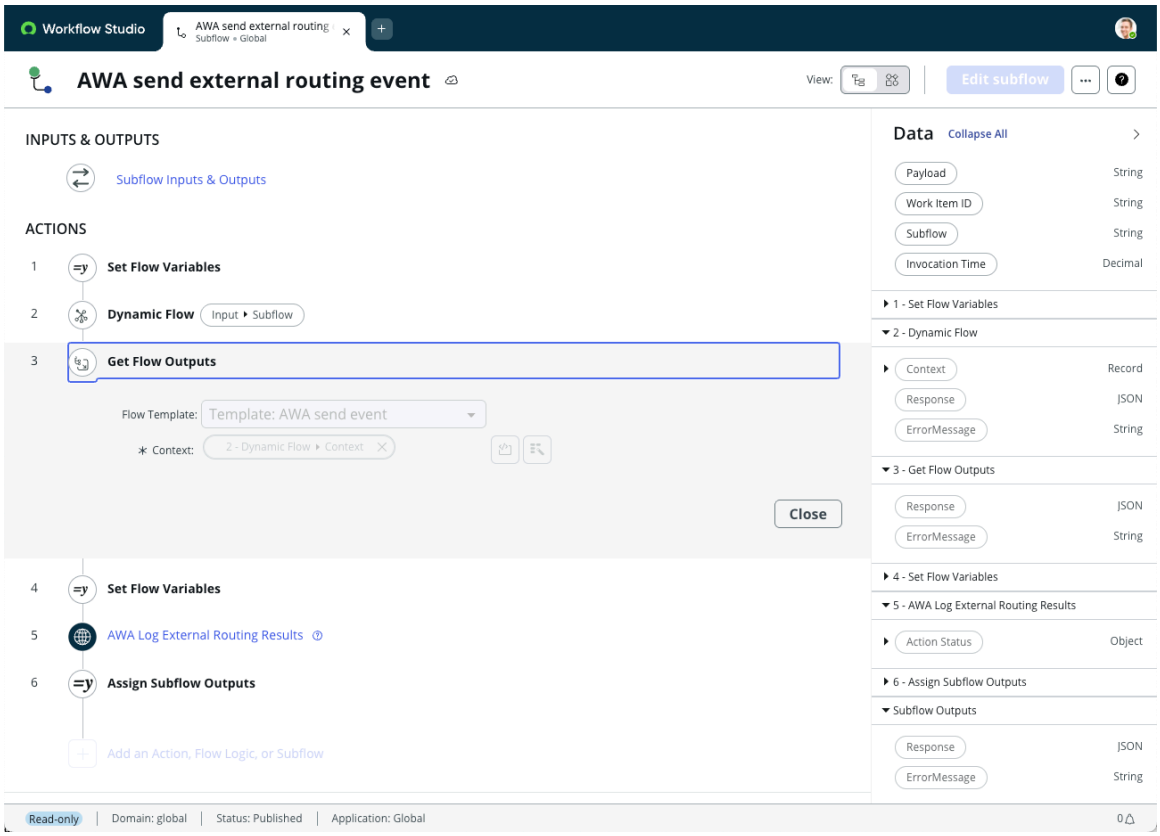
Output values generated by the dynamic flow or subflow. You can use the Response to set flow variables or assign subflow outputs.

Error Message

Data type: *String*

Error message generated if there is no response.

Example: Get outputs from dynamic AWA flows



The Advanced Work Assignment (AWA) application includes a read-only flow called AWA send external routing event. This flow runs a dynamic flow based on several input values. The Get Flow Outputs flow logic is configured to use the AWA send event flow as the **Flow Template** input value, and uses the output of the Dynamic Flow flow logic as the **Context** input value. The flow stores the **Response** output of the Get Flow Outputs flow logic as a flow variable. This flow variable is also assigned as a subflow output for use in other flows.

Go back to flow logic

Return to a prior step in the flow to repeat a sequence of actions.

Family release requirements

You can only add Go back to flow logic to new flows created from the Washington DC family release and forward. Flows that were created in versions prior to the Washington DC family release do not support Go back to flow logic.

Valid Go back to placement

You can only add Go back to flow logic within certain portions of a flow.

- The Go back to flow logic must be within a branch of a parent flow logic block.
  - Then branch of If, Else If, or Else flow logic
  - Answer branch of Make a decision flow logic
  - Catch branch of Try flow logic
- The Go back to flow logic must be outside of the Error handler section.

Valid Go back to target step

Go back to flow logic only allows you to select a valid return target step. A valid target step must meet all of these conditions.

1. The target must be a step before the Go back to flow logic.
2. The target can't be a step within a different branch of the flow.
3. The target can't be a step within the same branch as the Go back to flow logic.
4. The target can't be a step within a non-branching flow logic block unless the non-branching flow logic block also includes a child Go back to flow logic.
5. The target can be a step outside of a non-branching flow logic block except for Do the following in parallel flow logic.

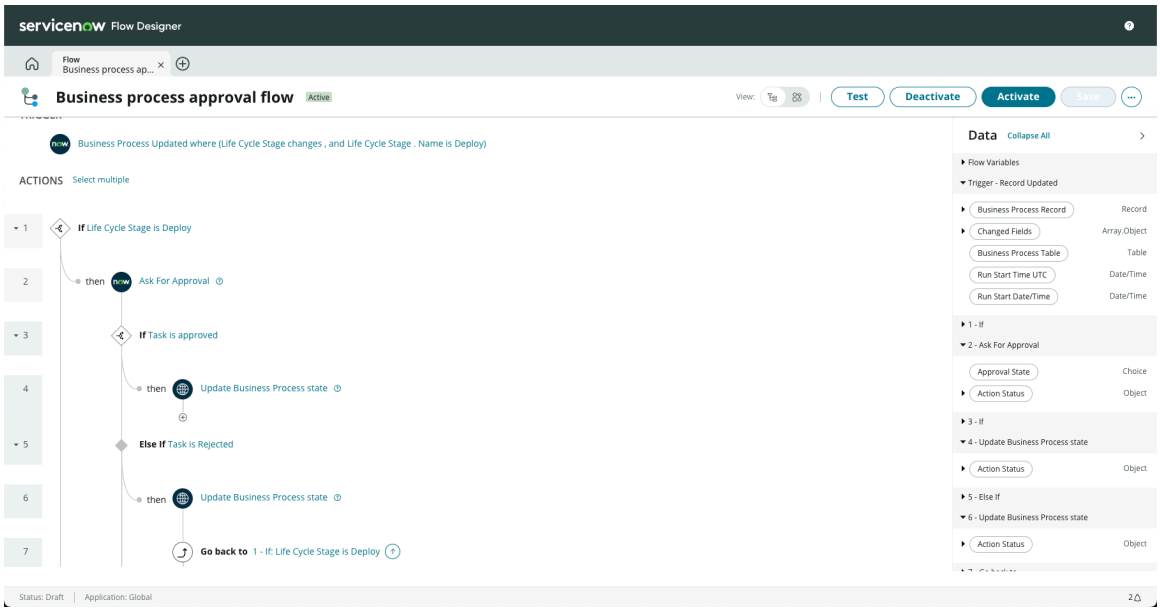
> **Warning:**
> Selecting a target step outside of a parent flow logic block exits the current loop and resets its loop iteration count. The system displays separate loop iteration counters for the Go back to loop and the parent flow logic block. All loops are limited by the maximum number of loop iterations property (*sn_flow_designer.max_iterations*).

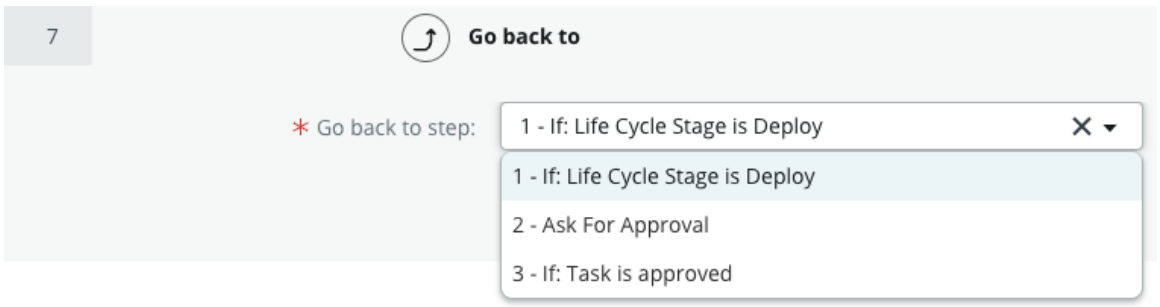
Inputs

Input	Description
Go back to step	Step in the flow that meets the conditions of a Go back to step target.

Example: Go back to start of flow

In this example, the flow goes back to the first step when the approval task for the trigger record is rejected. Valid Go back to targets include steps 1, 2, and 3. Step 4 violates rule 2 as it's a step within a different branch. Steps 5 and 6 violate rule 3 in that they're steps within the same branch of the flow.





Outputs

This flow logic has no outputs.

General guidelines

Use these general guidelines when adding Go back to flow logic.

Add Go back to flow logic after the flow structure is complete

Go back to flow logic depends on a fixed sequence of steps to function properly. Wait to add Go back to flow logic until the flow has valid target steps.

Avoid creating duplicate Go back to steps

A flow uses the first Go back to flow logic whose conditions are met. The flow ignores all Go back to flow logic steps after the first.

Avoid creating infinite loops

Specify a condition to resume the flow or to throw an error with each Go back to loop. Error and resume conditions prevent a flow from running until it reaches the maximum number of loop iterations (*sn_flow_designer.max_iterations* property). You can use an If flow logic to check for loop end conditions. For example,

create a flow variable that counts how many times the flow has run the Go back to flow logic. When the flow variable reaches a limit, end the flow.

Delete Go back to flow logic to move it

After you add Go back to flow logic, you can't move it to another location. You can only delete it from its current location and then add it to another valid location.

If flow logic

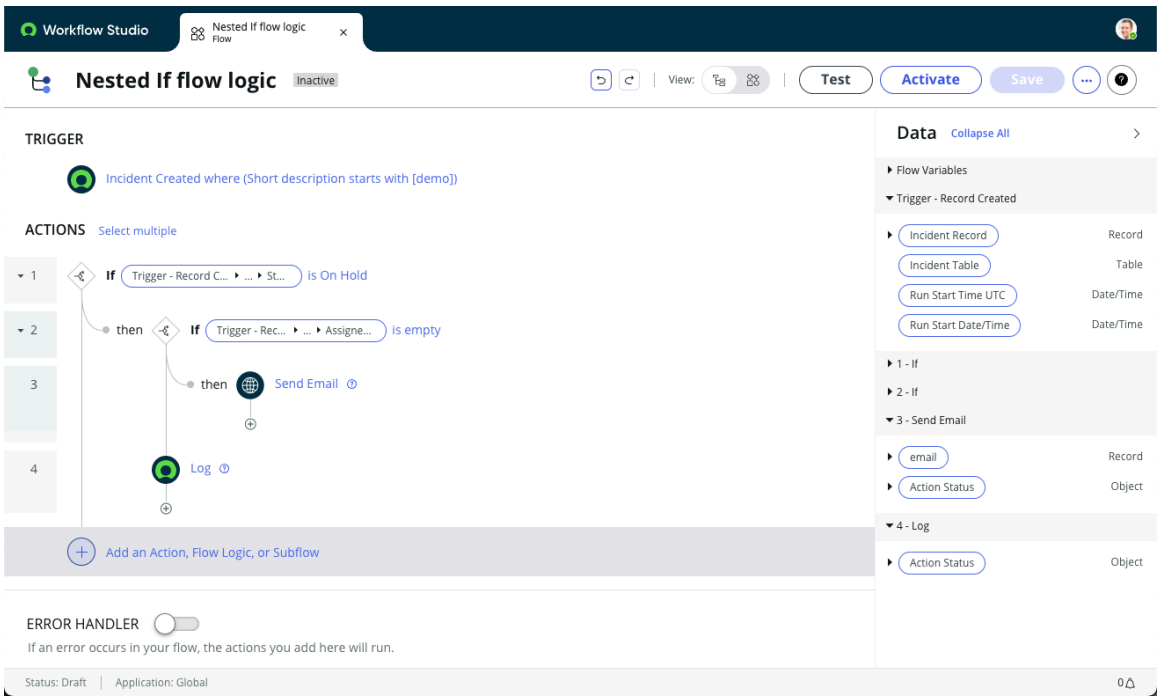
Selectively apply one or more actions only when a list of conditions is met.

Use this flow logic by specifying conditions that are based on the current record. Actions or subflows can be added to the flow within the if flow logic. The flow only runs the actions within this flow logic when the conditions evaluate to true. You build conditions based on data in records, such as the value of a task state or urgency.

Note:
When you set a data pill value from inside a Then branch of If flow logic, the data pill value is only available to other actions in the same branch. Referencing a data pill value that was set inside a Then branch from outside of the flow logic branch produces a null value.

Nested If flow logic blocks

You can add a child If flow logic block to a parent If flow logic block. Add the child If flow logic block to the Then branch of the parent flow logic block.



Inputs

Input	Description
Condition label	Descriptive label for the conditions of branch. A label can be easier to read than a long or complex condition data pill value.

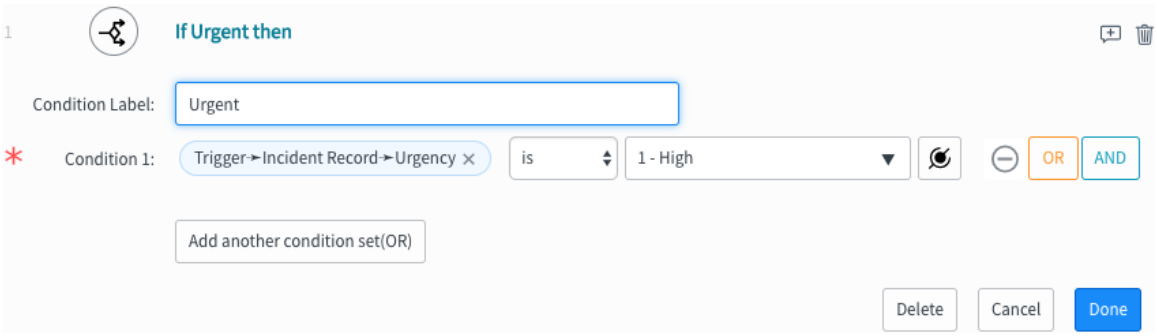
Input	Description
Condition	Conditions under which the branch runs. The flow only runs the contents of the Then branch when the conditions evaluate to true.

Outputs

This flow logic has no outputs.

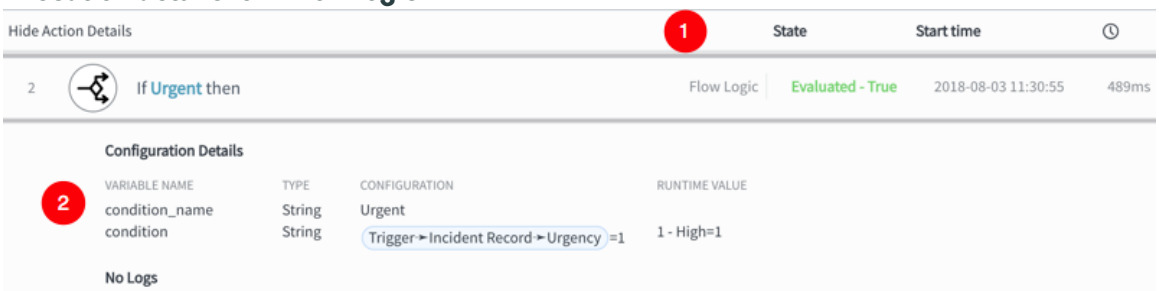
Example: Perform an action on if an incident has a high urgency

In this example, the action is triggered when the incident record has a high urgency value.



Execution details

Execution details for if flow logic



1. The header shows the state, start time, and runtime for the flow logic.
2. The Configuration Details section shows the details about the variables that are used by the flow, including the type, configuration, and runtime values for each variable. Use the *`condition`* variable to see if the branch conditions were met.

General guidelines

Use these general guidelines to create effective If flow logic blocks.

Avoid referring to data pill values outside of the Then branch

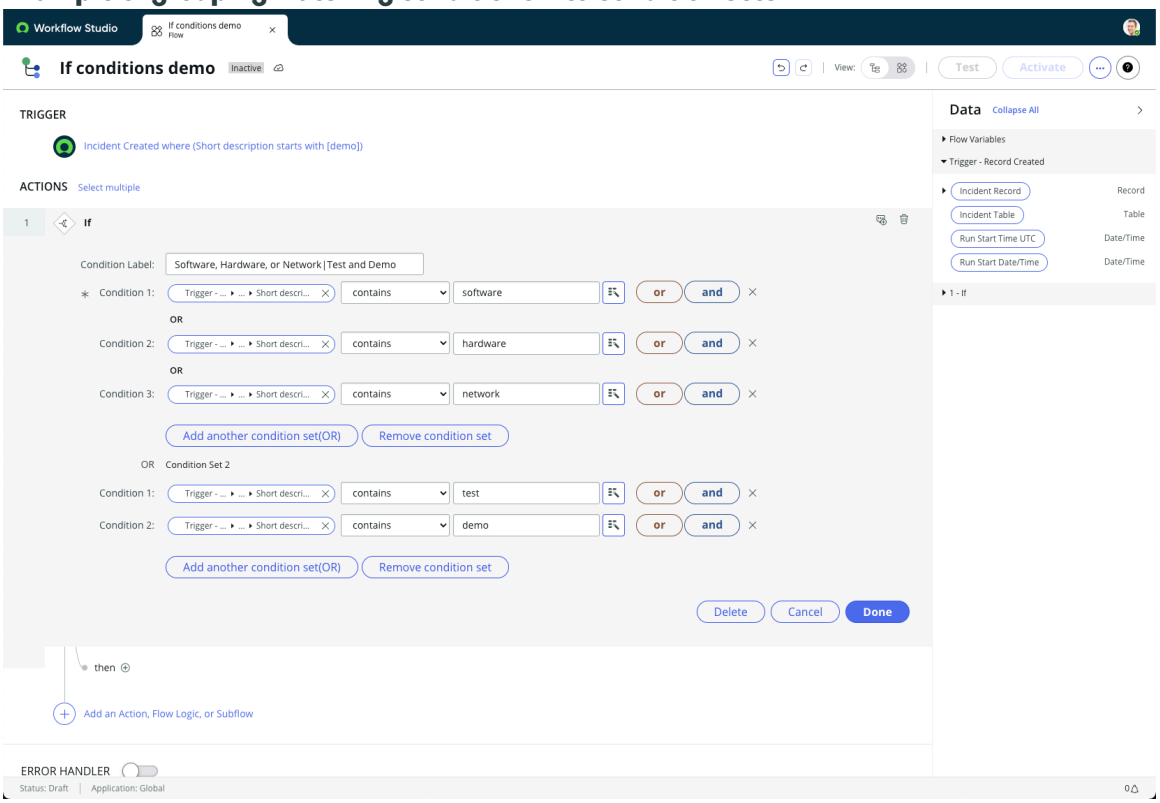
When you set a data pill value from inside a Then branch of If flow logic, the data pill value is only available to other actions in the same branch. Referencing a data pill value that was set inside a Then branch from outside of the flow logic branch produces a null value.

Group matching condition types in their own condition sets

Mixing conditions of different types for the same field values can produce unexpected results. For example, adding an AND condition to a group of multiple OR conditions for an incident short description can produce a situation where the

If condition never evaluates to true. Use condition groups to group similar condition types for the same field. For example, group all OR conditions for an incident short description in one condition set and group all AND conditions for an incident short description in another condition set.

Example of grouping matching conditions into condition sets

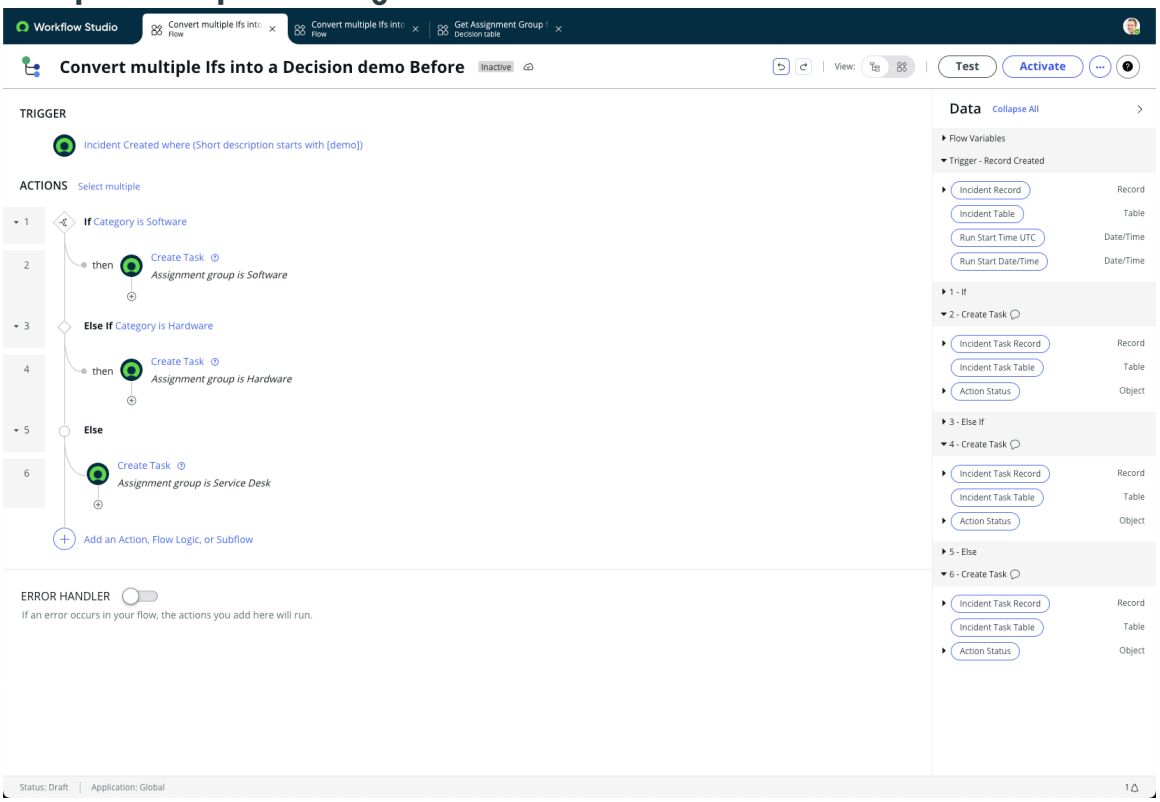


Replace multiple If flow logic blocks with a Make a decision flow logic block

Rather than create duplicate If flow logic blocks that only vary by their conditions, use a decision table to generate an answer. For example, suppose you want to use the incident category to set the assignment group of an incident task record. Rather than create a duplicate If flow logic block for each category value, use the Make a decision flow logic to provide an answer for the assignment group.

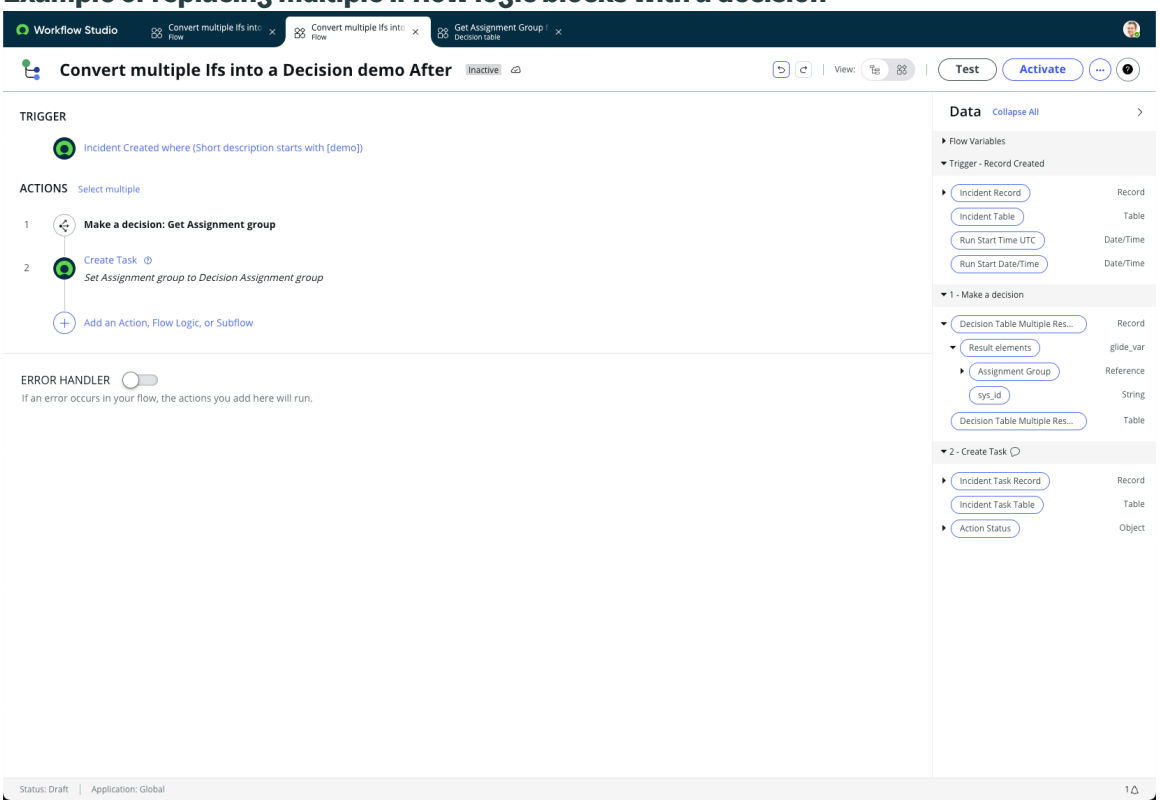
Here is an example flow that uses three If flow logic blocks that each create an incident task record. The only difference between the If flow logic blocks are the conditions of the incident category.

Example of multiple If flow logic blocks that do the same action



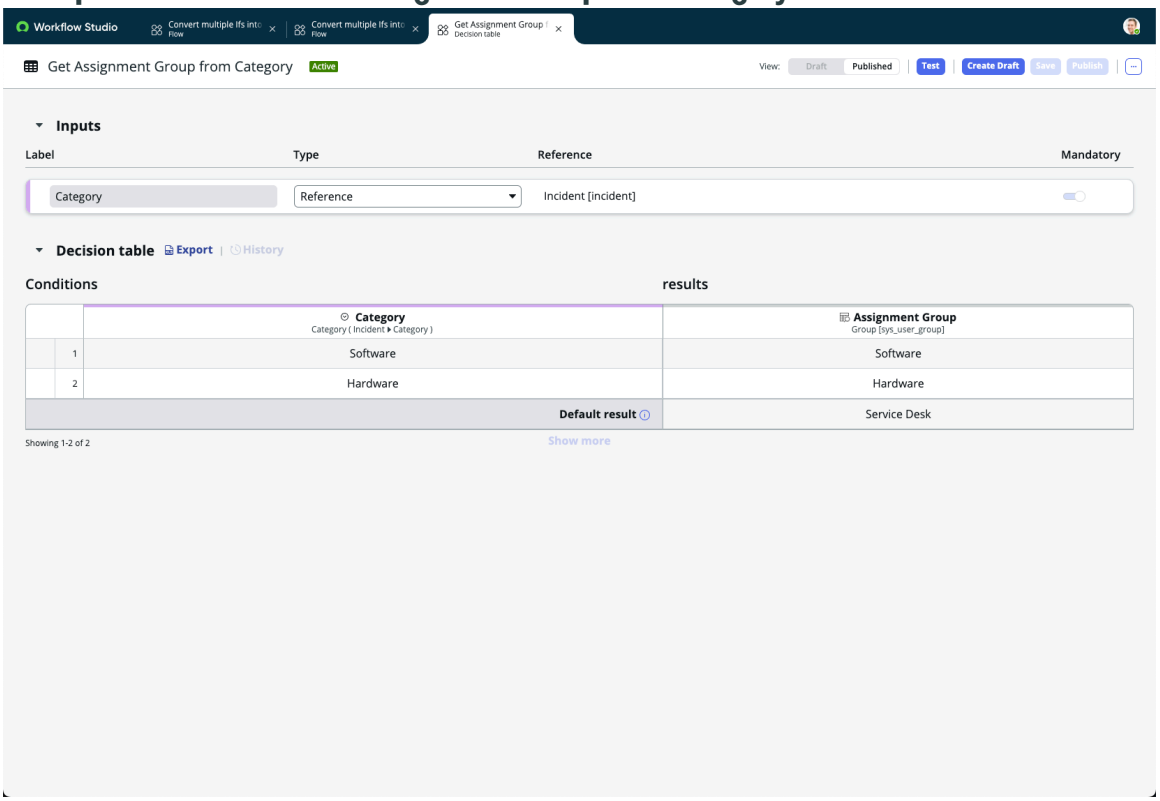
Here is an example flow that uses a single Make a decision flow logic block to determine the incident task assignment group from the incident category. The Create task action uses the output of the decision as an input.

Example of replacing multiple If flow logic blocks with a decision



Here is an example decision table that uses incident record values as an input. The Conditions column consists of two incident category values. The results column consists of the assignment group to use for each condition value.

Example decision table Get Assignment Group from Category



Make a decision flow logic

You can use the decision table branching logic in situations where multiple conditional paths are required, as an alternative to nested If, Else If, or Else flow logic. For example, if you want to determine what kind of car insurance you need, you can add inputs such as your age, accident history, and car model to the decision table to determine a level of insurance coverage. This logic can save you time and present a more readable format than nested if conditions or switch case statements.

The **Make a decision** flow logic requires that you create an external decision table for its branch paths. Each decision table answer produces a separate branch path within your flow. Decision tables accept any number of inputs and support any number of decisions. You can configure the Make a decision flow logic to return a single or multiple answers and to display the answers as branch paths or record data pills. For more information on decision tables, see Exploring decision tables.

Note:

- When the **Use Branches** check box is cleared, the Make a decision data pills in other flow components is supported only if there are no branches.
- The Make a decision data pill can be used in the answer branches only when the **Use Branches** check box is selected.
- When you set a data pill value from inside an answer branch of Make a decision flow logic, the data pill value is only available to other actions in the same branch. Referencing a data pill value that was set inside an answer branch from outside of the flow logic branch produces a null value.

Inputs

Input	Description
Decision Label	Descriptive label for the decision that you want to make. For example, you can create the `Recommended Insurance Policy` label if you want to determine the level of insurance coverage that you need. This value overrides the default action label.
Decision Table	Reference to a decision table [sys_decision] record. This record provides the decision input answers that are available to the flow. To create a new decision table, select the create new record icon (+).
Execution	Decision answers you want the flow to run. • **First decision that matches**: Run only the first matching decision answer. This option produces these outputs. ○ Answer record ○ Answer table • **Run all decisions that match**: Run all matching decision answers. This option produces these outputs. ○ Answer table ○ Ordered IDs ○ Answer records ○ Count **Note:** Set the **Use Branches** option to specify how your flow displays matching decision answers.
Use Branches	Option to display each possible decision answer in its own branch flow logic block. Use the branch flow logic block to specify what content to run where the condition matches a specific decision table answer. Each branch flow logic block is equivalent to an If flow logic block for each answer. Disable branches when you want to return one or more answer records instead of branch flow logic blocks.

Input	Description
	Warning: When you clear and confirm the **Use Branches** option, your flow removes the flow logic blocks for each branch and removes the **Include Otherwise** check box. Reselecting the **Use Branches** check box does not restore any removed branch flow logic blocks.
Include Otherwise	Option to add the **otherwise** branch to the list of available answers. This option is only available when **Use Branches** is selected. You can use this branch to specify the actions and subflows to run when the decision table does not generate an answer. An Otherwise branch is equivalent to an Else flow logic block. **Note:** If you specify a decision table default answer, this branch will never run because the decision table always selects an answer.
Decision table inputs	List of Decision Input [sys_decision_input] records that are associated with your decision table. Your flow displays a separate input for each record. For example, if you have decision inputs for `Units Ordered` and `Location of Sale`, an input displays for each record.

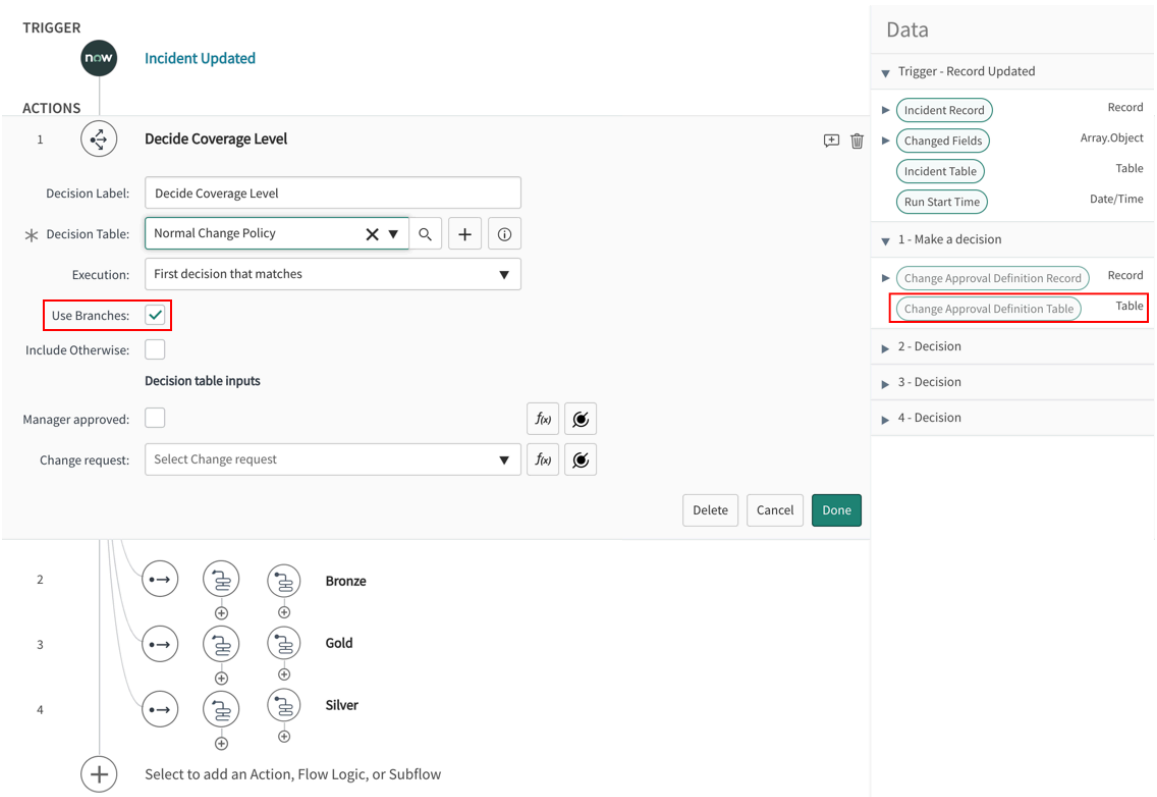
Outputs

Field	Description	Data Type
Answer table	Table containing the answer records. Each decision [sys_decision_question] table record refers to a matching answer record.	Table Name
Ordered IDs	List of matching answer record sys_id values generated by the decision table. Your flow only generates this output when the **Use Branches** option is false and the **Execution** option is **Run all decisions that match**. You can use this output as the input for a For Each flow logic block or a Look Up Record action. Your flow sorts the list by the Order value listed in the Decision [sys_decision_question] table.	List

Field	Description	Data Type
Answer records	Answer records returned by the decision table. Returns a single record when **Execution** is **First decision that matches**. Returns a list of records when Execution is **Run all decisions that match**.	Record or Records
Count	The number of answer records returned by the decision table. Only displayed when Execution is **Run all decisions that match**.	Integer

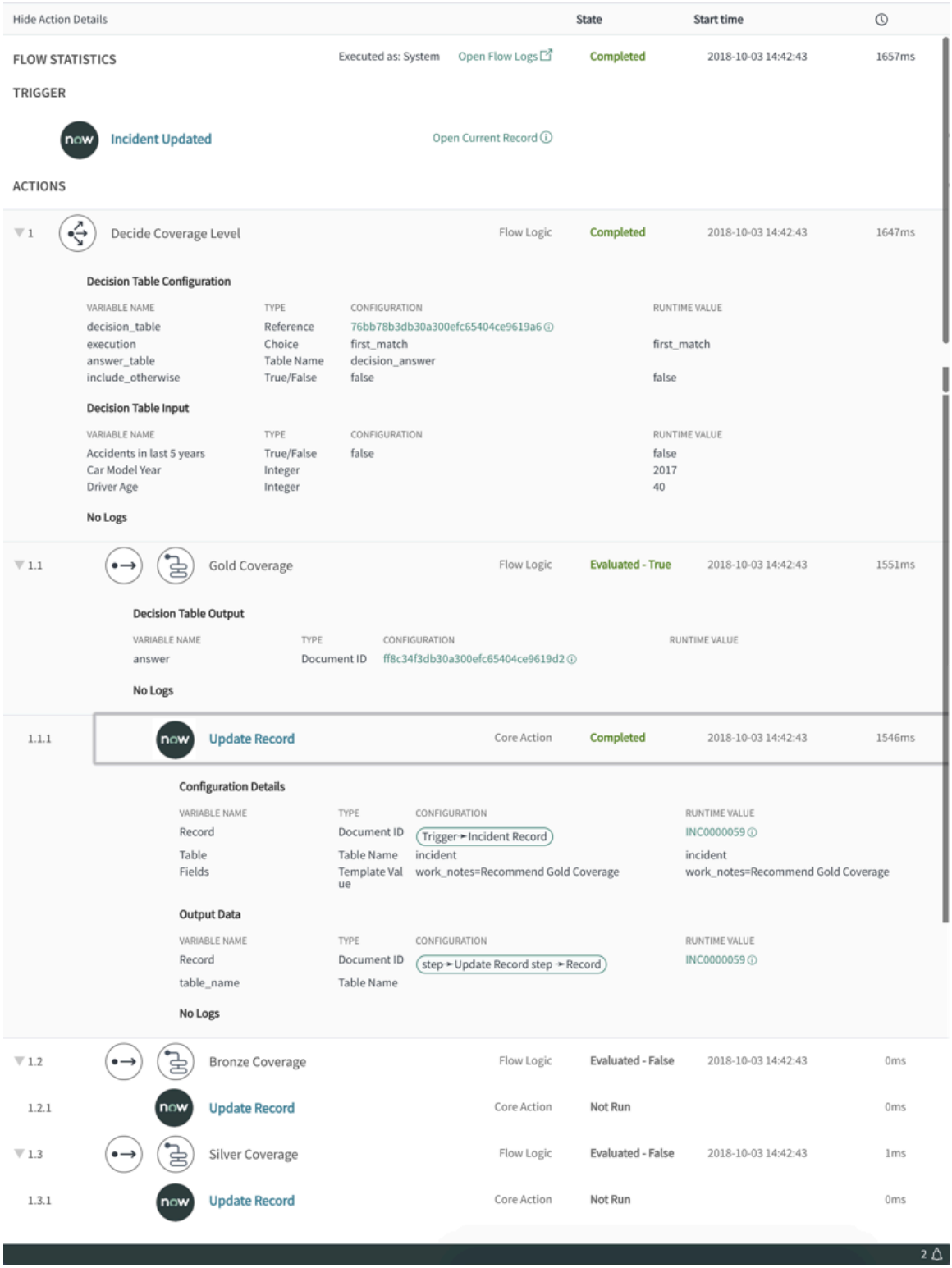
Example: Use make a decision flow logic to determine insurance coverage

In this example, the flow uses a decision from the **Insurance Coverage** decision table, which an administrator had configured to determine the insurance coverage that was based on three inputs. The flow displays all the inputs that were used by the decision table. These inputs can be entered manually, or by dragging data pills into the inputs from the data panel on the right side of the screen. Below this section, the branches for each answer are shown in the decision table.



Execution details

Make a decision flow execution details



The **Flow execution details** tab provides runtime information about the flow logic.

1. The header shows the state, start time, and runtime for the flow logic.
2. The action shows details about the decision table configuration and inputs.

3. Each possible answer for the decision table is represented as a branch. The state field indicates whether the branch was evaluated and the evaluation result. This section also displays details about the actions that are taken within a branch. Branches that evaluate to true are highlighted in green.

Set Flow Variables flow logic

Assign a value to one or more flow variables, which store flow data as data pills. Access flow variable values by referring to their data pill.

Important:
This flow logic sets values for flow variables that have already been created. For instructions on creating flow variables, see Create a flow variable.

Inputs

Field	Description
Name	Name of the variable. Select from the list of variables available for the flow.
Data	Value for the variable. Enter a string value, input a script, or use a data pill. Variable values can reference any data pill from earlier in the flow, including other variables. If you set variable values by reference to other data pills, you must maintain the order of the variable assignments. The referenced value must always come before the variable that uses the referenced value. Changing the order may produce null values. To assign an empty value, leave this field empty. **Note:** Flow variable values are set in the order in which they're assigned from top to bottom. If you set the value of the same variable multiple times, the flow only uses the last value set. To enter a script, select the **Toggle scripting on for [variable]** icon. Enter your script in the script editor. For more information about inline scripting, see Inline scripts.

Outputs

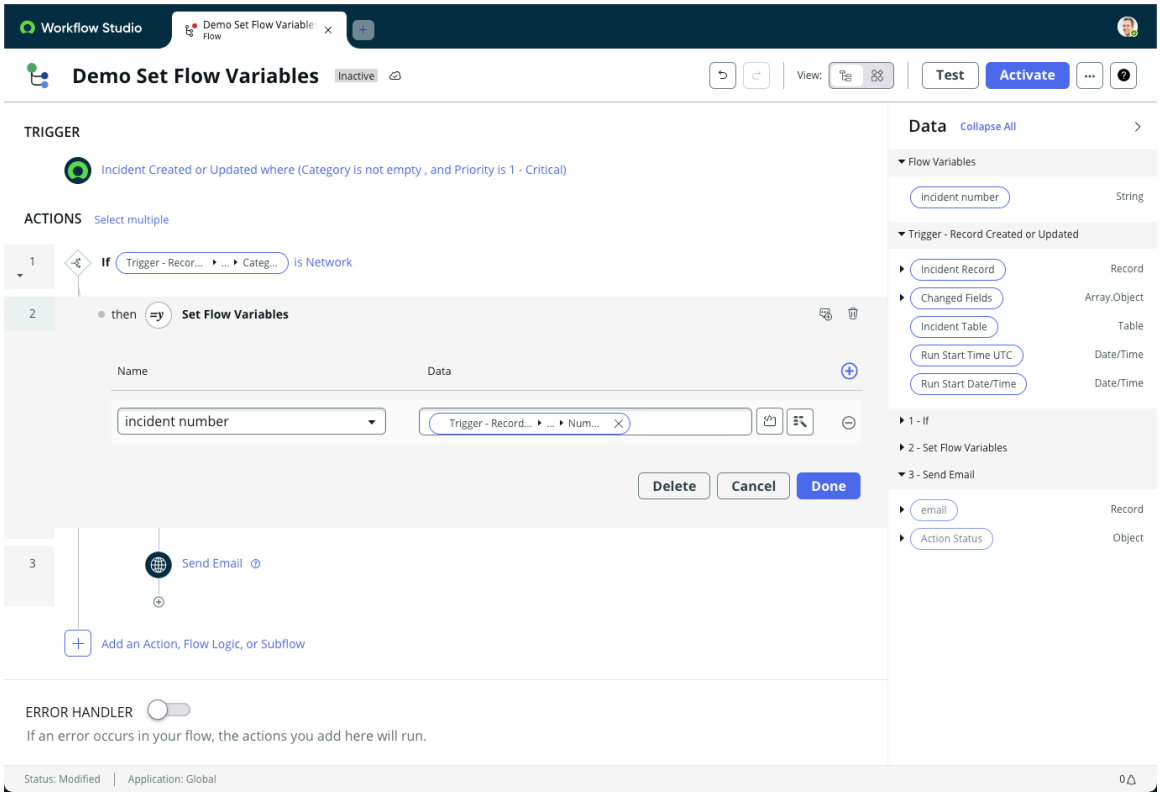
This flow logic produces no outputs but does change the value of flow variables.

Usage

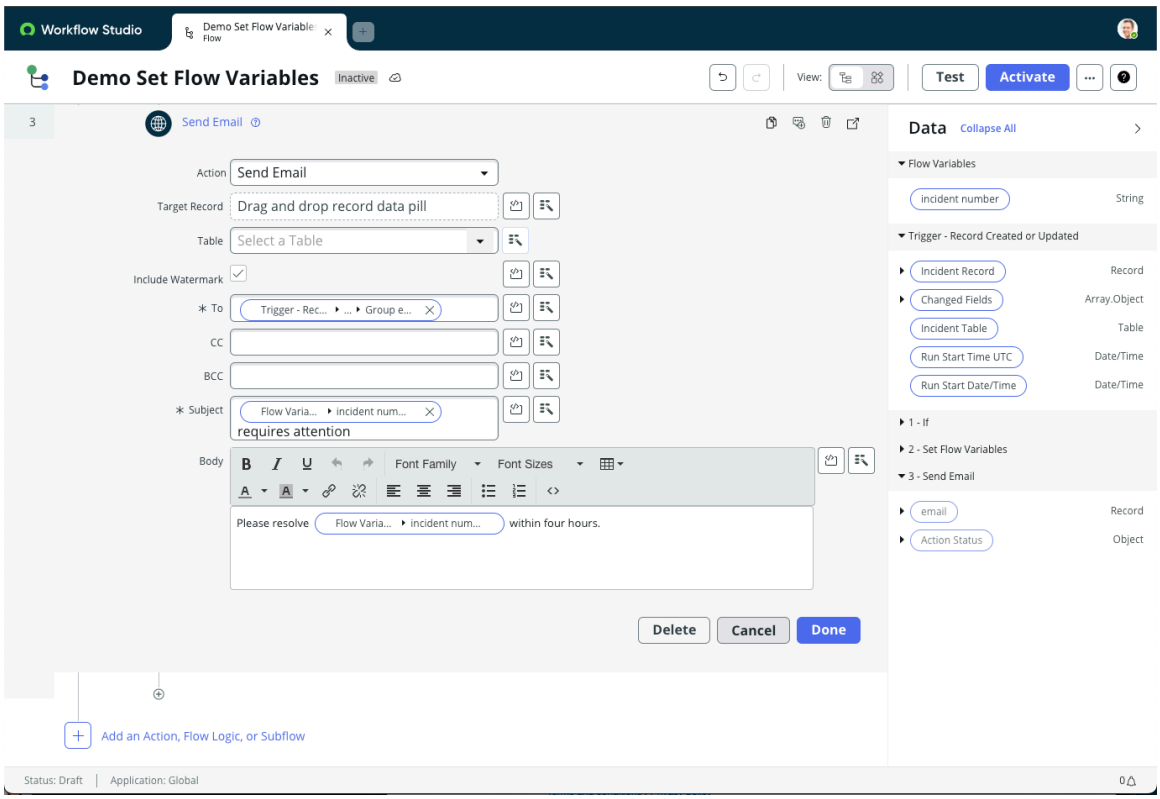
Flow variables store flow data as data pills of a specific data type. You can access flow variable data pills from the Flow Variables section of the Data pane. To use a flow variable value, select the data pill from the Data pane or the pill picker just as you would any other data pill.

Example: Set the incident number variable value to a flow data pill value

In this example, the flow checks the category of an incident record. If the category is network, a flow variable is used to store the record number.



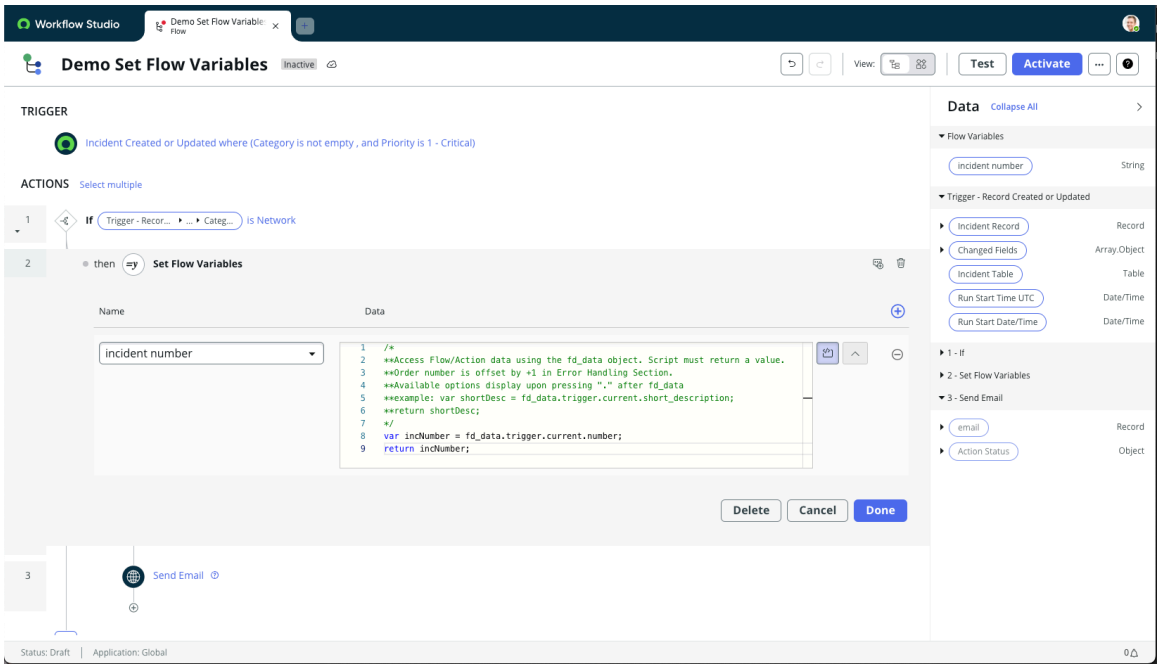
Later in the flow, the Send Email action uses the incident number flow variable as part of the email subject and body.



Example: Set the incident number variable value using a script

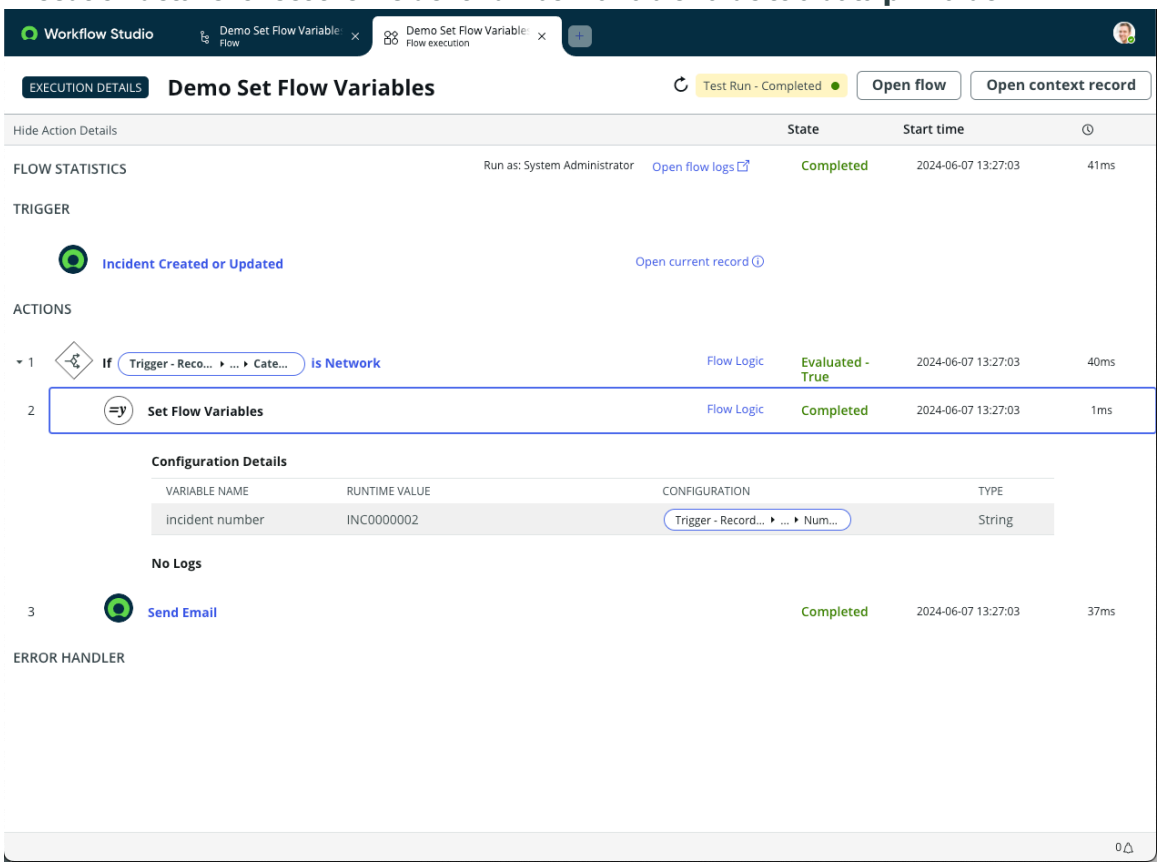
In this example, the flow checks the category of an incident record. If the category is network, a flow variable is used to store the record number. In this example, the flow variable is set from a script rather than a data pill value.

```
/*
**Access Flow/Action data using the fd_data object. Script must
 return a value.
**Order number is offset by +1 in Error Handling Section.
**Available options display upon pressing "." after fd_data
**example: var shortDesc =
 fd_data.trigger.current.short_description;
**return shortDesc;
*/
var incNumber = fd_data.trigger.current.number;
return incNumber;
```

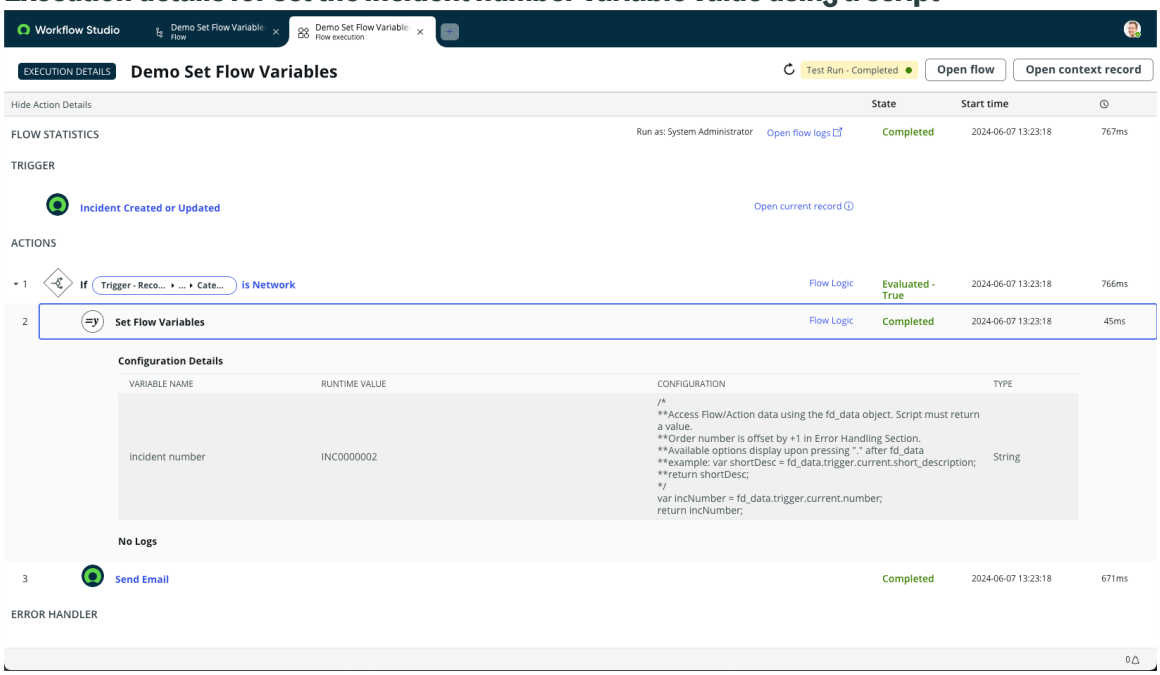


Execution details

Execution details for set the incident number variable value to a data pill value



Execution details for set the incident number variable value using a script



Skip Iteration flow logic

Skip the current iteration of a flow logic loop when the conditions of an If flow logic are met. Continue running the flow logic loop with the next item in the list. This flow logic is also known as continue.

Valid Skip Iteration placement

You can only add Skip Iteration flow logic within certain portions of a flow. The Skip Iteration flow logic must be within a branch of a parent looping flow logic block such as For Each flow logic or Do the following until flow logic. In addition, the flow logic loop must define a set of conditions to skip an iteration. You can only add a Skip Iteration flow logic within a Then branch.

- Then branch of an If flow logic block
- Then branch of an Else If flow logic block
- Then branch of an Else flow logic block

Inputs

This flow logic has no inputs.

Outputs

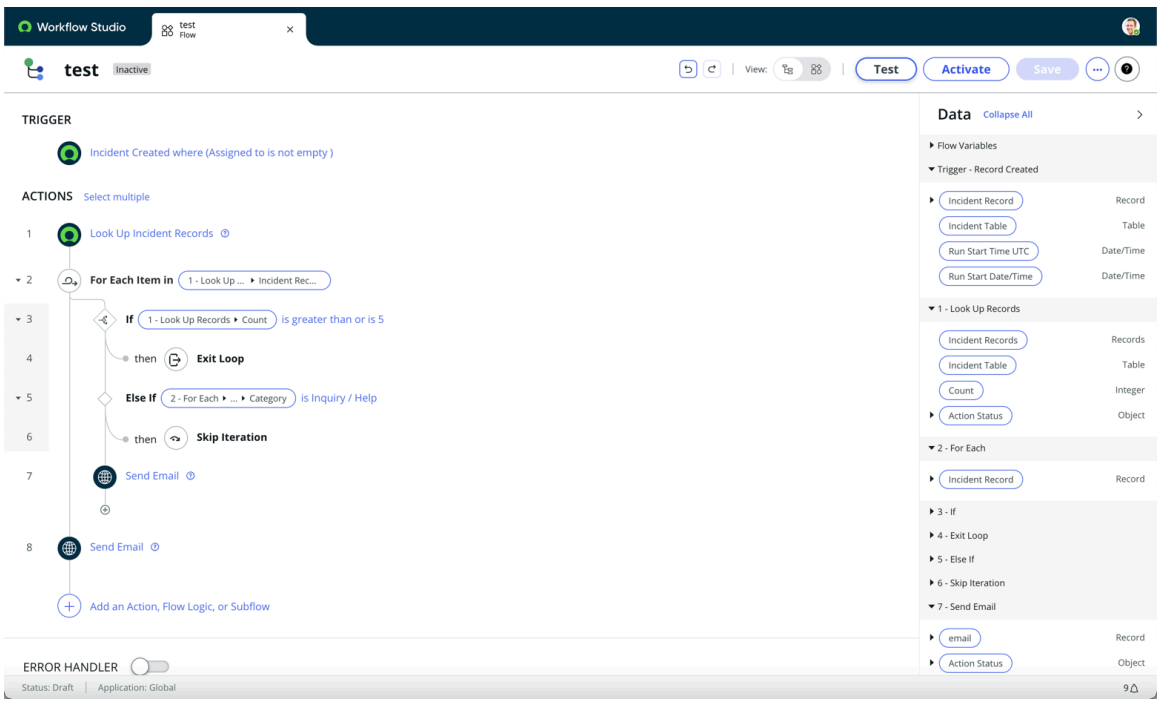
This flow logic has no outputs.

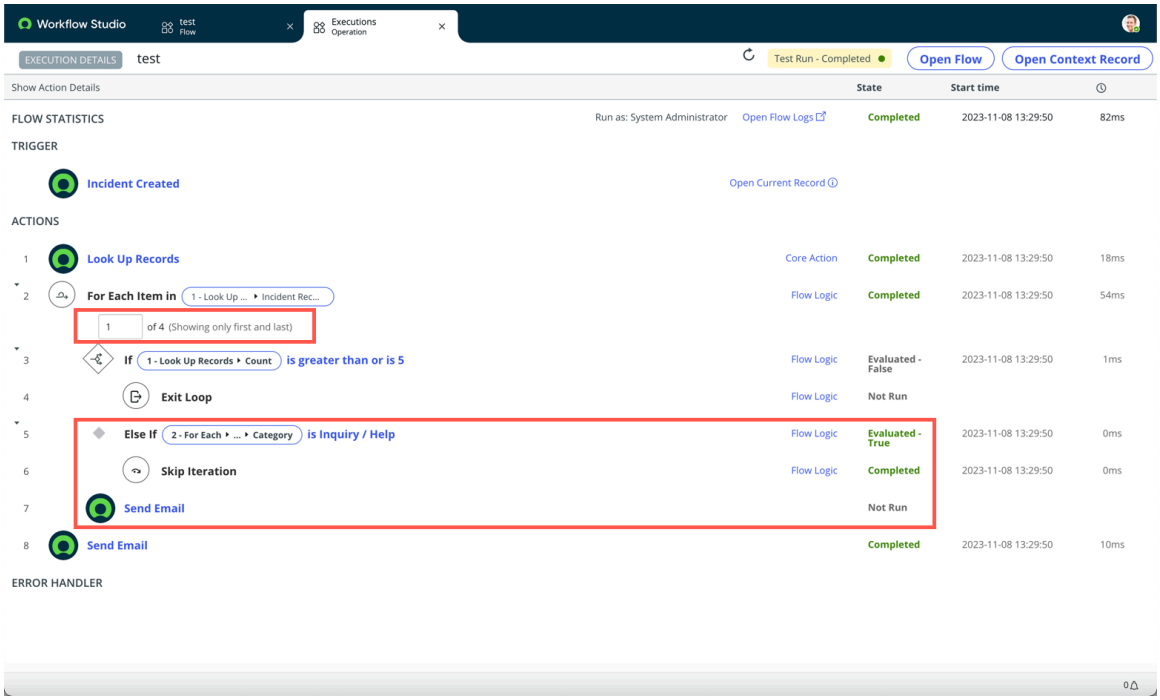
Execution Details

When a flow skips an iteration, the Skip Iteration flow logic has a state of Completed for the item that was skipped. Any following steps in the same For Each or Do the following until flow logic loop have a status of Not Run for the skipped item.

Example: Skip iteration when an incident is an inquiry category

In this example, a flow generates a list of incidents assigned to a user. For each incident that is assigned to the user, the flow sends an email. If the current incident record is in the Inquiry/Help category, then the flow skips the current item. The flow continues with the next incident record in the For Each flow logic loop.





In this example, the first item is an incident in the Inquiry category, which meets the skip iteration conditions. The flow does not run the Send Email action for this iteration.

Try flow logic

Allow a flow to continue running when an error occurs within a flow logic block. Run a sequence of actions in response to errors within the flow logic block.

Inputs

Try flow logic does not have field inputs. Instead, it displays a plus (⊕) icon that enables you to add a sequence of actions, flow logic, or subflows to attempt to run and evaluate for errors. You can add multiple items to a Try block. When an error occurs, the flow runs the sequence of actions, flow logic, or subflows within the error section of the flow logic block.

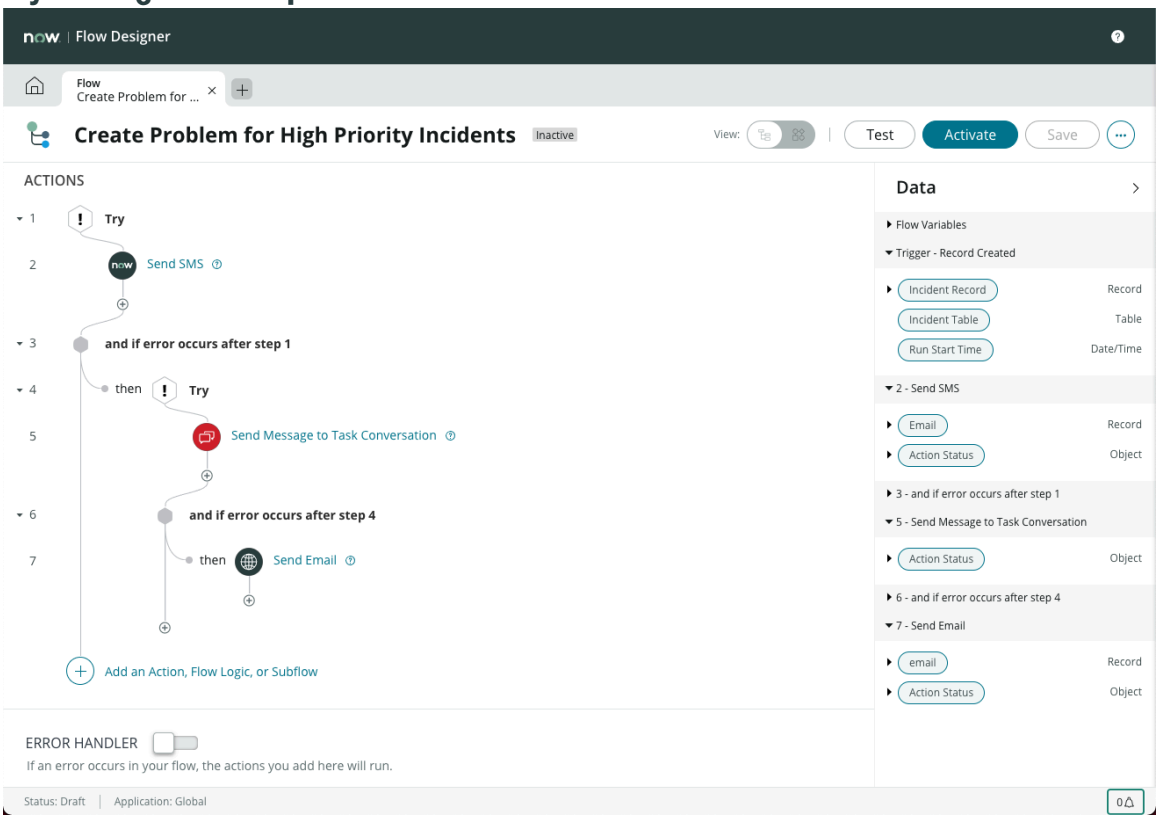
Outputs

This flow logic has no outputs, but the actions, flow logic, and subflows within the Try block may have outputs. When an error occurs within the Try flow logic block, the failing action returns an action status of **Completed (error caught)**. You can use this action status to determine which item failed and build conditional logic in the Catch block. For example, if a Create record action in the Try block has a status of Completed (Error caught), then run a Create record action in the Catch block that has different values.

Example: Create a Problem record for critical priority incidents

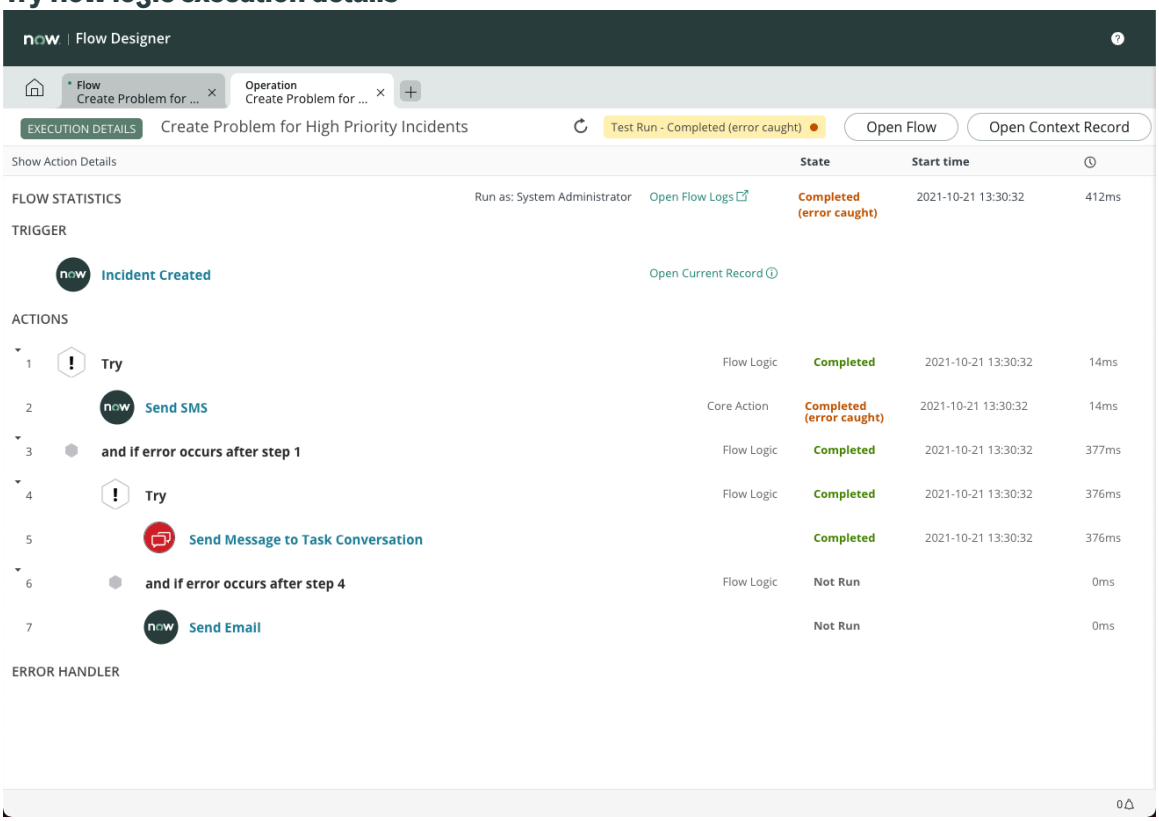
In this example, the flow attempts to send one of three possible notifications when a critical priority network incident is created. The flow first tries to send an SMS message, and if that fails, it tries to send a Connect message. If the Connect message fails, it sends an email.

Try flow logic for multiple notification methods



Execution details

Try flow logic execution details



Flow and action error handling resources

For more information about using error handling in actions and flows, see the ServiceNow® Community post Flow and Action Error Handling Overview: Why and how to test for errors - Worfklow Automation CoE.

- Flow and Action Error Handling Level 1: Retry and Action Error Evaluation - Workflow Automation CoE
- Flow and Action Error Handling Level 2: Flow Logic - Workflow Automation CoE
- Flow and Action Error Handling Level 3: Flow Error Handling - Workflow Automation CoE
- Flow and Action Error Handling Level 4: Good Practices and Summary - Workflow Automation CoE

Wait for a duration flow logic

Use this flow logic to give your users time to act during automated processes or to wait for a specific date and time to complete actions.

Inputs

Input	Description
Duration Type	• **Explicit Duration**: Wait for a specific time period, such as 5 minutes. • **Relative Duration**: Wait for a specific time period from a selected Duration data pill or date/time value, such as 5 minutes after the flow start. • **Percentage Duration**: Type to specify a certain percentage of time duration between the start of the flow logic and specified end time. **Note:** The percentage value must be from **0** through **100** only.
Wait for	Set this value manually or select a Duration data pill from the data pill picker ( ). For example, use a Look Up Record action to select an SLA Definition record and return the value of the **Duration** field. • **Explicit Duration**: Wait duration in hours, minutes, and seconds. • **Relative Duration**: Wait duration in hours, minutes, and seconds before or after a specific time. Select **Relative Duration** to specify a wait duration from a specific date. **Note:** Past dates don't affect the wait duration. You can enter a wait value of up to 999 hours. **Note:** The actual wait duration can vary due to the instance processing time. The flow always waits for the time that you specify for this field, but other work in the queue may add to the wait time.
Wait for Percentage	Wait duration as a percentage of the time period between the start of flow logic and specified end time. If you select a past date for the end time, the wait duration

Input	Description
	is set to **0**. This field appears when **Percentage Duration** is selected from the Duration Type list.
During the following schedule	Select the schedule used to calculate the **Scheduled End date/time** value from the selected wait duration. For example, waiting for a 10-hour duration as part of an 8-5 weekdays schedule causes the flow to wait for one or more business days. If you leave this field blank, the timer runs without a schedule. For information on creating schedules, see Define a schedule.

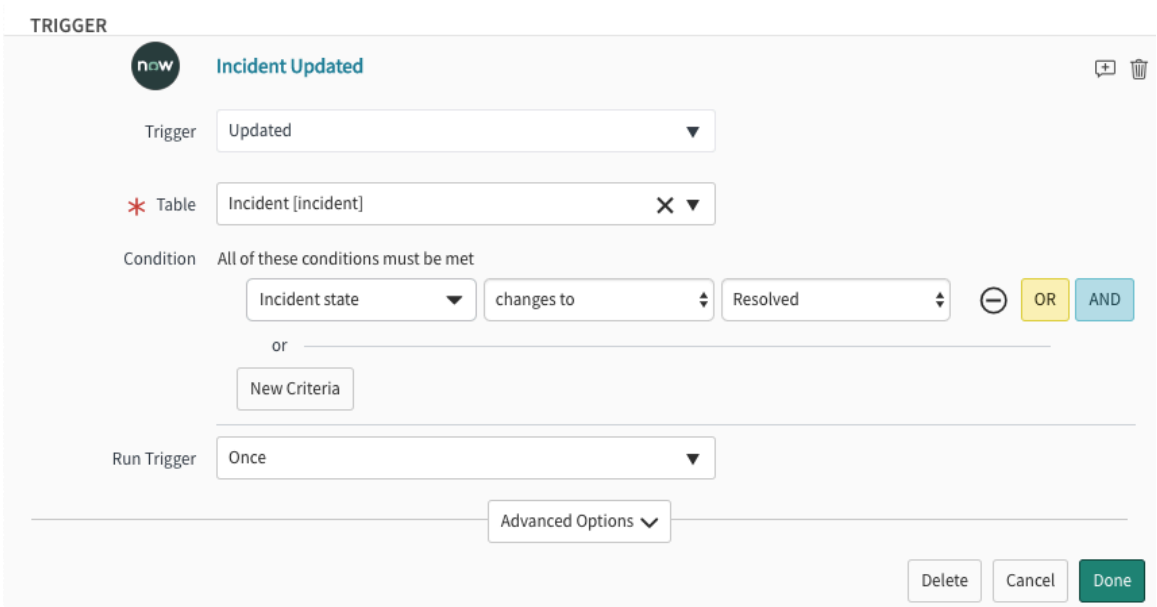
Outputs

Output	Description
Duration	Total time that the flow ran in milliseconds. You can drag this data pill into the duration fields.
Date/ time	Date/time that the flow completed. You can drag this data pill into the date/time fields.

Example: Close an incident if it has been in the resolved state for 10 days

In this example, a flow starts when the incident state changes to Resolved.

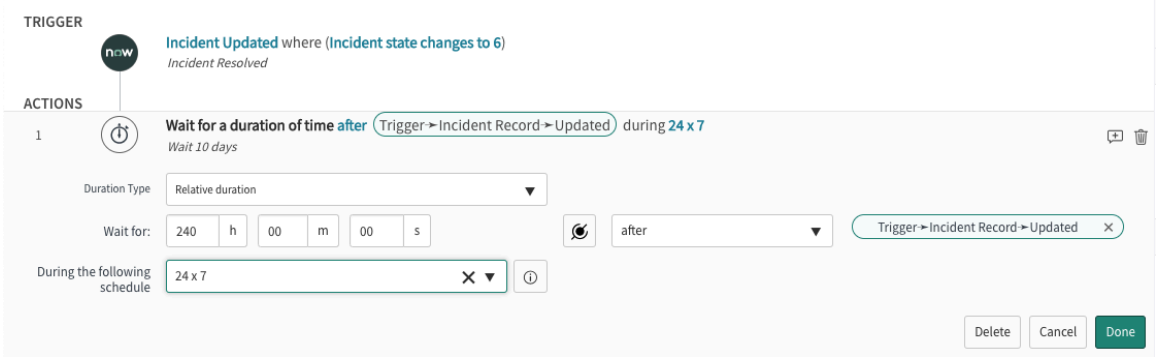
Flow trigger



Example: Wait 10 days after the last update to a record

In this example, the flows waits for 10 days after the incident record has been resolved.

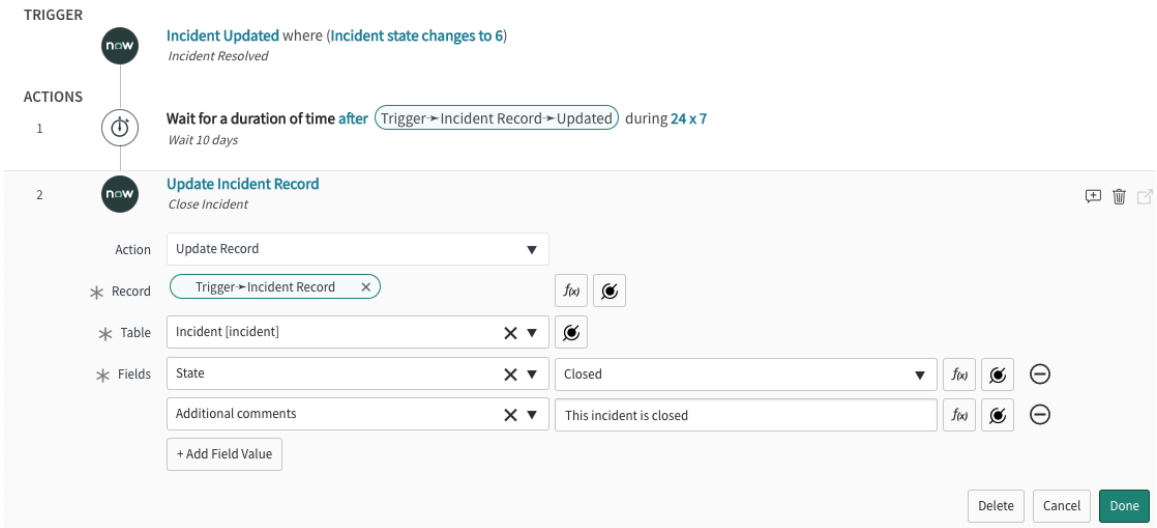
Wait for Duration flow logic



Example: Update a record after 10 days

In this example, the flows closes the incident record 10 days after it was resolved.

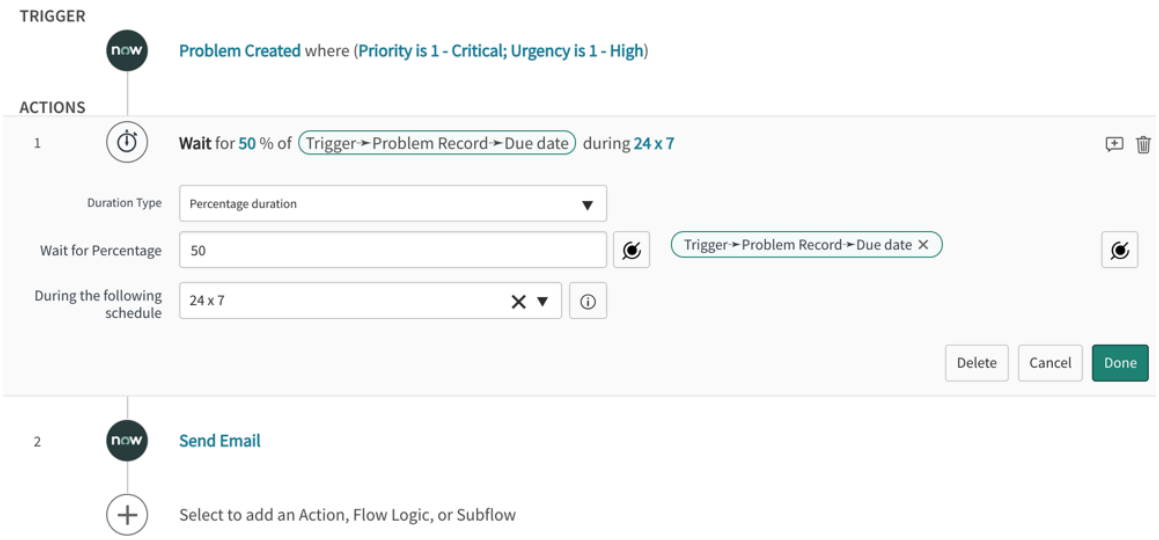
Action used to close the incident



Example: Wait for a duration of 50% of the time between the start of the flow logic and the due date

In this example, the flows send a notification email to the relevant manager when a critical problem is created and 50% of the time between the problem record creation and the problem due date has lapsed.

Wait for percentage time duration flow logic



Relative Duration type

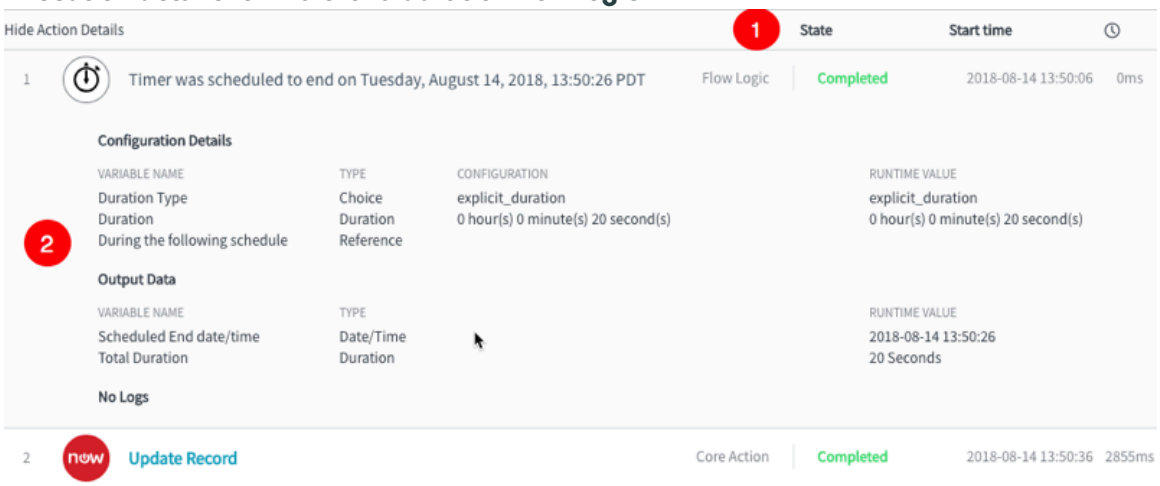
When the duration type is **Relative Duration**, the flow logic first evaluates the relative date and time, the schedule, and finally, the duration. Schedules, dates, and times set in the past don't affect the wait duration. This table provides examples of how the flow processes the wait duration in these scenarios.

Duration Setting	Relative Date/Time	Schedule	Effect
Set to 0.	None	None	Duration ends immediately.
Greater than 0.	Past date	None	Duration ends immediately.
Greater than 0.	Future date	None	Flow waits for the date/time, and then waits for the duration.
Greater than 0.	Past date	Future date	Flow waits for schedule, and then waits for the duration.
Greater than 0.	Future date	Past date	Flow waits for the date/time, and then waits for the duration.
Greater than 0.	Future date	Future date	Flow waits for the future date, then for the schedule, and then for the duration.

The timer waits for the next instance of a selected schedule. For example, if you set a schedule for Monday through Friday from 8 a.m. to 5 p.m., and the timer is initiated on Saturday, the timer waits until Monday at 8 a.m. before starting.

Execution details

Execution details for Wait for a duration flow logic



1. The header shows the state, start time, and runtime for the flow logic.

   Note:
   The runtime value in the header only includes the time that it takes to execute the flow logic and doesn't include the wait duration that is specified in the flow.

2. The Configuration Details section shows details about the variables that are used by the flow, including the type, configuration, and runtime values for each variable.

Workflow Studio flow integrations

Expand the capabilities of Workflow Studio flows with additional subscriptions and spokes.

ServiceNow AI Platform integrations

Subscribing to ServiceNow AI Platform features and applications adds Workflow Studio flow content.

Integration	Content available
App Engine Studio	• Build flows from App Engine Studio. • Create flows from flow templates.
Integration Hub	• Access integration steps from Workflow Studio. • Install integration spokes.
Process Automation Designer	• Call flows and actions from Workflow Studio. • Gather process data with flows.
Robotic Process Automation (RPA) Hub	Build flows to run robots on Microsoft Windows systems.

Workflow Studio spokes

Add application-specific content to Workflow Studio by installing spokes.

Spokes

Add application-specific content to Workflow Studio by installing spokes.

A spoke is a scoped application containing Workflow Studio content dedicated to a particular application or record type. For example, the **ITSM Spoke** contains actions for managing Task records such as the **Create Task** action. Spokes are activated when their parent application is activated. For example, the **ITSM Spoke** is activated when the Incident, Problem, and Change applications are activated. Creating a spoke requires familiarity with application development as developers must add Workflow Studio content to a scoped application.

Default spokes available

Spoke	Description	Plugin	Included with
Benchmarks Spoke	Provides read-only actions for the read-only Benchmark Recommendation Evaluator flow.	[com.sn_bm_client.spoke]	Benchmarks application.
Connect spoke	Provides actions to automate the creation of conversations, to add users to a conversation, and to send messages to a conversation. These actions work with Connect API version 3 and later.	[com.glide.connect_v3.spoke]	ServiceNow AI Platform
Customer Service Spoke	Provides actions for flow designers to use when creating Customer Service Management business processes.	[com.snc.customer_service.spoke]	Customer Service Management application
External Related Files spoke	The External Related Files spoke stores information about files in third-party systems and helps you manage the information.	[com.sn.external.files]	ServiceNow AI Platform
Field Service Spoke	Provides actions for flow designers to use when creating Field Service Management business processes.	[com.snc.field_service.spoke]	Field Service Management application
ITSM spoke	Provides flow and actions associated with ITSM. Requires	[com.snc.itsm.spoke]	IT Service Management application

Default spokes available (continued)

Spoke	Description	Plugin	Included with
	the ITSM application suite.		
Machine Learning solutions for Flow Designer	Provides actions to make predictions from trained Predictive Intelligence solutions.	[com.snc.ml_flowdesigner]	Predictive Intelligence
Robotic Process Automation (RPA) Spoke	Provides RPA actions for flow designers to assign users to attended automation process, add work queue items to queue, update work items, fetch process jobs and execution status of a specific process job, trigger a specific bot process, and unassign users from attended automation process.	[com.sn_rpa_foundation]	Robotic Process Automation (RPA) Hub
Security Operations spoke	Provides Security Operations actions for flow designers to manage Security Incident Response flow templates.	[com.snc.secops.spoke]	Security Operations application
Visual Task Board (VTB) Spoke	Provides VTB actions for flow designers to manage the boards, lanes, cards, board members, and assignees.	[com.glide.ui.vtb.ah]	ServiceNow AI Platform

Additional spokes are available with an Integration Hub subscription. To see a list of Integration Hub spokes, see Integration Hub available\n spokes. For more information about requesting an Integration Hub subscription, see Request Integration Hub.

Benchmarks Spoke

Provides read-only actions for the read-only Benchmark Recommendation Evaluator flow.

The Benchmarks Spoke is designed for the Recommendations feature of the Benchmarks application.

Action	Description	Action Inputs	Action Outputs
Create Recommendation Activity Records	Create or update recommendation activity records.	Recommendation	N/A

Action	Description	Action Inputs	Action Outputs
Delete Recommendation Evaluations	Delete recommendation evaluations for the specified month.	Activity record	N/A
Evaluate Recommendation Condition	Evaluate the conditions and script specified for the recommendation.	• Record count • Threshold • Direction • Recommendation • Activity record	• Result • Score

Connect spoke

Provides actions to automate the creation of conversations, to add users to a conversation, and to send messages to a conversation. These actions work with Connect API version 3 and later.

Action	Description
Add Group Users to Task Conversation	Create a task conversation, and add all users of a group to it.
Add User to Task Conversation	Create a task conversation, and add a user to it.
Send Message to Task Conversation	Send a message to all users of a task conversation.

Customer Service Spoke

Provides actions for flow designers to use when creating Customer Service Management business processes. Requires the Customer Service Management [com.sn_customerservice] plugin.

Action	Description
Get Case	Retrieve a case record using the case number. If multiple records are found, only the first record is returned.
Create Case	Create a case using one or more attributes. This action mimics the structure of the Case table (sn_customerservice_case) and exposes the fields present on the Case table.
Create Quick Case	Create a case using the customer, description, channel, priority, and category attributes.
Create Task on Case	Create a case task and optionally associate it with a case.
Update Case	Update a case by providing the case reference and the fields that you want to update.
Assign Case	Assign a case using matching rules. To use this action, you must first define the matching

Action	Description
	rules that match cases with resources (assignment groups, agents).
Escalate Case	Request case escalation. This action does not automatically approve escalation. Approval is based on the selected escalation template.
Escalate Account	Request account escalation. This action does not automatically approve escalation. Approval is based on the selected escalation template.
Add Work Note to Task	Add a work note to a task or to task extended objects (for example, a case or case task).
Add Comment to Task	Add a comment to a task or to task extended objects (for example, a case or case task).

External Related Files spoke

The External Related Files spoke stores information about files in third-party systems and helps you manage the information.

External Related Files spoke tables

Table	Description
External Provider [sn_ext_files_spoke_provider]	Stores information about the external provider. For example, Box or DocuSign services.
External Related Files [sn_external_related_files]	Stores metadata information about the files in third-party systems. This table is extensible. **Note:** If you extend the table, ensure that you perform the data separation for the scoped applications • To store metadata information of files in a specific third-party system, create a table with a column that contains a reference field to the External Related Files table. For more information about reference fields, see Reference field type. • To establish a relationship between a specific ServiceNow table and External Related Files table, Create defined related lists.

External Related Files spoke actions

Action	Description
Create External File Record	Creates a record in the External Related Files table.
Update External File Record	Updates a record in the External Related Files table.
Delete External File Record	Deletes a record in the External Related Files table.

External Related Files spoke user roles

Role	Description
sn_ext_files_spoke.doc_reader	Read records in the External Related Files and External Provider tables.
sn_ext_files_spoke.file_admin	• Read, update, and delete records in the External Related Files table. • Read records in the External Provider table.
sn_ext_files_spoke.provider_admin	Read, update, and delete records in the External Provider table.

Field Service Spoke

Provides actions for flow designers to use when creating Field Service Management business processes.

Action	Description
Get Work Order	Retrieve a work order record using the work order number. If multiple records are found, only the first record is returned.
Create Work Order	Create a work order and optionally associate it with a case.
Update Work Order	Update a work order by providing the work order reference and the fields that you want to update.
Get Work Order Task	Retrieve a work order task record using the work order task number. If multiple records are found, only the first record is returned.
Create Work Order Task	Create a work order task and optionally associate it with a work order.
Update Work Order Task	Update a work order task by providing the work order task reference and the fields that you want to update.
Add Work Note to Task	Add a work note to a task or to task extended objects (for example, a work order or work order task).

ITSM spoke

Provides flow and actions associated with ITSM. Requires the ITSM application suite.

Action	Description
Add Comment	Adds a comment to a task record.
Add Worknote	Adds a work note to a task record.
Apply Change Approval Policy	Controls the approval process for a change request by creating user and group approvals

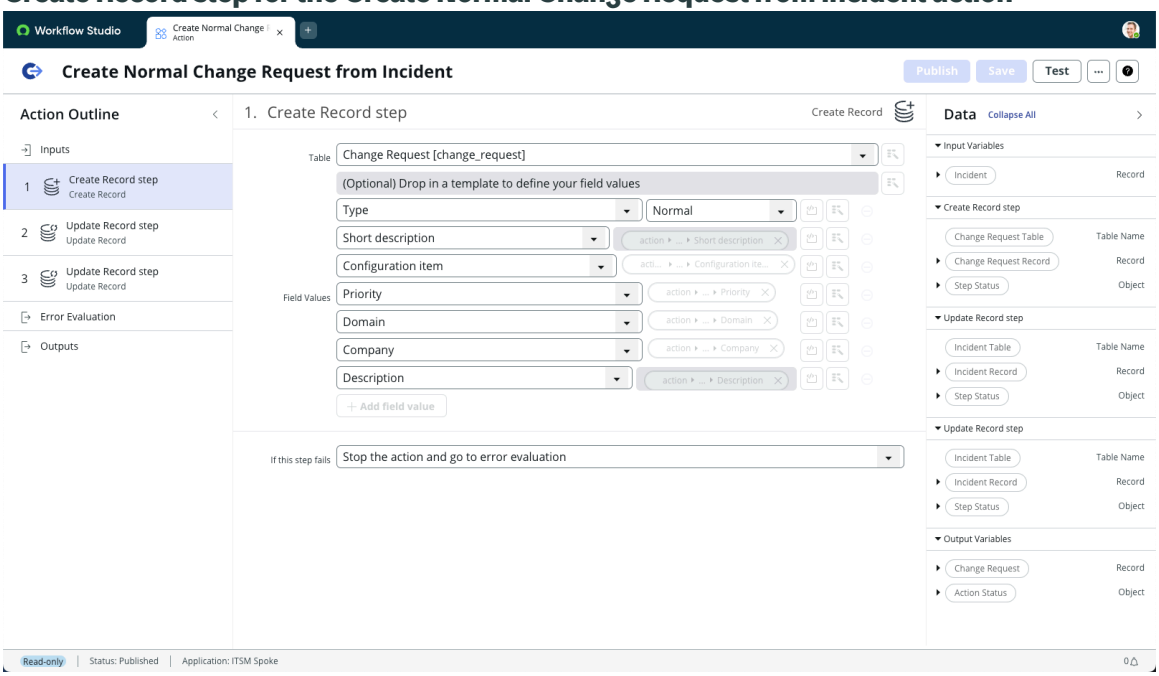
Action	Description
	according to a change approval policy record. Multiple actions can be used in a flow, where each action references the same or different Change Approval Policies.
Assign Incident to CI Support Group	Updates an incident record to assign it to the CI Support Group.
Cancel Change Tasks from Flow	Cancels all related pending and open change tasks that are created from Flow.
Check Change for User Approval	Checks if the specified user has already approved the change request.
Create Catalog Task on Request	Creates a Catalog Task record from a Request record.
Create Catalog Task on Request Item	Creates a Catalog Task record from a Request Item record.
Create Change Task on Change Request	Creates a Change Task record from a Change Request record.
Create Emergency Change Request	Creates a Change Request record of type Emergency.
Create Emergency Request from Incident	Creates a Change Request record of type Emergency from an Incident record.
Create Incident	Creates an Incident record.
Create Incident Task on Incident	Creates an Incident Task record from an Incident record.
Create Normal Change Request from Incident	Creates a Change Request record of type Normal from an Incident record.
Create Outage	Creates a cmdb_ci outage record for a configuration item. The **Task** field is populated only if the source is a task record. If the source is non-task record such as an alert record, the **Task** field is empty.
Create Problem from Incident	Creates a Problem record from an Incident record.
Create Request	Creates a Request record.
Create Standard Change Request	Creates a Change Request record of type Standard. For more information, see IT Service Management.
Create Task	Creates a child task record for a Task table record. For example, creates an Incident Task record for an Incident record.
Create Task Outage Relationship	Creates a Task Outage Relationship record where cmdb_ci outage record and task record are inputs to the action.
Create Standard Change Request from Incident	Creates a Change Request record of type Standard from an Incident record.

Action	Description
Disregard Change Approvals	Sets all related pending approvals to no longer required.
Update Assignee	Updates the Assigned to field of a Task table record.
Update Assignment Group	Updates the Assignment Group field of a Task table record.

Data mappings

Users with the admin, flow_designer, or action_designer roles can open and view the steps of ITSM Spoke actions. To view data mappings between ITSM data types, view the Create Record step of the ITSM Spoke action. Most ITSM Spoke data is stored in extensions of the Task [task] table, which means that many of the data types share common fields.

Create Record step for the Create Normal Change Request from Incident action



For example, the Create Normal Change Request from Incident action uses the common Task table fields for Short description, Configuration Item, Priority, Domain, Company, and Description to create a Change Request from an Incident.

Machine Learning solutions for Flow Designer

Provides actions to make predictions from trained Predictive Intelligence solutions.

Predictive Intelligence subscription

This spoke requires a subscription to Predictive Intelligence. For more information, see Install Predictive Intelligence.

Key features

Predictive Intelligence for Flow Designer provides four frameworks that you can use to create machine-learning solutions in your instance. Each framework delivers a different solution type for training the system to predict, recommend, and organize data outcomes.

- Classification framework
- Similarity framework
- Clustering framework
- Regression framework

Spoke requirements

- A sharedservice.worker user to train solutions
- A pre-trained solution for your Predictive Intelligence framework

Spoke dependencies

If you're having trouble installing the app, ensure that these dependent plugins are installed:

- Predictive Intelligence (com.glide.platform_ml) plugin
- Predictive Intelligence Reporting (com.glide.platform_ml_pa) plugin

Note:
Some of these plugins are licensable features and require appropriate licenses, if used outside the spoke implementation.

Spoke actions

Predictive Intelligence for Flow Designer provides actions to make predictions using existing models without having to write or maintain script. Available actions include:

Action	Description
Classification Batch Prediction	Obtain a predicted value from an active classification solution definition using multiple input records.
Classification Prediction	Obtain a predicted value from an active classification solution definition using a single input record.
PI Confidence Check	Compare an output value (number) from Predictive intelligence with a number specified by the user. For example: Compare the confidence value of a prediction with a specified value.
Regression Batch Prediction	Obtain a predicted value from an active regression solution definition using multiple input records.
Regression Prediction	Obtain a predicted value from an active regression solution definition using a single input record.
Similarity Prediction	Obtain similar records that exist in the table specified by the user in their similarity solution definition.

Spoke user roles

Predictive Intelligence for Flow Designer provides these user roles to control access to data:

User role	Description
ml_admin	Grants access to all Predictive Intelligence features

Related topics

Use Predictive Intelligence in Flow Designer with ML actions

Robotic Process Automation (RPA) Spoke

With Robotic Process Automation, your flow designers can use actions to assign and unassign users to and from an attended automation process, add work items to a queue, update work items, fetch process jobs, get the status of a process job, and trigger a bot process.

RPA actions

Action	Description	Action inputs	Action outputs
Add WorkItem to Queue Action	Adds WorkItem in a specified queue in the RPA Hub application and returns the WorkItem ID.	• **Queue Name**: Record • **Name**: String • **Content**: String • **Type**: String • **Priority**: String.Choice • **Stage**: String • **SLA**: Date/Time The maximum limit for **Content** field (Request Content and Response Content) is 8000 characters. If the **Is Work Item Name Sensitive** check box on the associated work queue form is selected, the data in the **Request Content** field is encrypted. Sensitive data with encryption is stored in the work item, so that it can be passed to the RPA Desktop Design Studio or robots.	• **Action Status**: Object • **WorkItemID**: GUID • **IsNameExists**: Boolean • **status**: String • **message**: String
Assign User to Attended Automation Process Action	Assigns a user to the attended automation process.	• **Process Name**: Record • **Username**: Record	• **Action Status**: Object • **status**: String • **Message**: String

RPA actions (continued)

Action	Description	Action inputs	Action outputs
Assign User to Attended Robot Action	Assigns a user to the attended robot.	**Username**: Record	• **Action Status**: Object • **status**: String • **Message**: String
Change Life Cycle Stage Status of a Bot Process Action	Changes the life cycle stage status of a bot process that is not retired. If a bot process is in **Build** life cycle stage status, if you have the required role, you can update it to **Published** state, but you cannot update it to **In Maintenance**. If a bot process is in **Published** life cycle stage status, if you have the required role, you can update it to **In Maintenance** state, and vice versa.	• **Process name**: Record • **Life cycle stage status**: Choice	• **result**: String • **message**: String • **Action Status**: Object
Fetch Created Jobs Action	Fetches the process jobs in RPA Hub.	• **Process Number**: String • **Robots**: Array[String]	• **Action Status**: Object • **Result**: String • **Jobs**: Array[String] • **startedRobots**: Array[String] • **pendingRobots**: Array[String]
Fetch Execution Status Action	Fetches the execution status of a specific process job in RPA Hub.	**Process Job Number**: String	• **Action Status**: Object • **ProcessJob**: Object ○ **Process Name**: String ○ **Robot Name**: String ○ **State**: String ○ **Started At**: Date/Time

RPA actions (continued)

Action	Description	Action inputs	Action outputs
			○ **Completed At**: Date/Time ○ **Message**: String • **status**: String • **message**: String
Fetch Work Item Status Action	Fetches the status of specific work queue items, asynchronously.	**Work Queue Item ID**: GUID	• **Action Status**: Object ○ **Code**: Integer ○ **Message**: String • **WorkItem**: Object ○ **sysId**: String ○ **name**: String ○ **type**: String ○ **priority**: String ○ **status**: String ○ **lockedBy**: String ○ **additionalComments**: String ○ **requestContent**: String ○ **responseContent**: String ○ **deferredTill**: Date/Time ○ **sla**: Date/Time ○ **created**: Date/Time ○ **startedOn**: Date/Time ○ **completedOn**: Date/Time ○ **lastStartedTime**: Date/Time • **status**: String • **message**: String
Start Process Action	Triggers a specific bot process in RPA Hub and returns the status.	• **Process Name**: Record • **Robots**: Array[String] Ignore an entry in this field if a robot pool is	• **Action Status**: Object • **Result**: String • **Robots**: Array[String]

RPA actions (continued)

Action	Description	Action inputs	Action outputs
		assigned to the bot process. When invoking the **Start Process** Workflow Studio action, the robots that are in the **In Maintenance** life-cycle stage status are treated as failed robots.	• **Process**: String • **Failed Robots**: Array[Object]
Stop Process Action	Stops a bot process. If Graceful Stop is enabled, this action gracefully stops the bot process.	• **Process Name**: Record • **Robots**: Array[String] • **Wait For Graceful Stop**: True/False If this check box is selected, then bot process must stop gracefully. For more information about stopping a bot process, see Stop an unattended bot process in RPA Hub. For more information about Graceful stop, see Using Graceful Stop functionality in RPA Hub. For more information about Robot Pool, see Robot pool in RPA Hub.	• **Result**: String • **Action Status**: Object • **Robots**: Array[String] • **Process**: String • **Failed Robots**: Array[Object]
Unassign User from Attended Automation Process Action	Unassigns the user from the attended automation process.	• **Process Name**: Record • **Username**: Record	• **status**: String • **message**: String • **Action Status**: Object
Unassign User from Attended Robot Action	Unassigns the user from the attended robot and retires the attended robot.	**Username**: Record	• **Action Status**: Object • **Status**: String • **Message**: String

RPA actions (continued)

Action	Description	Action inputs	Action outputs
Update Process Parameter Action	Updates the **Value** field for the process parameter.	• **Process**: Record • **Parameter Name**: String • **Value**: String	• **status**: String • **Action Status**: Object • **message**: String
Update WorkItem Action	Updates work queue items.	• **Queue Name**: Record • **WorkItemID**: String • **Stage**: String • **DefferedTill**: Date/Time • **Name**: String • **Priority**: Choice • **Status**: Choice • **Request Content**: String • **Type**: String • **Remarks**: String • **SLA**: Date/Time • **Response Content**: String The maximum limit for **Request Content** and **Response Content** fields is 8000 characters. If the **Is Work Item Name Sensitive** check box on the associated work queue form is selected, the data in the **Request Content** field is encrypted. Sensitive data with encryption is stored in the work item, so that it can be passed to the RPA Desktop Design Studio or robots.	• **IsNameExist**: True/False • **Status**: String • **Message**: String • **Action Status**: Object
Verify HashCode of a Package Version	Verifies the HashCode of a package version in RPA Hub to validate the HashCode of the manually uploaded automation zip file with the associated	**Package Version**: Record	• **Action Status**: Object • **Message**: String • **isAttachmentValid**: True/False

RPA actions (continued)

Action	Description	Action inputs	Action outputs
	package version record. For more information about verifying the HashCode of a package version, see Verify the HashCode of a package version in RPA Hub.		

RPA subflow

Subflow	Description	Subflow inputs	Subflow outputs
Import Package Version Attachment Subflow	Imports the package version attachments automatically for a package version. To automatically migrate the package attachment (automation zip file) from a lower (non-production) to a higher (production) environment. For more information about importing a package version attachment in RPA Hub, see Import a package version attachment in RPA Hub.	• **Email**: Email • **Package Version**: Record	After the attachment is uploaded successfully or if an error occurs while uploading the attachment, an email notification (if email is provided as an input) is sent to the user who performs this import attachment action.
Start Process Subflow	Triggers a specific bot process in RPA Hub. Returns process job details.	• **Process Name**: Record • **Robots**: Array[String] Ignore an entry in this field if a robot pool is assigned to the bot process. If a process job is not in the **Running** status, the robot	• **Failed Robots**: Array[Object] • **Stopped Robots**: Array[String] • **Pending Robots**: Array[String] • **Process Jobs**: Array[String]

RPA subflow (continued)

Subflow	Description	Subflow inputs	Subflow outputs
		that is associated with the process job is treated as **Failed** in the **Start Process** subflow output.	• **Result**: String • **Message**: String
Stop Process Subflow	Triggers a specific bot process to stop in RPA Hub. Returns robots details and process job status. If Graceful Stop is enabled, this subflow gracefully stops the bot process.	• **Graceful Stop**: True/False • **Process Name**: Record • **Robots**: Array[String]	• **Failed Robots**: Array[Object] • **Stopped Robots**: Array[String] • **Pending Robots**: Array[String] • **Process Jobs**: Array[String] • **Message**: String • **Result**: String

Security Operations spoke

Provides Security Operations actions for flow designers to manage Security Incident Response flow templates.

Security Incident Response flow templates

The Security Incident Response flow templates are created using Workflow Studio.

Note:
Each of the flows is triggered when the **Category** in a security incident is set or changed.

Template	Description
Security Incident Confidential Data Exposure flow template	Perform a series of tasks designed to handle the exposure of sensitive data.
Security Incident Denial of Service flow template	Perform a series of tasks designed to handle Denial of Service (DOS) attacks.
Security Incident Lost Equipment flow template	Perform a series of tasks designed to handle lost equipment.
Security Incident Malicious Software flow template	Perform a series of tasks designed to handle malicious software on your network.
Security Incident Phishing flow template	Perform a series of tasks designed to handle spear phishing emails on your network.

Template	Description
Security Incident Policy Violation flow template	Perform a series of tasks designed to handle security policy violations.
Security Incident Reconnaissance flow template	Perform a series of tasks designed to handle reconnaissance on your network.
Security Incident Rogue Server or Service flow template	Perform a series of tasks designed to handle activity from rogue servers or services affecting your network.
Security Incident Spam flow template	Perform a series of tasks designed to handle email spam on your network.
Security Incident Unauthorized Access flow template	Perform a series of tasks designed to handle unauthorized access to your network.
Security Incident Web/BBS Defacement flow template	Perform a series of tasks designed to handle vandalism directed against one of your BBS or web sites.

Visual Task Board (VTB) Spoke

Provides VTB actions for flow designers to manage the boards, lanes, cards, board members, and assignees.

Board Management Actions

Action	Description	Action Inputs	Action Outputs
Create Freeform VTB Action	Creates a Freeform VTB for any task type. The default lanes are: Todo, Doing, and Done. These lanes can be modified with actions: Add Lane, Rename Lane, Reorder Lane, and Delete Lane	• Name • Board Owner • Default view • Label visibility • Picker visibility • Background color	Board record
Create Flexible VTB Action	Creates a Flexible VTB bound to a single Task table. The default lanes are: Todo, Doing, and Done. These lanes can be modified with actions: Add Lane, Rename Lane, Reorder Lane, and Delete Lane	• Name • Task table • Filter • Board Owner • Default view • Label visibility • Picker visibility • Background color	Board record
Create Guided VTB Action	Creates a data-driven VTB bound to a single Task table along with the fields the lanes are derived.	• Name • Task table • Land field • Filter • Board Owner	Board record

Board Management Actions (continued)

Action	Description	Action Inputs	Action Outputs
		• Default view • Label visibility • Picker visibility • Background color	
Add VTB Member Action	Add a user to a VTB. Only members of the VTB can access the board, Any VTB member can add other members.	• Board record • User record	N/A
Remove VTB Member Action	Remove a user from a VTB. Only members of a VTB can access the board. Any VTB member can remove other members.	• Board record • User record	N/A

Lane Management Actions

Action	Description	Action Steps	State
Add VTB lane	Add a lane to a Freeform or Flexible VTB. This action does not apply to Guided boards, which are constrained to fixed lanes based on fields configured.	• Board record • Lane name	VTB lane record
Rename VTB Lane	Rename an existing lane on a Freeform or Flexible VTB.	• Lane record • New lane name	N/A
Reorder VTB Lane	Reorder lanes on any VTB.	• Lane record • New lane name	N/A
Delete VTB Lane	Delete an existing lane from a Freeform or Flexible VTB.	Lane record	N/A

Card Management Actions

Flow/Action	Description	Action Steps	State
Create VTB Card	Create a VTB card on a Freeform board for a task.	• Lane record • Task record	Card record

Card Management Actions (continued)

Flow/Action	Description	Action Steps	State
Assign VTB Card	Assign a user to a VTB card.	• Card record • User record	N/A
Move VTB Card	Move a VTB card from one lane to another lane. **Note:** For Flexible boards, use the Update Record action to change the state of the underlying task. For Guided boards, this action changes the field on the task associated with that card.	• Card record • Lane record	N/A
Remove Assignee from VTB Card	Remove an assignee from a card.	Card record	N/A

Related topics

Workflow Studio flow system properties

Configure how the system processes flows.

These properties are available for Workflow Studio.

To set Workflow Studio system properties, select **Process Automation** > **Properties** or navigate to the System Properties [sys_properties] table.

Properties for Workflow Studio

Property	Description
The maximum number of records to return when fetching data sn_flow_designer.action_picker_limit	Specify the maximum number of records a look up action or step can return. Workflow Studio ignores records that exceed this limit. • Type: integer • Default value: 1000 • Location: **Process Automation** > **Properties** • More information: Architecture Overview
Set to True to show duration in the stage column com.glide.hub.flow_engine.stage_display.show_duration	Specify whether flows with stages display a duration.

Properties for Workflow Studio (continued)

Property	Description
	• Type: true | false • Default value: true • Location: **Process Automation** > **Properties** • More information: Flow and subflow stages
Allow the option for select users to write a script to populate the value of an input on Flow and Action Designers. sn_flow_designer.input_scripts_enabled	Control permission to write inline scripts to compute input values. • Type: true | false • Default value: true • Location: **Process Automation** > **Properties** • More information: Inline scripts
Specify the log level of system log entries to replicate to the flow log. The system only replicates log entries of the specified level or higher. com.glide.hub.flow_engine.listener_trace.threshold	Specify the threshold required for Workflow Studio to replicate a system log entry to the flow log. These messages are generated by the ServiceNow AI Platform in response to flow execution outputs. For example, a message generated by a business rule that was triggered by a flow record operation. The system only replicates log entries of the specified level or higher. Choices include: **None** The system doesn't replicate system log entries to the flow log. **DEBUG** The system replicates all messages, which include information messages, warnings, and errors, produced by the ServiceNow AI Platform. This option is the lowest log level. **INFO** The system replicates information messages, warnings, and error messages produced by the ServiceNow AI Platform.

Properties for Workflow Studio (continued)

Property	Description
	WARN The system replicates both warnings and errors produced by the ServiceNow AI Platform. **ERROR** The system only replicates errors produced by the ServiceNow AI Platform. This option is the highest log level. • Type: choice • Default value: ERROR • Location: **Process Automation** > **Properties**
com.glide.hub.flow_engine.log_level	Specify the level of messages generated by the Flow Designer execution engine to write to the sys_flow_log table. These messages are generated when the flow engine runs the actions and flow logic of a flow. For example, a message generated by running the Create record action. The system only writes log entries of the specified level or higher. Choices include: **DEBUG** The system logs all messages, which include information messages, warnings, and errors, produced by the flow engine. This option is the lowest log level. **INFO** The system logs information messages, warnings, and errors produced by the flow engine. **WARN** The system logs both warnings and errors produced by the flow engine. **ERROR** The system only logs errors produced by the flow

Properties for Workflow Studio (continued)

Property	Description
	engine. This option is the highest log level. The log level determines what if any log messages are added to the sys_flow_log table. • Type: choice • Default value: WARN • Location: **Process Automation** > **Properties** • More information: Flow execution details
The maximum number of iterations that a loop will run in Workflow Studio. sn_flow_designer.max_iterations	Specify the maximum number of times that a loop can run before being stopped. A loop stops running when it iterates beyond this value, preventing infinite loops. This property only applies to Do the following until and Go back to flow logic. • Type: integer • Default value: 1000 • Location: **Process Automation** > **Properties** • More information: Architecture Overview **Note:** Changing this value doesn't apply to flows that are already running.
Enable flow engine debug messages in the system log com.glide.hub.flow_engine.debug	Enable or disable logging Workflow Studio debug messages in the system log. All debug messages start with a `Flow Designer:` string prefix. • Type: true | false • Default value: false • Location: **Process Automation** > **Properties** • More information: Architecture Overview
Number of times that a flow or subflow can be indirectly triggered during a transaction	Specify the maximum number of times a flow or subflow permits indirect recursion. Workflow Studio ignores all

Properties for Workflow Studio (continued)

Property	Description
com.glide.hub.flow_engine.indirect_recursion_limit	further calls or trigger condition matches from indirect recursion after the limit has been reached. Set the value to any integer equal to or greater than one. The system ignores any property value less than one and instead uses a limit of one. Set the value to one to prevent all indirect recursion. • Type: integer • Default value: 3 • Location: **Process Automation** > **Properties** • More information: Architecture Overview
The maximum number of actions allowed on a flow. sn_flow_designer.max_actions	Specify the maximum number of actions a flow or subflow can contain. Workflow Studio prevents you from adding further actions after the maximum number of actions has been reached. Consider the performance impact raising the maximum number of actions may have. For example, running more actions may conflict with the default transaction quota rule that prevents flows from running longer than an hour. • Type: integer • Default value: 50 • Location: **Process Automation** > **Properties** • More information: Architecture Overview
The maximum number of allowed steps on an action. sn_flow_designer.max_action_steps	Specify the maximum number of steps that an action can contain. Workflow Studio prevents you from adding further steps after the maximum number of steps has been reached. Consider the performance impact raising the maximum number of steps may have. For example, running more steps may conflict with the default transaction quota rule that prevents flows from running longer than an hour. • Type: integer • Default value: 20

Properties for Workflow Studio (continued)

Property	Description
	• Location: **Process Automation** > **Properties** • More information: Architecture Overview
Level of reporting data generated by the flow engine. com.snc.process_flow.reporting.level	Specify when Workflow Studio generates execution details and what information the details include. Options include: **Off** The system doesn't generate flow execution details. The system only generates execution details when you run a test. **Note:** Testing an action or flow generates execution details at the Trace level. **Basic: Runtime states and durations only** The system generates runtime execution details for each flow, subflow, and action run. You can see the runtime state and duration for these basic items. You can also see configuration and runtime values for flow triggers, subflow inputs, and subflow outputs. **Full: Action configuration and runtime values (for debugging only)** The system generates configuration and runtime execution details for each flow, subflow, and action run. You can see the runtime state, duration, input values, and output values for all items. For custom actions, you can also see the runtime state, duration, input values, and output values of its steps. You can also see the configuration values for flow

Properties for Workflow Studio (continued)

Property	Description
	triggers, subflows, actions, and steps that are part of a custom action. **Important:** Only users with the fd_read_operations_all role can see configuration and runtime information such as record values in the flow execution details. Users without this role will only see basic details about the state and duration. **Trace: All values (for testing and Support only)** The system generates configuration and runtime execution details for each flow, subflow, action, and step run. You can see the runtime state, duration, input values, and output values for all items. You can also see the configuration values for flow triggers, subflows, actions, and steps. **Important:** Only users with the fd_read_operations_all role can see configuration and runtime information such as record values in the flow execution details. Users without this role will only see basic details about the state and duration. Testing an action or flow generates execution details at the Trace level. The reporting level determines what if any flow execution details are generated. If a flow runs while reporting is off, execution details are never available for

Properties for Workflow Studio (continued)

Property	Description
	the flow, even if the reporting level later changes. If a flow runs while reporting is activated, execution details are always available for that flow execution, even if the reporting level later changes. • Type: choice • Default value: Off • Location: **Process Automation** > **Properties** • More information: Flow execution details
Number of recent iterations to report for Do until and For Each loops. com.snc.process_flow.reporting.iteration.lastn	As of the Xanadu release, this property has been removed and can no longer specify the number of recent iterations to report in the flow execution details. By default, all flow loops only store execution details for the first and last iterations of a loop. To report on all iterations of a loop, create a flow execution setting record for each flow that you want to collect loop execution details. For more information about flow execution settings, see Flow execution settings. • Type: Removed property • Default value: First and last loop iterations only • Location: Removed from the properties page • More information: Flow execution settings
Truncate runtime values in the flow execution details step configuration com.snc.process_flow.reporting.serialized.val_size_limit	Specify the number of bytes the system has to write runtime values to the flow execution details. To prevent truncation of runtime values, set the value to an integer equal to or less than zero. Preventing truncation of runtime values requires additional system resources and may impact system performance. • Type: integer • Default value: 16384

Properties for Workflow Studio (continued)

Property	Description
	• Location: System Properties [sys_properties] table • More information: Flow execution details
Maximum inputs per action sn_flow_designer.max_action_vars	Specify the maximum number of inputs that can be added to an action. Workflow Studio prevents you from adding further inputs after the maximum number of inputs has been reached. Consider the performance impact raising the maximum number of action inputs may have. For example, processing more action inputs may risk the action running for more than an hour and being stopped by the default transaction quota rule. • Type: integer • Default value: 20 • Location: System Properties [sys_properties] table
Maximum script variables per Script step sn_flow_designer.max_script_variables	Specify the maximum number of input and output variables that can be added to a Script step. Workflow Studio prevents you from adding further script variables after the maximum number of variables has been reached. Consider the performance impact raising the maximum number of script variables may have. For example, processing more script variables may risk the Script step running for more than an hour and being stopped by the default transaction quota rule. • Type: integer • Default value: 20 • Location: System Properties [sys_properties] table
Maximum number of branches allowed for the **Make a decision** flow logic sn_flow_designer.max_decision_branches	Specify the maximum number of branches to use when the **Use Branches** option is selected for the **Make a decision** flow logic.

Properties for Workflow Studio (continued)

Property	Description
	• Type: integer • Default value: 100 • Location: System Properties [sys_properties] table
Option to enable users to create flow variables. sn_flow_designer.flow_variables_enabled	Specify whether users can create custom variables for their flow. • Type: true | false • Default value: true • Location: System Properties [sys_properties] table
Maximum number of popular actions sn_flow_designer.action_picker.popular_actions.max_number	Specify the maximum number of popular actions to display in the Action picker. • Type: integer • Default value: 10 • Location: System Properties [sys_properties] table
Number of consecutive days used to generate popular actions sn_flow_designer.action_picker.popular_actions.last_num_of_days	Specify the number of consecutive days used to generate popular actions. For example, the default value of 7 generates popular actions based on action usage during the last week. • Type: integer • Default value: 7 • Location: System Properties [sys_properties] table
com.glide.hub.flow.restricted_caller_access.track_flows_as_source	Allow the system to generate restricted caller access privilege requests for flows and actions. The access privileges for flows and actions supersede any existing access privileges for script includes and business rules that call flows and actions. This property is inactive on instances upgraded from San Diego and earlier releases. Customers who want to continue using script-based access privileges for flows and actions shouldn't enable this property. Enabling this property requires you to regenerate and approve access privileges for your flows and actions.

Properties for Workflow Studio (continued)

Property	Description
	• Type: true | false • Default value: true for Tokyo and later releases. False for San Diego and earlier releases. • Location: System Properties [sys_properties] table • More information: Restricted caller access privilege settings
com.glide.cs.fdih.interactive.timeout	Specify the length of time, in seconds, before the Workflow Studio Integration Hub action workflow times out. • Type: integer • Default value: 120 • Location: System Properties [sys_properties] table • More information: Specify the action workflow timeout.
com.glide.hub.pause_low_priority_flows_enabled	Enable or disable the pausing of low-priority flows when there are high-priority flows waiting to run. • Type: true | false • Default value: true • Location: System Properties [sys_properties] table • More information: Flow priority
com.glide.hub.flow_engine.stage_display.show_approvers	Show or hide the list of approvers assigned to a stage from a stage field. Set the value to true to show the list of approvers assigned to a stage. Set the value to false to hide the list of approvers assigned to a stage. • Type: true | false • Default value: true • Location: Add to the System Properties [sys_properties] table • More information: Flow and subflow stages
com.glide.hub.flow_engine.stage_display.show_approvers_limit	Specify the maximum number of approvers to display in a stage field as

Properties for Workflow Studio (continued)

Property	Description
	an integer value. Setting this value above 10 risks causing rendering errors in a list view. The stage field for one record can become so big that the list cannot display additional records in the list. • Type: integer • Default value: 5 • Location: Add to the System Properties [sys_properties] table • More information: Flow and subflow stages
com.glide.hub.flow.current_stage_status_on_cancel	Specify the status to apply to the currently running stage when the flow is cancelled. Options include: • cancelled • complete Any remaining stages after the current stage are set to the cancelled status. For example, suppose that a flow has three stages, and that the flow is currently running the first stage. Cancelling the flow sets that status of stage one to complete. Stages two and three are set to the cancelled status. • Type: string • Default value: complete • Location: Add to the System Properties [sys_properties] table • More information: Flow and subflow stages
com.glide.hub.flow.approval.allow_inactive_entity	Specify which types of inactive entities to create approval records for. Options include: • INDIVIDUAL • GROUP • INDIVIDUAL,GROUP • null Use the INDIVIDUAL option to create user approval records for inactive users who are not members of a group. This

Properties for Workflow Studio (continued)

Property	Description
	option doesn't create group approval records for inactive groups, nor does it create user approval records for group members. Use the GROUP option to create a group approval records for inactive groups. This option also creates individual user approval records for group members who have active user records. This option doesn't create individual user approval records for group members who have inactive user records. Use the INDIVIDUAL,GROUP option to create user approval records for inactive users and group approval records for inactive groups. This option also creates individual user approval records for group members who have active user records. Inactive group members don't receive user approval records. Use a null or empty value to prevent creating group approval records for inactive groups, and also prevent creating user approval records for inactive users. • Type: string • Default value: INDIVIDUAL,GROUP • Location: Add to the System Properties [sys_properties] table • More information: Ask for Approval action
com.glide.hub.flow.approval.default_approval_field	When true, the Ask for Approval action uses a table's default approval field for the value of the Approval field input when no input value is provided. For example, the Request Item [sc_req_item] table uses the Approval field to display the approval state. Set this value to false when you want the Ask for Approval action to behave like classic Workflow approval activities. • Type: true | false • Default value: true

Properties for Workflow Studio (continued)

Property	Description
	• Location: Add to the System Properties [sys_properties] table • More information: Ask for Approval action
com.glide.oneapi.fdih.async.quick.mode	The option to generate flow execution details when running skills. When false, flows, subflows, and actions that are run from a custom skill generate flow execution details. When true, the flow, subflow, or action runs in quick mode, which doesn't produce flow execution details. Enable flow execution details when testing running a flow, subflow, or action from a custom skill. Since flow reporting is off by default, you must either turn flow reporting on for all flows, or create a flow execution settings record for a specific flow, subflow, or action. For more information about flow reporting options, see Activate flow reporting. • Type: true | false • Default value: true • Location: Add to the System Properties [sys_properties] table • More information: Activate flow reporting

Workflow Studio flow trigger types

Each trigger type defines when a flow starts and the starting data available to it. There are triggers for record operations, dates, and application operations.

Record triggers

Use record triggers to start a flow when a record is created or updated.

Trigger	Description
Created	Starts a flow when a record is created in a specific non-system table. **Note:** Some common record types such as requests have their own dedicated triggers. See the application trigger types for a list of application records that have dedicated triggers.

Trigger	Description
Updated	Starts a flow when a record is updated in a specific non-system table. Requires selecting when to run the flow. • **For each unique change**: Triggers the flow for every unique update to a non-system field even if the flow is currently running. **Note:** The system stores a history of every change to a record and determines whether the change is unique. For example, if an incident record's **State** field changes from In Progress to On Hold, the flow can run. However, if the **State** field then changes back to In Progress, the flow can't run. **Note:** Flows that have a record trigger that runs **For each unique change** can produce recursions when run in a non-interactive session. When this type of flow makes a change to the trigger record, the change meets the flow trigger conditions and causes a recursion. • **Once**: Triggers the flow once for the life of the record. • **Only if not currently running**: Triggers the flow for every unique record change if the flow is not currently running on this record. This behavior is the same as the **Always** option in previous releases. • **For every update**: Triggers the flow every time that the record is updated, regardless of whether there has already been or currently are any running contexts for the flow.
Created or Updated	Starts a flow when a record is either created or updated in a specific non-system table. Requires selecting when to run the flow. • **For each unique change**: Triggers the flow for every unique update to a non-system field even if the flow is currently running. **Note:** The system stores a history of every change to a record and determines whether the change is unique. For example, if an incident record's **State** field changes from In Progress to On Hold, the flow can run. However, if the **State** field then changes back to In Progress, the flow can't run. **Note:** Flows that have a record trigger that runs **For each unique change** can produce recursions when run in a non-interactive session. When this type of flow makes a change to the trigger record, the change meets the flow trigger conditions and causes a recursion. • **Once**: Triggers the flow once for the life of the record. • **Only if not currently running**: Triggers the flow for every unique record change if the flow is not currently running on this record. This behavior is the same as the **Always** option in previous releases. • **For every update**: Triggers the flow every time that the record is updated, regardless of whether there has already been or currently are any running contexts for the flow.

Note:
Flows including approval actions should only run the trigger once. In cases where you need to update and resubmit an approval, consider using a Go back to flow logic to ask for approval again.

REST triggers

Use REST triggers to start a flow after a specific REST API request.

Note:
This feature requires an Integration Hub Enterprise subscription. For more information, see Request Integration Hub.

Trigger	Description
REST API - Asynchronous	Start a flow from an inbound API call or webhook from an external system. Configure the trigger start conditions without having to write or maintain custom code. For more information, see REST API trigger.

Scheduled triggers

Use scheduled triggers to start a flow after a specific date and time or repeatedly at scheduled intervals. Scheduled triggers use the instance timezone to determine when to start a flow.

Note:
Because flows are processed asynchronously, a flow with a scheduled trigger may not run at the exact scheduled time its trigger conditions were met. For example, if a scheduled flow is triggered during core business hours, the system may have to process other events in the queue before it can run the scheduled flow.

Trigger	Description
Daily	Starts a flow at a specific time every day.
Weekly	Starts a flow at a specific time every week.
Monthly	Starts a flow at a specific time every month.
Run Once	Starts a flow once at a specific time but does not repeat. If you select a past date or time, the system schedules the flow to run as soon as possible.
Repeat	Starts a flow at regular intervals you define.

Application triggers

Use application triggers to start a flow when application-specific conditions are met.

Trigger	Description
Kafka Message	Starts a flow when there's a message in a topic in your Kafka environment. For more information, see Create a flow with a Kafka Message trigger.
MetricBase	Starts a flow when a MetricBase trigger is met. Requires the MetricBase application. For more information, see Create a flow with a MetricBase trigger.

Trigger	Description
Proactive Analytics	Starts a flow when Proactive Analytics KPI score or KPI threshold values are met. Requires a Performance Analytics subscription to Proactive Analytics. For more information, see Create a flow with a Proactive Analytics trigger.
Service Catalog	Starts a flow from a Service Catalog item request. For more information, see Create a flow with a Service Catalog trigger. **Note:** Service Catalog triggers do not support catalog variables as part of the trigger condition. Instead, get or create catalog variables in the main body of the flow.
SLA Task	Starts a flow from an SLA Definition record. For more information, see Create a flow with an SLA Task trigger.

Inbound email triggers

Start a flow when your instance receives an email.

Inbound email flows take priority over inbound email actions. If you create flows with inbound email triggers, emails are first processed by the inbound email triggers before they are processed by inbound email actions.

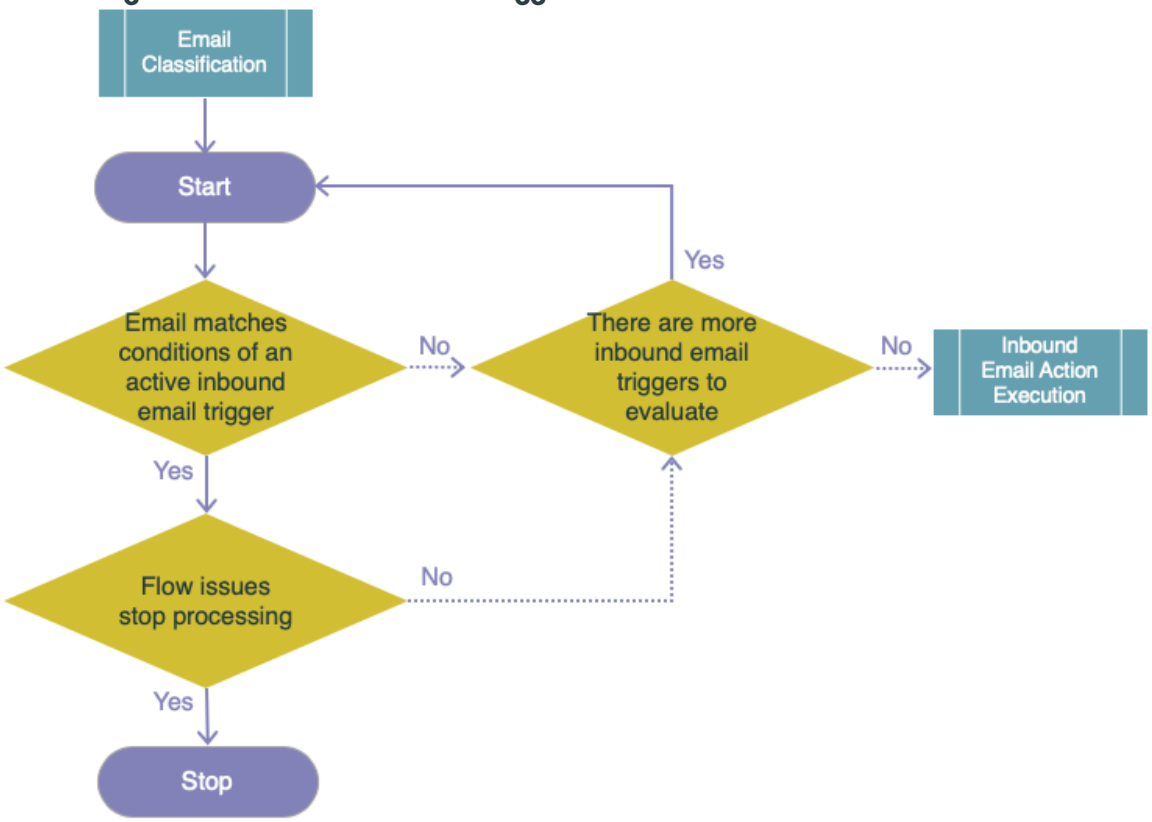
With inbound email actions, you don't have full control over email attachment handling or assigning the target record of an email. When you create a flow with an inbound email trigger, you can perform these actions with the Move Email Attachments to Record action and the Associate Record to Email action. For greater control over email attachments, you can also use the Look up email attachments action to access a specific attachment as a data pill.

Although you can process an inbound email with multiple inbound email actions, you can't process an inbound email with multiple flows by default. Additional configuration is required. For information on how to stop processing in inbound email actions, see Specifying the inbound email processing order.

For more information on running multiple flows on an inbound email, see Allow multiple triggers to process an inbound email.

The following diagram shows how inbound emails are processed by inbound email triggers. After the email has been classified as a reply, forward, or new email, the system tries to match the email to an active inbound email trigger. If the email meets the conditions of an inbound email trigger, the flow runs. If the flow issues stop processing, the email is finished being processed. If the flow does not issue stop processing, the system evaluates the conditions of more inbound email triggers. If there are no more inbound email triggers to evaluate, the system tries to match the email with an active inbound email action instead.

Processing emails with inbound email triggers



Important:
Inbound email flows use the email sender as the user who initiates the session. If the system doesn't recognize the sender, the inbound email flow runs as the Guest user. Setting the inbound email flow to run as the user who initiates the session ensures that the flow actions are limited by user access controls. If the initiating user needs elevated privileges for some reason, have the inbound email flow call a subflow that runs with the required roles. To test access controls for an inbound email flow, impersonate a typical inbound email user and manually trigger the flow.

Spoke triggers

Spokes can have conditional and event-driven external triggers or webhooks that start from third-party applications. The webhooks act as the triggers that provide the data to a flow. For example, when you create a P1-level issue in a third-party issue-tracking application, it updates the incident database record in the ServiceNow instance. To implement this flow, follow these steps:

1. Set up a flow
2. Set up external trigger endpoints.

Advanced options

Specify the user session requirements needed to start a flow with **Advanced Options**.

When to run the flow

Determine the type of session that can trigger the flow, whether to run the flow when triggered by certain users, and which tables can trigger the flow.

Interactive session options

Option	Description
Only Run for Non-Interactive Session	Flow that is only triggered in non-interactive sessions. See Non-interactive sessions.
Only Run for User Interactive Session	Flow that is only triggered in interactive sessions.
Run for Both Interactive and Non-Interactive Sessions	Flow that is triggered in all sessions.

User options

Option	Description
Do not run if triggered by the following users	Flow that does not trigger for a selected list of users. Click the Add User icon () to add users to the list.
Only run if triggered by the following users	Flow that triggers only for a selected list of users. Click the Add User icon () to add users to the list.
Run for any user	Flow that runs for any user.

Table options

Option	Description
Run only on current table	Flow that is only triggered for the selected table.
Run on current and extended tables	Flow that is triggered for the selected table and any extended tables.

Where to run the flow

Determine whether to run the flow in the background or in the current session.

Option	Description
Run flow in background (default)	Flow that runs asynchronously in the background. Use this option for flows that do not require immediate updates and to allow other system processes to run at the same time.
Run flow in foreground	Flow that runs synchronously in the current session. Use this option to provide immediate updates to an end user. For example, if a flow opens a task after the previous task closes, use this option to open the next task immediately after a user closes one.

Option	Description
	Note: Running a flow in foreground may block the current session thread and prevent user input until the flow finishes. Avoid running flows in the foreground when they contain actions that cannot be interrupted, such as actions that run script. Actions or flow logic that pause a flow will not block a session.

Data pills available by trigger type

Flow designers have access to data pills from the trigger.

Trigger Type	Data pills available
Record	**[Table Label] Record** An object containing the triggering record. **Changed Fields** An array of objects containing the field values that changed. This data pill is only available for the **Updated** or **Created or Updated** trigger types. **Note:** To process the **Changed Fields** array data pill, you will need to use For Each flow logic. For more information on working with array data pills, see Complex data. **[Table Label] Table** The Sys ID of the table containing the trigger record. **Run Start Date/Time** Date/Time object that stores when the flow started in the system's local timezone. Use this data pill to pass a Date/Time value to other actions and steps such as the Create record action or the Update record action. **Run Start Time UTC** Date/Time string that stores when the flow started in Coordinated Universal Time (UTC). Use this data pill to pass data to legacy flows that expect UTC date-time strings.
REST API - Asynchronous	**Path Parameters** An object containing path parameters in the inbound request. **Query Parameters** An object containing query parameters in the inbound request. **Request Headers** An object containing headers in the inbound request. **Request Body**

Trigger Type	Data pills available
	Complex data object that defines the body structure of the inbound request. For more information on complex objects, see Complex data.
Date	**Run Start Date/Time** Date/Time object that stores when the flow started in the system's local timezone. Use this data pill to pass a Date/Time value to other actions and steps such as the Create record action or the Update record action. **Run Start Time UTC** Date/Time string that stores when the flow started in Coordinated Universal Time (UTC). Use this data pill to pass data to legacy flows that expect UTC date-time strings.
SLA Task	**Task SLA Record** An object containing the triggering Task SLA record. **sla_flow_inputs** An Object containing Task SLA Definition values.
Inbound Email	**Email Record** An object containing the triggering Email record. **[Table Label] Table** The Sys ID of the table associated with the target email. **Body Text** A String containing the body of the email message. **Subject** A String containing the subject of the email message. **User Record** An object containing the user who sent the triggering email. If the sender does not have an associated User record, the data pill lists the object for the Guest user. **From address** A String containing the sender email address.
Metric Base	**MetricBase Trigger Definition Record** An object containing the triggering MetricBase Trigger Definition Record. **Level** The Integer value of the MetricBase trigger level. **Time of Metric Event** The Date/Time value of when the metric event occurred. **Record** An object containing the record for which metric events have been collected.
Service Catalog	**Requested Item Record** An object containing the triggering Requested Item record.

Trigger Type	Data pills available
	Run Start Date/Time Date/Time object that stores when the flow started in the system's local timezone. Use this data pill to pass a Date/Time value to other actions and steps such as the Create record action or the Update record action. **Run Start Time UTC** Date/Time string that stores when the flow started in Coordinated Universal Time (UTC). Use this data pill to pass data to legacy flows that expect UTC date-time strings. **Table Name** The table name containing the requested catalog item.
Kafka Message	**Messages** An array of objects containing the messages received from the Kafka topic. Each message has the following data pills. • Headers: An array of headers, with each header containing a Key and a Value. The Key-Value pair provides additional information about the message. Both the Key and the Value are strings. • Payload: A string containing the text of the message. • Key: A sting identifying the insertion order for the message. Messages with the same key are processed in order.

General guidelines

Follow these general guidelines when creating record triggers.

Determine whether your flow needs a trigger or variable input

Flows always run when their trigger conditions are met. Triggers always provide the same data as input for flows. If you need variable input to initiate a flow instead, create a subflow.

Add conditions to specify what record values start your flow

Starting a flow only when needed consumes fewer system resources than starting a flow, pausing it, and waiting to resume the flow until a specific record condition applies. Instead of creating a flow that starts with a Wait for condition action, redesign the flow to include the wait condition as part of the record trigger.

Create unique conditions for record triggers on the same table

To prevent flows from overwriting each other, create unique conditions for each flow running on the same table. If multiple flows on the same table have the same filter conditions, there is no way to know the order in which the flows run. Using conditions also helps to optimize flow performance by returning a more precise, smaller set of records.

Ignore records added or updated by import and update sets

Record triggers ignore records added or updated by applying an update set or importing an XML file. These operations apply to the entire application or table rather than an individual record.

Replace record triggers on Service Catalog tables with Service Catalog application triggers

Flow Designer no longer displays Service Catalog tables as options for record triggers. Instead, create flows that use the Service Catalog application trigger type.

Verify that the users who trigger a flow have access to trigger condition data

Since flows typically run as the user who triggers them, verify that users have access to all of the data specified in the trigger conditions. Avoid creating trigger conditions to related tables that typical users don't have access to. If your flow trigger conditions require access to role-restricted data, run your flows with the role needed to access that data.

Workflow Studio input and output data variables

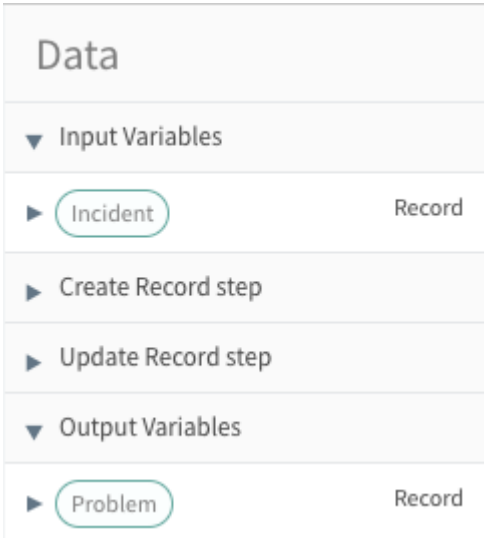
Actions and subflows use variables to store input and output data. The variable data type determines what kind of data it stores and its advanced configuration options.

Data variables available for actions

The Workflow Studio UI displays action inputs as data pills in the Input Variables section of the Data pane. Action outputs are listed as data pills in the Output Variables section of the Data pane.

Workflow Studio does not display step input variables in the data pane. The output variables produced by each step are available in the data pane as part of the step that created them. For more information on using input and output variables to create a custom action, see Building actions.

Sample Workflow Studio data panel



Each data pill consists of a label and a data type description. For example, the Incident input variable uses the Record data type to store an incident Sys ID. The Problem output variable uses the Record data type to store a problem Sys ID. You can expand data pills to see the contents and hierarchy of any child elements.

Data variables available for flows and subflows

The Workflow Studio UI displays the flow trigger as a data pill in the data pane, and displays subflow inputs as data pills in the Subflow Inputs section of the Data pane. Subflow outputs are listed as data pills in the Subflow Outputs section of the Data pane.

Workflow Studio does not display action input variables in the data pane. The output variables produced by each action are available in the data pane as part of the action that created them. For more information on using input and output variables to create a flow, see Building flows.

Supported variable data types

Workflow Studio supports variable data types to store ServiceNow AI Platform record data and complex data. Variables that store record data must have a data type matching the ServiceNow AI Platform field type of the source data. Variables for complex data must match the type of complex data stored, either an array or an object.

Note:
This reference documentation doesn't list all possible ServiceNow AI Platform data types available to actions and flows. Instead, the reference information focuses on the data types used by ServiceNow provided actions and flows. For a list and descriptions of available ServiceNow AI Platform data types, see field type.

Approval rules data type

Store the conditions for approving or rejecting an approval requests.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

General guidelines

Provide a default value

Create or select an approval rule as a default value.

Array.Boolean data type

Store a sequence of true or false values in an array.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for Array variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Max rows	Specifies the maximum number of entries to display in the Workflow Studio interface. The array can store more values than it displays.

Advanced options for Boolean variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

Array.Choice data type

Store a sequence of choice list values in an array.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for Array variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Max rows	Specifies the maximum number of entries to display in the Workflow Studio interface. The array can store more values than it displays.

Advanced options for Choice variables

Option	Description
Name	Displays the name used to identify the data variable in script calls.
Max length	Specifies the maximum length a user can enter for a choice value. Use this option to restrict the length of input values stored during action design.
Hint	Provides guidance to flow or action designers on how to configure the data.
Choice	Specifies whether the choice list has a value for no selection. Options include: • Dropdown with --None-- • Dropdown without --None-- **Note:** The *Dropdown with --None--* option requires selecting a default choice.
Default	Specifies the choice used when a flow or action designer does not select a choice.
Choices	Specify the choices available to select. Use the add button (⊕) to create a choice. Each choice must have a **Name**, **Value**, and **Order**. See Choice list field type for more information about choice lists.

Array.Datetime data type

Store a sequence of date-time values in an array.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for Array variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Max rows	Specifies the maximum number of entries to display in the Workflow Studio interface. The array can store more values than it displays.

Advanced options for Datetime variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

Array.Integer data type

Store a sequence of numeric integer data in an array.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for Array variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Max rows	Specifies the maximum number of entries to display in the Workflow Studio interface. The array can store more values than it displays.

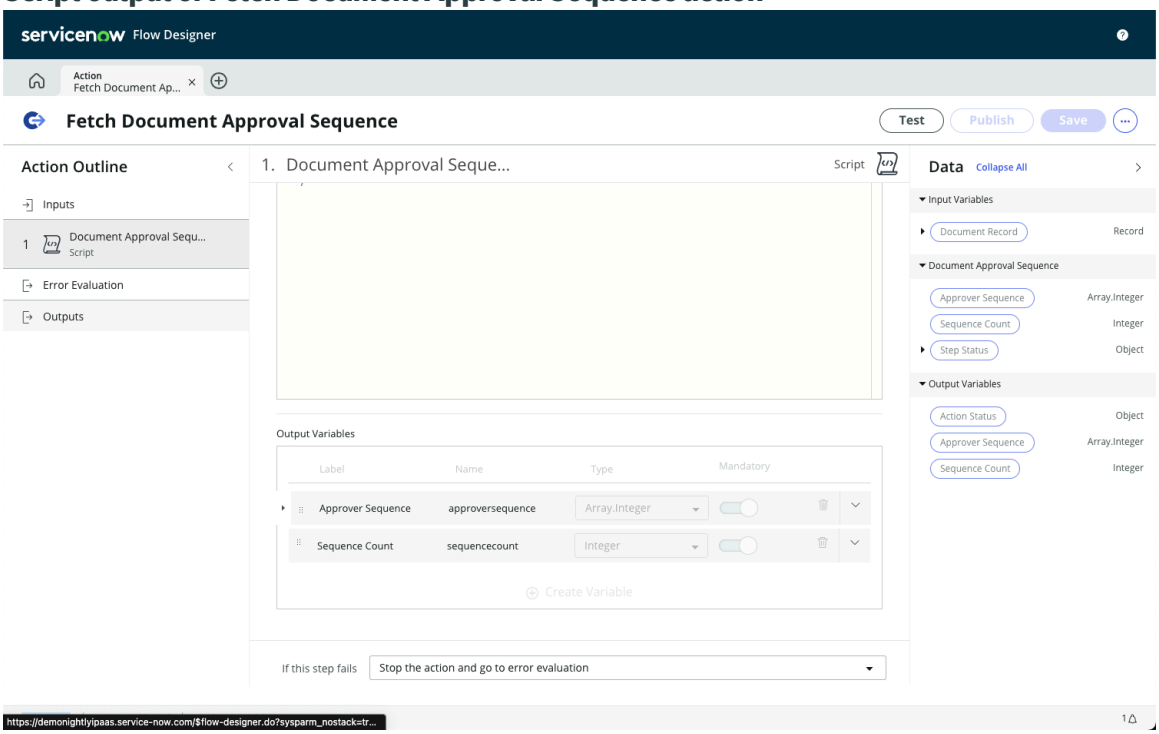
Advanced options for Integer variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

Example

You can use an array of integers to store the output of a script method call. For example, the Fetch Document Approval Sequence action uses a script step to generate an approval sequence data pill. The script output variable stores the sequence of approvers as an array of integers.

Script output of Fetch Document Approval Sequence action



Array.Object data type

Store a sequence of JavaScript objects in an array.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for Array variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.

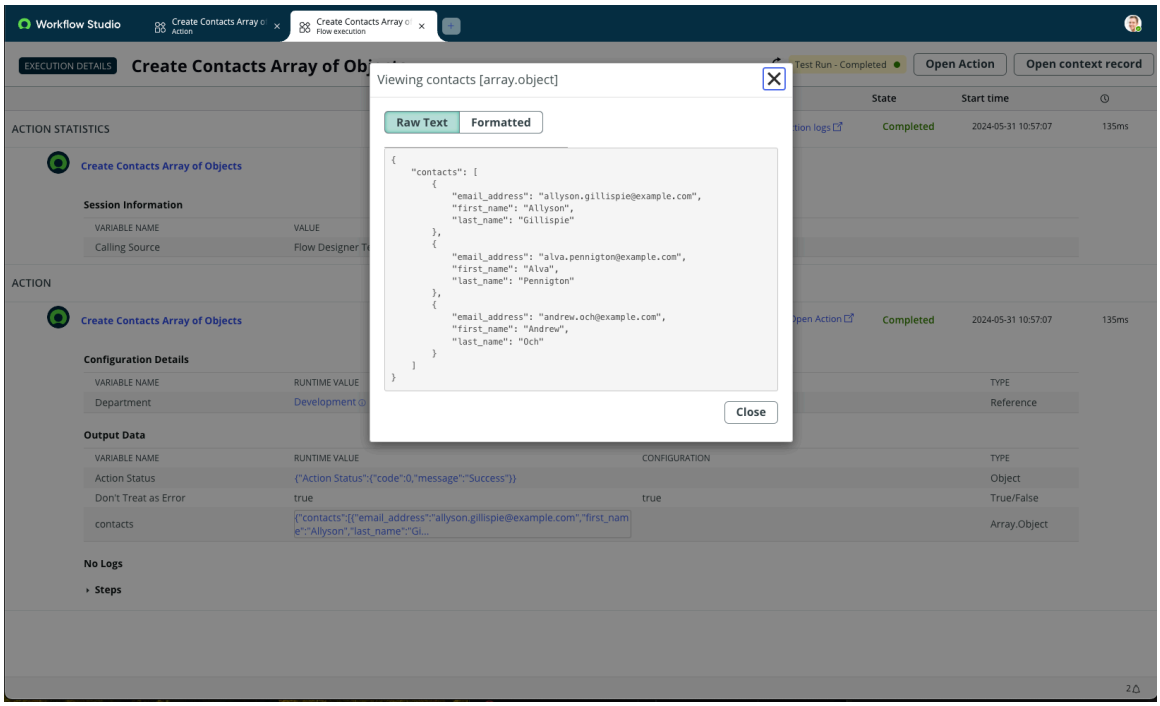
Option	Description
Max rows	Specifies the maximum number of entries to display in the Workflow Studio interface. The array can store more values than it displays.

Advanced options for Object variables

Option	Description
Structure	Specifies how to structure the object hierarchy. Options include: • Create Structure Manually • Start from Template **Note:** Creating the structure manually enables the **Save as Template** option. Starting from a template enables the **Template** option
Save as Template	Stores a manually created object structure for later reuse.
Template	Specifies the existing object structure to apply to this object.

Note:
For more information on using complex object variables, see Complex data.

Example: Create a list of contacts from a list of users



This example uses a custom action to generate a list of contacts details from users in a specific department. To create the custom action to generate an array of objects, see Create a custom action to generate an array of objects from a list of records.

In this example, the contacts array contains three users from the Development department.

```
{
    "contacts":
        "contact": [
            {
                "email_address":
 "allyson.gillispie@example.com",
                "first_name": "Allyson",
                "last_name": "Gillispie"
            },
            {
                "email_address": "alva.pennigton@example.com",
                "first_name": "Alva",
                "last_name": "Pennigton"
            },
            {
                "email_address": "andrew.och@example.com",
                "first_name": "Andrew",
                "last_name": "Och"
            }
        ]
    }
}
```

Array.String data type

Store a sequence of alphanumeric text values in an array.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

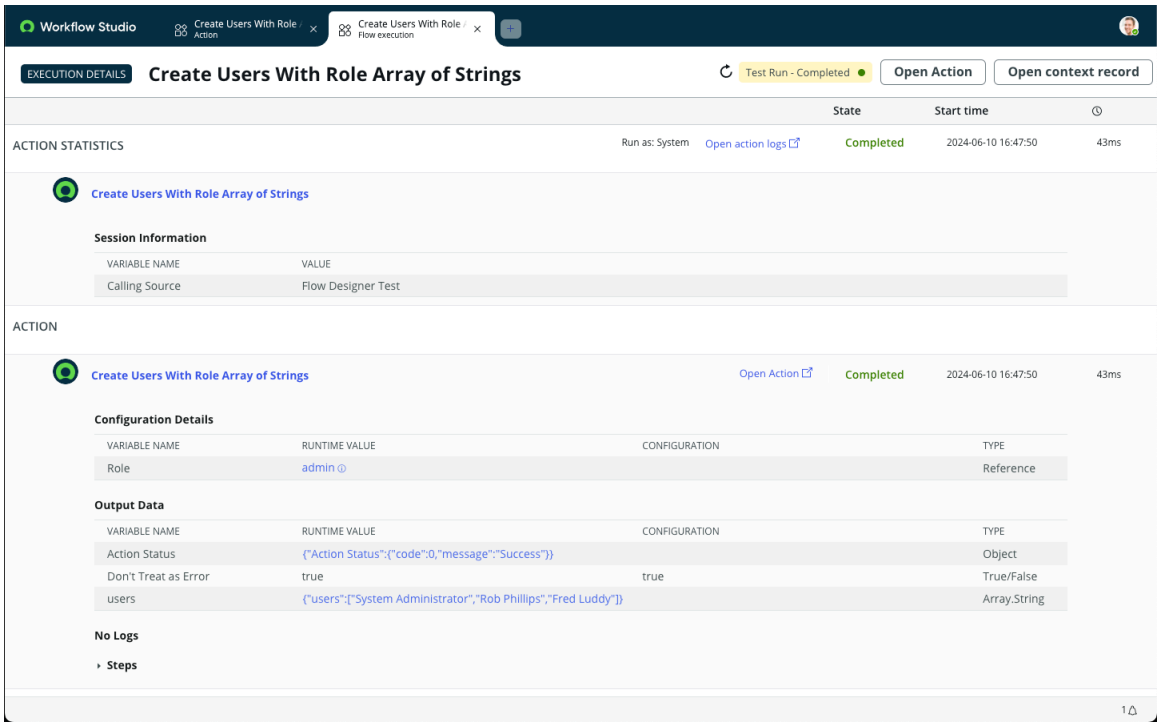
Advanced options for Array variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Max rows	Specifies the maximum number of entries to display in the Workflow Studio interface. The array can store more values than it displays.

Advanced options for String variables

Option	Description
Max length	Specifies the maximum length a string value can have when entered from the user interface. The variable can store longer strings than it can display.
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

Example: Create a list of users who have a user role



This example uses a custom action to generate a list of users who have a specific user role. To create the custom action to generate an array of strings, see Create a custom action to generate an array of strings from a list of records.

In this example, the users array contains three users who have the admin role.

```
{
    "users": [
        "System Administrator",
        "Rob Phillips",
        "Fred Luddy"
    ]
}
```

Choice data

Store choice list values for a specific choice field.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for Choice variables

Option	Description
Name	Displays the name used to identify the data variable in script calls.
Max length	Specifies the maximum length a user can enter for a choice value. Use this option to restrict the length of input values stored during action design.
Hint	Provides guidance to flow or action designers on how to configure the data.
Choice	Specifies whether the choice list has a value for no selection. Options include: • Dropdown with --None-- • Dropdown without --None-- **Note:** The *Dropdown with --None--* option requires selecting a default choice.
Default	Specifies the choice used when a flow or action designer does not select a choice.
Choices	Specify the choices available to select. Use the add button (⊕) to create a choice. Each choice must have a **Name**, **Value**, and **Order**. See Choice list field type for more information about choice lists.

Conditions data type

Store a set of conditions for a specific type of record. You must select a source table to define the conditions.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system

Option	Description
	automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

Datetime data type

Store date-time values.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

File attachment data type

Store a single file attachment as part of the action or flow's associated record rather than a record in the system Attachment table.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

General guidelines

Create one input per attachment

Create a separate file attachment input for each file attachment you want to store. The file attachment data type only supports adding one attachment file per input. The input stores the file attachment in the same record as that associated with the action or flow. You may have to create a custom field of the data type file attachment to store a single attachment.

Manage attachments with existing actions

Use the existing attachment actions to manage attachments associated with records and email. There are existing actions to copy, delete, get from record, look up, and move attachments. Storing an attachment as a file attachment data type prevents you from managing attachments with the standard attachment actions.

Integer data type

Store numeric integer data. Integers are whole numbers only. Fractions and decimals values are not supported.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.

Option	Description
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

JSON data type

Store JSON formatted text values. JSON values can be generated by integration steps or by scripts.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for JSON variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

Example: Choice list items as a JSON array

This example shows storing three choices of a choice list field as JSON formatted text. In this example, the data attribute has an array of three objects as its value. Each object in the array has label and name attributes.

```
{
    data: [
        {
            label: "Choice Option 1",
            name: "choice_option_1"
```

```
        },
        {
            label: "Choice Option 2",
            name: "choice_option_2"
        },
        {
            label: "Choice Option 3",
            name: "choice_option_3"
        }
    ]
}
```

List.[Table] data type

Stores a list of record Sys IDs associated to a specific table. This variable supports ServiceNow AI Platform List field options such as default records and reference qualifiers.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for List variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.
Add [Record Label]	Select one or more records to include as default values for the list. If you filter the list with a reference qualifier, you can only select records that match the reference qualifier conditions.
Reference qualifier conditions	Select the filter conditions applied to the list of records. Flow designers can only select records that match the reference qualifier conditions.

General guidelines

Add a reference qualifier to filter list records

Filter the records the list variable displays as valid options by adding a reference qualifier. The reference qualifier acts as a required list filter and causes the list variable to display only records that match the reference qualifier conditions.

For example, to only displays active incident records add the reference qualifier condition **[Active][is][true]**.

Avoid selecting default records for actions intended for ServiceNow Store

Avoid selecting default records for a list unless you know that all instances have access to the selected records. Spoke developers typically do not have access to the data of the customers who install their custom action. If you want to publish a custom action on the ServiceNow Store, you may need to provide default records as demo data.

Use List variables in For Each flow logic

You can use a List variable to specify the records to process within For Each flow logic. The For Each flow logic ignores any non-record sys_id present in the data. For example, if the List variable contains an email address, the flow logic ignores it.

Object data type

Store a JavaScript object.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for Object variables

Option	Description
Structure	Specifies how to structure the object hierarchy. Options include: • Create Structure Manually • Start from Template **Note:** Creating the structure manually enables the **Save as Template** option. Starting from a template enables the **Template** option
Save as Template	Stores a manually created object structure for later reuse.
Template	Specifies the existing object structure to apply to this object.

Note:
For more information on using complex object variables, see Complex data.

Password (2 Way Encrypted) design considerations

Store encrypted password data that can be decrypted.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options

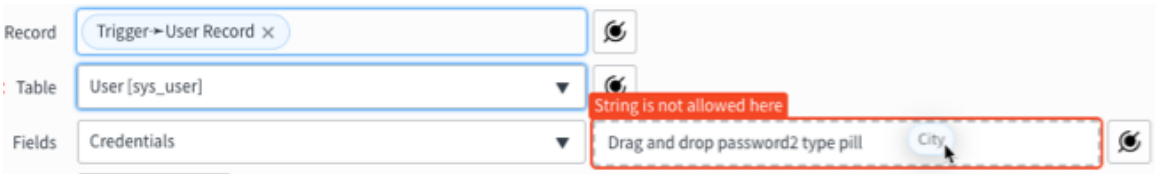
Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

General guidelines

Follow these general guidelines when designing flows containing Password (2 Way Encrypted) data.

Assign values using existing Password (2 Way Encrypted) data pills.

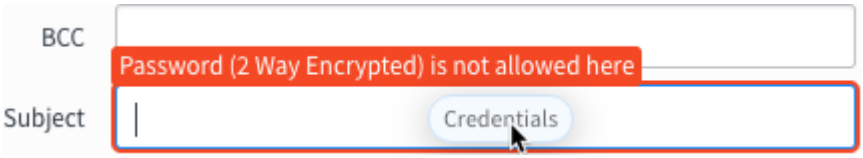
You can only assign a value to a password2 variable by selecting an existing password2 data pill. Selecting values from other field types is not supported. Workflow Studio presents a warning message when invalid data pill types are selected.



Note:
You cannot manually enter Password (2 Way Encrypted) values.

Use Password (2 Way Encrypted) variables only for valid field types

Workflow Studio prevents selecting Password2 data pills as the value for invalid field types. The system presents a warning message when the field is an incompatible type.



Workflow Studio only allows Password2 data pills to be dragged into the following field types.

- Email body fields
- HTML fields
- Password 2 Fields
- PowerShell Input Variables
- REST fields
  - Variables
  - REST payload body
  - Query parameters
  - Headers
  - REST multi-part form values
  - Form URL-encoded values
- SOAP fields
  - Headers
  - Envelope

Note:
you cannot use Password (2 Way Encrypted) variables as conditions

Flow Designer performs a validation check when a user saves, publishes, or tests actions and flows. This check shows that an alert for any data pills dropped in restricted field types and prevents the action or flow from executing. Update the action or flow to remove the invalid data pill and then retry the action.

Set up encryption modules for decryption

Only users with a valid encryption module access can decrypt and view the contents of password2 variables. To specify the encryption algorithm and which roles can access encrypted data, see Password2 encryption with KMF .

Records.[Table] data type

Stores one or more Sys ID references to records in a specific table. The Records data format is also known as a Glide List since it stores a list of GlideRecord Sys ID values.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system

Option	Description
	automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

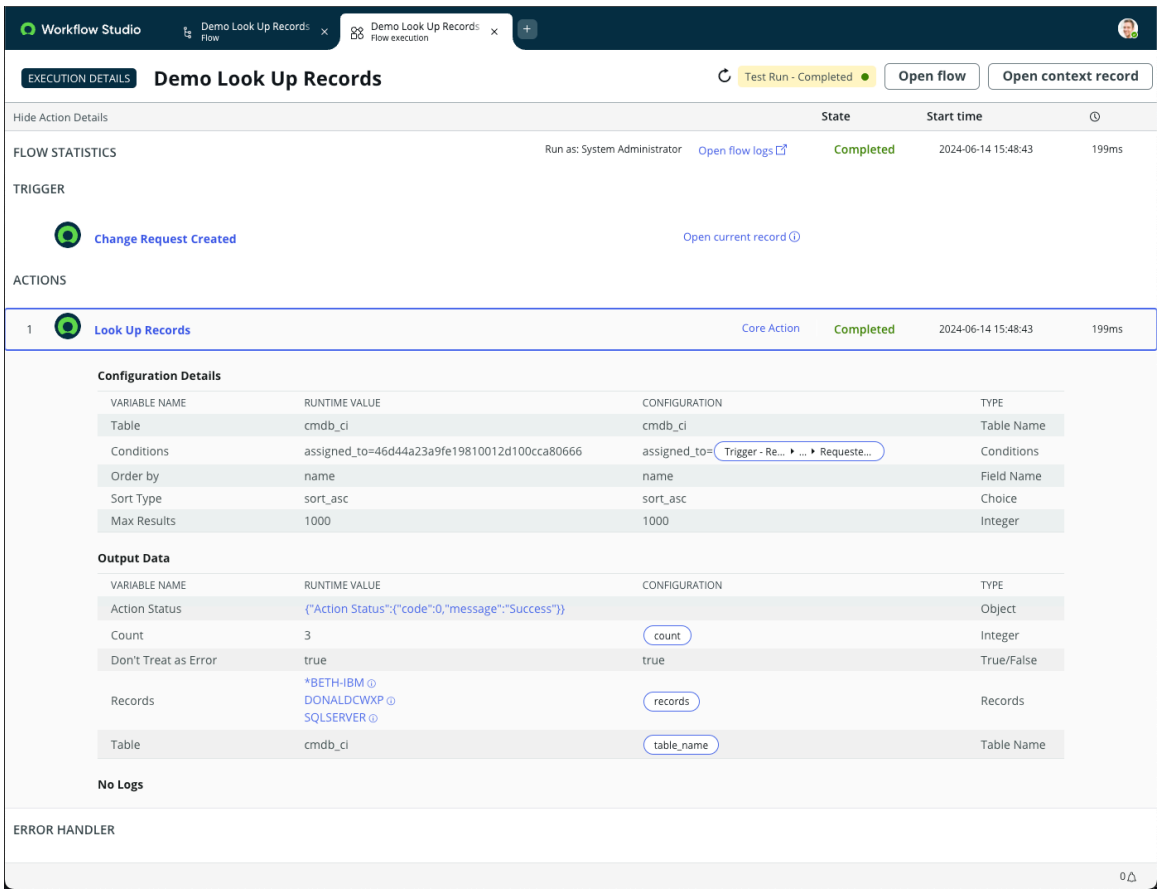
Advanced options

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

Script support

The records data type is supported in script by the GlideRecord class, which can produce an array containing one or more sys_id values. For more information about the GlideRecord class and its available methods, see GlideRecord - Global.

Example: Output of a Look Up Records action



The Look Up Records action uses the Records data type to store its results. In this example, the action looked up Configuration Item records assigned to the requester of a change request.

Each configuration item record is a reference by its Sys ID. The flow execution details shows the configuration item display value, which in this case is the name.

Reference.[Table] data type

Store a single Sys ID reference to a record in a specific table.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for Reference variables

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.
Reference qualifier conditions	Specifies the conditions used to filter records from the target table. The system only displays records from the target table that match the reference qualifier conditions. Use the condition builder to add one or more conditions.

String data type

Store alphanumeric text values using JavaScript data conventions.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options for String variables

Option	Description
Max length	Specifies the maximum length a string value can have when entered from the user interface. The variable can store longer strings than it can display.
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

Table name data type

Store a table name value as specified in the database dictionary. Table names are always alphanumeric strings.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

True/false data type

Store true or false values using JavaScript data conventions.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system

Option	Description
	automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

Advanced options

Option	Description
Hint	Provides guidance to flow or action designers on how to configure the data.
Default value	Specifies the value used when a flow or action designer does not provide a value.

Variables.[Table] data type

Store a reference to a specific table of Glide variables such as a decision input variable.

Basic options

Option	Description
Label	Displays the label used to identify the data variable in the Workflow Studio interface. The label can consist of any text.
Name	Displays the name used to identify the data variable in script calls. The name can only consist of alphanumeric and underscore characters. The system automatically converts the label into a valid name by removing or replacing any special characters.
Type	Indicates the type of data stored by the data variable.
Mandatory	Indicates whether the data variable must contain a value when configured in an action.

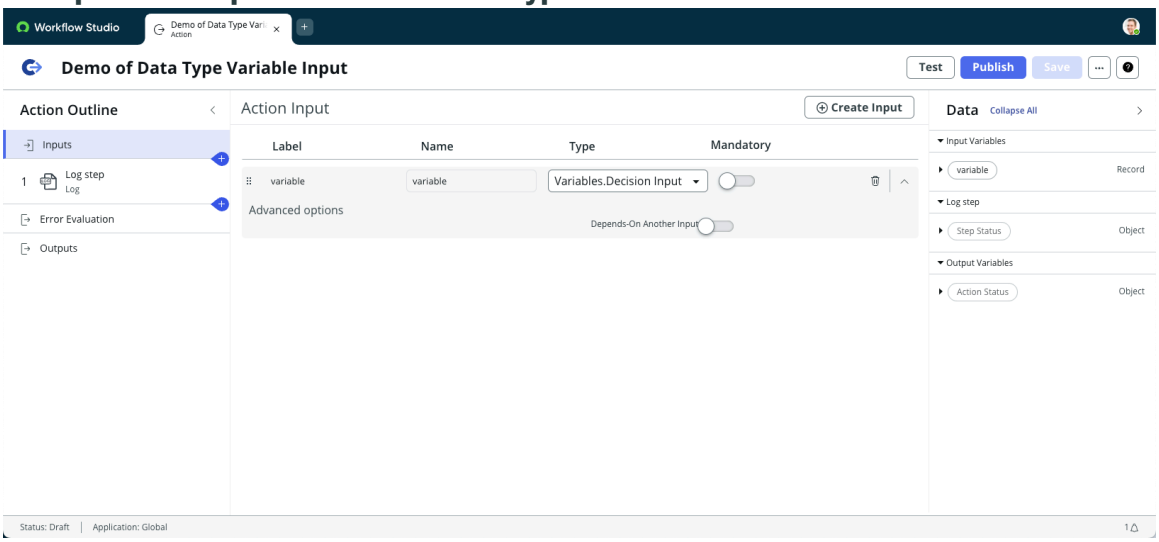
Advanced options for Variables

Option	Description
Data Definition	Specifies the table that stores Glide variables. Each variable type has its own source table to store variables and their associated data as name-value pairs. Some tables that store variables don't support directly adding or editing variable records. For example, the variables of an action or subflow are defined when you create the action or subflow. Edit the action or subflow to change the variables, rather than directly editing the table that stores the variables.
Depends-On Another Input	Specifies whether the data definition value is determined by another input. Use this option to dynamically set the data definition rather than hard-code it to a specific value.

Example: Log decision input variables

This example logs a decision input variable. Create an action with a single input of type variable. Select the Decision Input [sys_decision_input] table as the source of the variable.

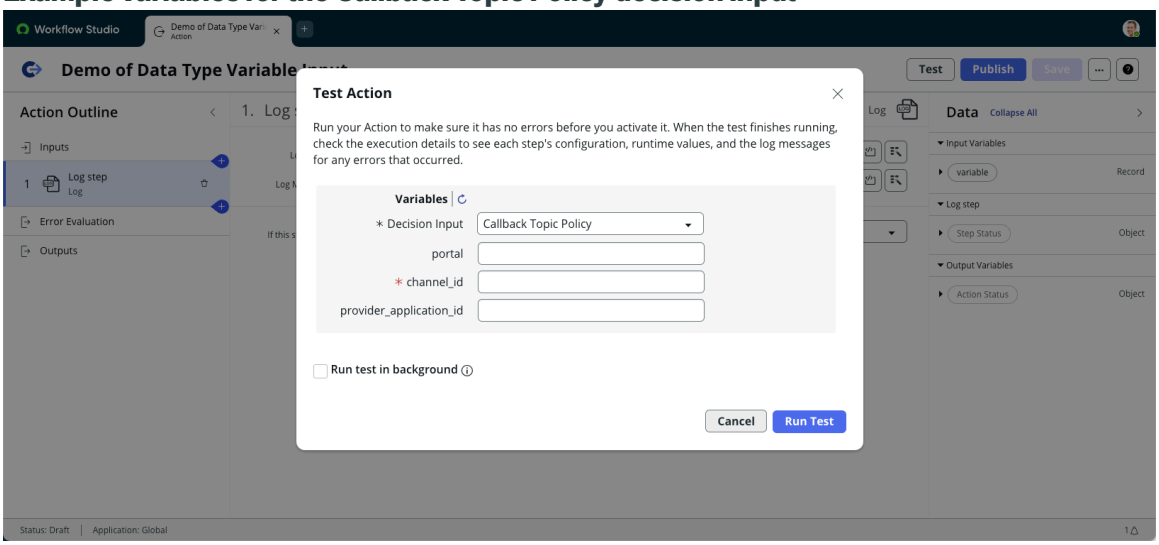
Example action input with variable data type



Add a log step to store the results of your variable selection. Test the action to see a choice list of available decision inputs. Select a decision input type to see its variables. For example, select the Callback Topic Policy to display three variables.

- portal
- channel_id
- provider_application_id

Example variables for the Callback Topic Policy decision input



Each decision input that you select displays a different set of variables.

Workflow Studio steps

A step performs a single operation in an action. You can use Workflow Studio to add steps to a custom action.

A step is a single reusable operation within an action. For example, the **Create Record** step allows action designers to specify the table and field values to use during record creation. Step configuration requires subject matter expertise with application tables, fields, and business logic. Application developers or IT generalists add steps to actions from the Workflow Studio action design environment. Workflow Studio provides a set of ServiceNow core steps to automate ServiceNow AI Platform processes. You can add application-specific steps by activating the associated spoke.

Search steps

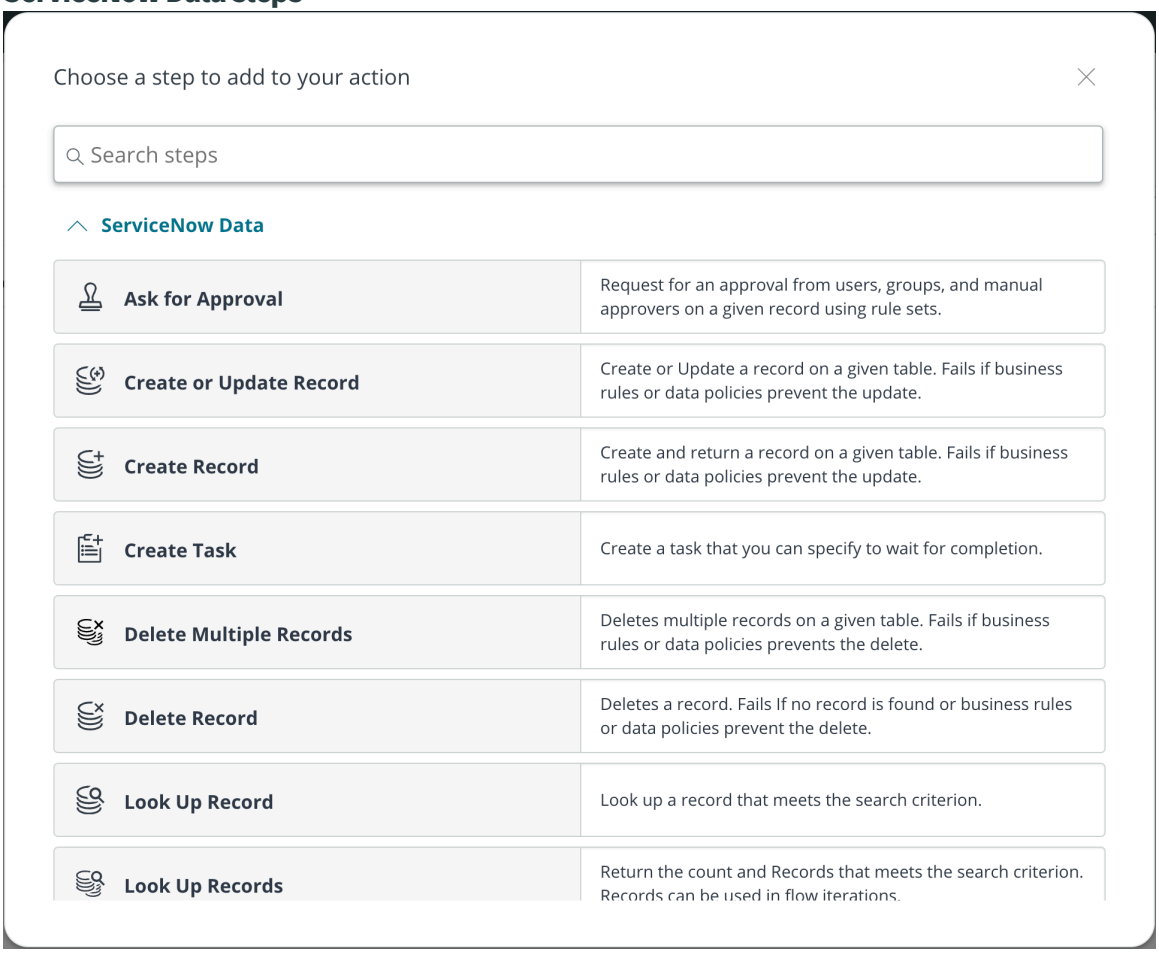
You can use the **Search steps** filter to find a step by name or spoke. As you enter data consisting of at least three characters, Workflow Studio displays a list of steps that match your search criteria.

Search Actions filter

ServiceNow data

ServiceNow Data steps are available to all actions regardless of the spokes installed. Use these steps to perform record operations on your data.

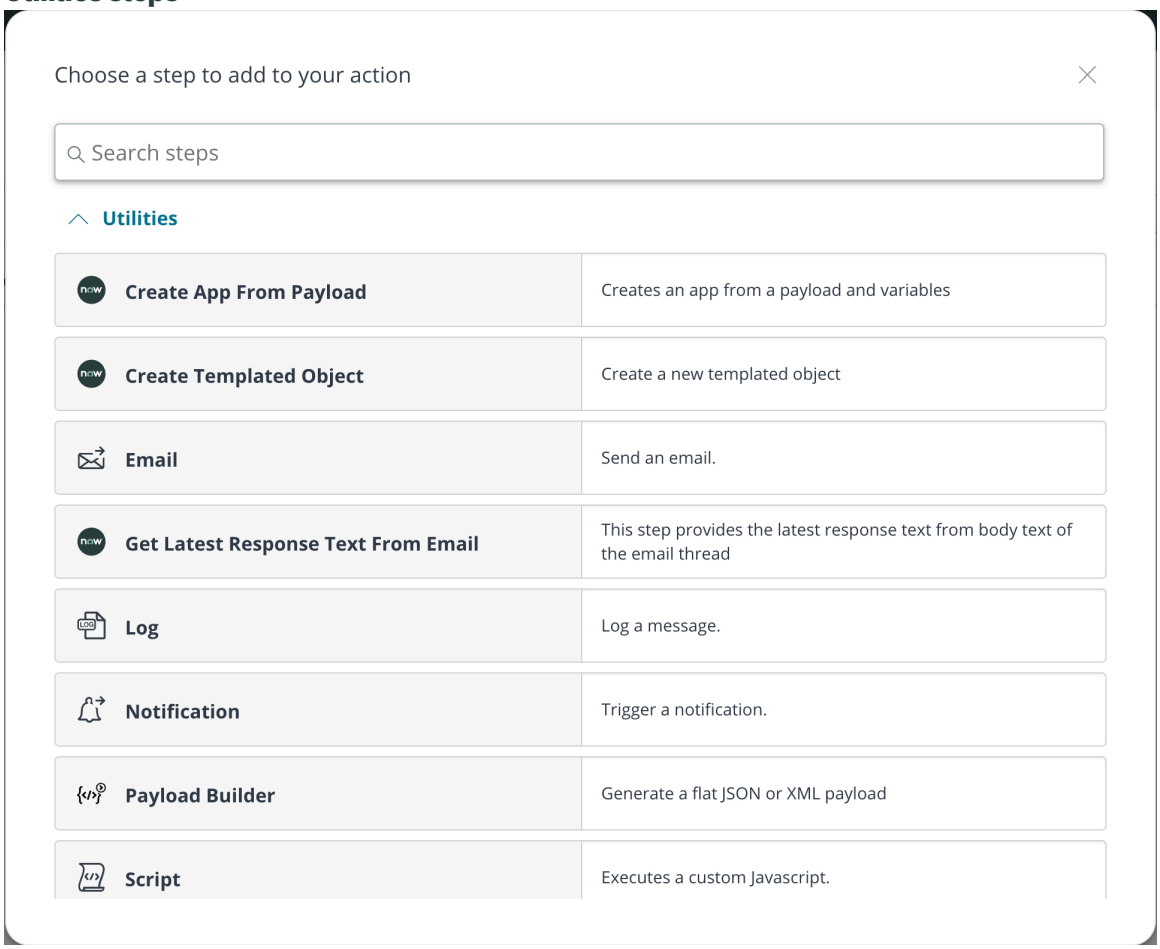
ServiceNow Data steps



Utilities

Utility steps enable you to build payloads, compress data, run scripts, and send notifications.

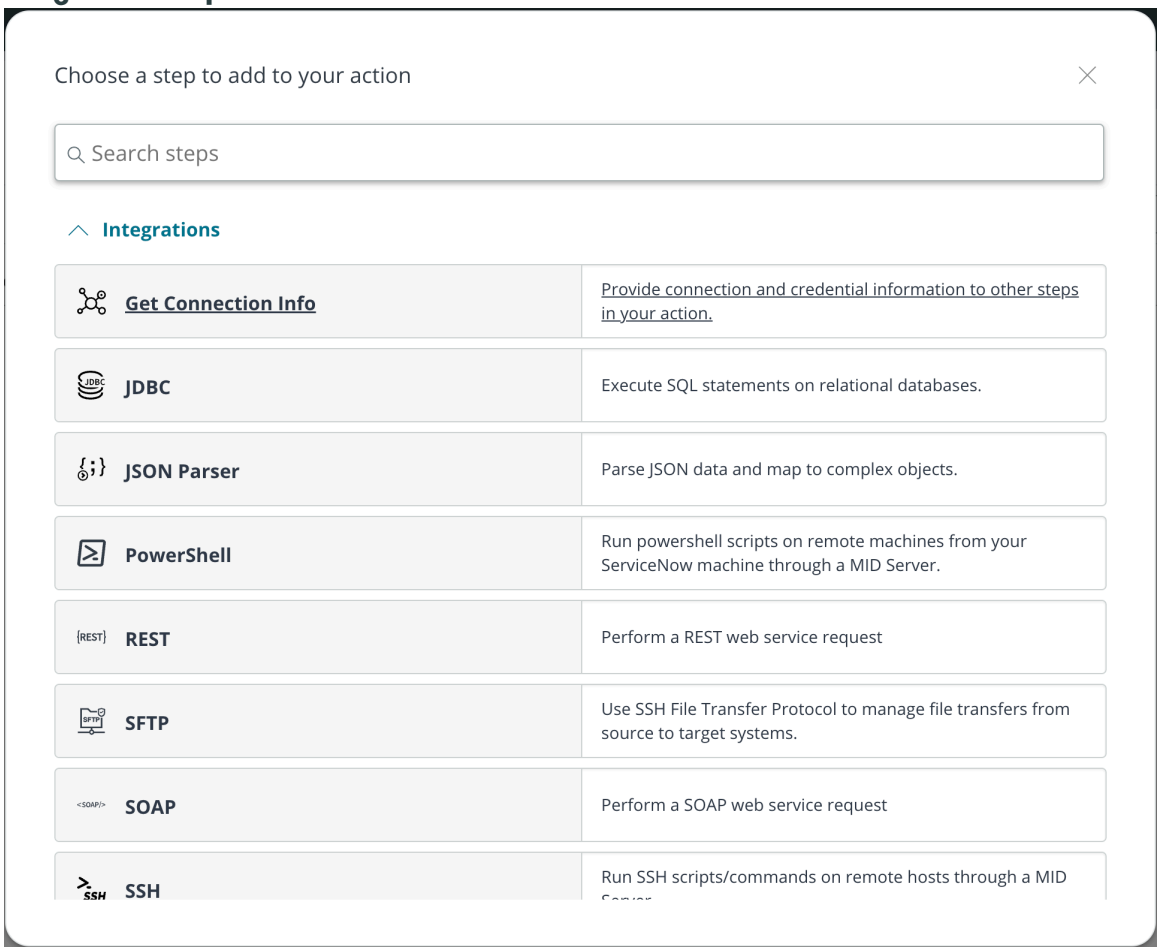
Utilities steps



Integrations

Enable custom actions to integrate with external systems by activating Integration Hub, which adds integration steps to the Workflow Studio interface.

Integrations steps



Ask for Approval step

Request approval for a record with an approval field. You can configure a rule set for an approval, rejection, or cancellation. If a due date is added to an approval, the approval is automatically approved, rejected, or canceled if the approvers have not responded by the designated time.

Classic approvals is a platform feature that enables users or groups to approve or reject a task.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Inputs

Provide a value for each input that your action needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Record

Data type: *Record*

Reference to the record to approve. If the record contains an approval field, Workflow Studio automatically sets the Approval Field input.

Table

Data type: *Table Name*

Table name of the record associated with the approval request. The table that you select must support approvals by having an approval state field. For example, the Task table and its extensions contain approval fields.

Approval Field

Data type: *Field Name*

Field containing the results of approval requests.

Journal Field

Data type: *Field Name*

Field to store history and comments associated with the approval request.

Rules

Data type: *Approval Rules*

Approval and rejection rules to determine which users can approve or reject requests, and what happens after approval or rejection.

Approval or rejection rules include:

- Anyone approves
- All users approve
- All responded and anyone approves
- % of users approve
- # of users approve

In the field beside the approval rule, add the desired approvers. To add approvers:

- Select individual users or groups.
- Drag or select a field from a record.
- Select Manual approvers to allow a manual approver to process an approval or rejection. A manual approver is a user manually added to the Approvers related list who can then approve the request. For example, you can manually add a subject matter expert to a task to approve the request. To learn more about adding manual approvers, see Generate approvals using the approvers related list.

Note:
By default, Ask for Approval generates approval records for inactive users and groups. This behavior allows a flow or action to continue working even when a specific user or group is later made inactive. If you want to change the behavior of generating approvals for inactive entities, set the com.glide.hub.flow.approval.allow_inactive_entity system property. See Workflow Studio flow system properties.

Define rejection rules by adding another OR rule set. When defining approvals, include rejection rules that run when there are no matching approvals. Such rejection rules prevent the flow from remaining in a waiting state. For example, if an approval can be approved by anyone, create a time-based rejection rule in case no one approves it.

Note:
If you set an approval rule with no rejection rule (or vice versa) and the expected approval state is not met, the runtime value will be **canceled**.

For information about how to use inline script to specify approval rules, see the Scripted Approvals in Flow Designer with Flow Variables blog post on the ServiceNow Community.

Due Date

Data type: *Schedule Date/Time*

Due date for an approval state to prevent the flow from endlessly waiting for approval.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Output

These outputs appear in the Data panel. You can use them as inputs elsewhere in your action.

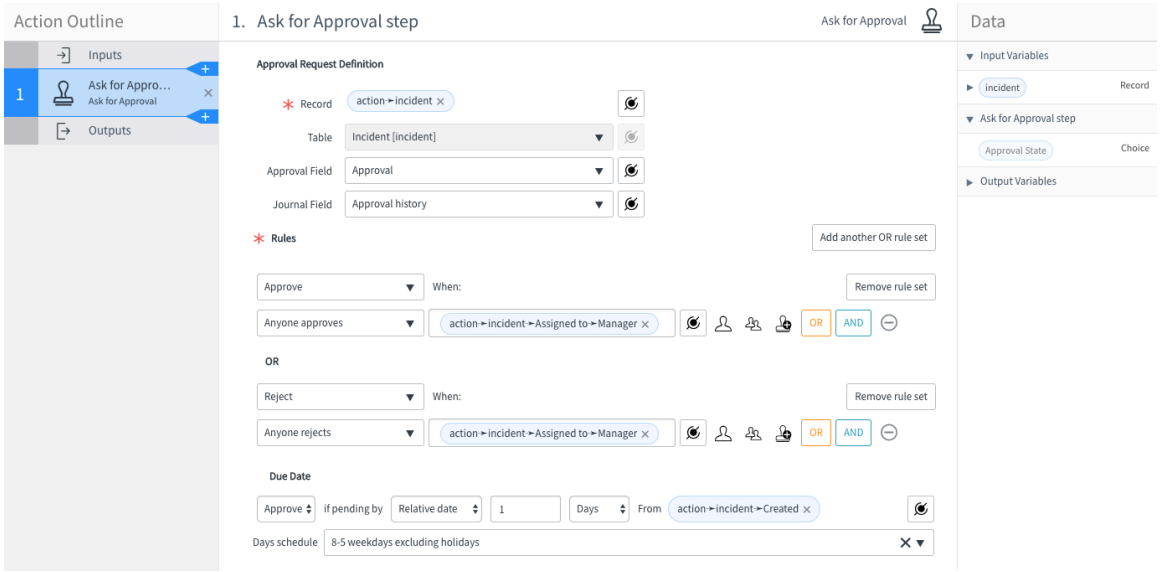
Approval State

Data type: *Choice*

Completion state of the approval request. The flow execution details page displays one of these values.

- Not Yet Requested [not requested]
- Requested [requested]
- Approved [approved]
- Rejected [rejected]
- Cancelled [cancelled]
- No Longer Required [not_required]
- Skipped [skipped]

Example



Create Record step

Creates a record on any table. You can dynamically add and configure fields for the record.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Table

Data type: *Table Name*

Table in which to create record.

Fields

Data type: *Template Values*

Field values to set for the record. For example, to set the short description to a certain value, select **Short description** and set the desired value.

> **Important:**
> The system does not support updating multiple journal fields such as the additional comments or work notes of a task record.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Table

Data type: *Table Name*

Table where record was created.

Record

Data type: *Record*

Reference to record created.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Create or Update Record step

Create or update a record in a ServiceNow table using a single operation. Update a record that exists, or create a record using the values provided.

Identification of existing records

The Create or Update Record step identifies existing records by searching for matching values in the fields that you select as unique identifiers. For example, you can specify that the short description and priority fields uniquely identify an incident. When the step finds an incident with a matching short description and priority, it updates the matching record rather than creating a new record.

Note:

- If no field is selected as a unique identifier, the step creates a record with the field values provided.
- If more than one record matches the value of the unique identifiers, the step doesn't update any records and displays an error message in the flow execution details.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Inputs

Provide a value for each input that your action needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Table

Data type: *Table Name*

Table in which to create or update a record.

Fields

Data type: *Template Values*

Field values to set or update for the record. For example, to set the short description to a certain value, select **Short description** and set the desired value.

To learn about creating template value input, see Create a template value input.

Important:
The system does not support updating multiple journal fields such as the additional comments or work notes of a task record.

If adding the action to a subflow, you can Create a template value input. Dynamically set field values can trigger server-side validation rules but cannot trigger UI policies.

Determines uniqueness

Data type: *True/False*

Option for selecting the field as a unique identifier, which determines when to update or create a record. A record is updated when the incoming field value matches an existing record field value. A record is created when the incoming field value does not match an existing record field value. This option appears when the required table name and fields are selected.

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. This option has no effect on the Step Status output. Choices include:

- **Don't stop the action and go to the next step**: Continues running the action at the next step.
- **Go to Error Evaluation**: Stops running the action and goes to the Error Evaluation section.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your action.

Record

Data type: *Record*

Reference to record created or updated.

Table

Data type: *Table Name*

Table where record was created or updated.

Error Message

Data type: *String*

Error message produced when the record operation fails.

Status

Data type: *Choice*

Completion status of the action. The flow execution details page displays one of these values.

- Created [created]: The action created a record.
- Updated [updated]: The action updated a record.
- Error [error]: The action produced an error.

Step Status

Data type: *Object*

Object data pill containing runtime details about the step. Each step in an action returns a Step Status.

Step Status Code

Data type: *Integer*

Integer data pill indicating whether the step produced an error. A step returns a value of 1 when it produces an error for any reason. For example, a step can produce an error if it is missing mandatory input data or returns output in the wrong data type. A step returns a value of 0 when it runs successfully. You cannot customize these codes.

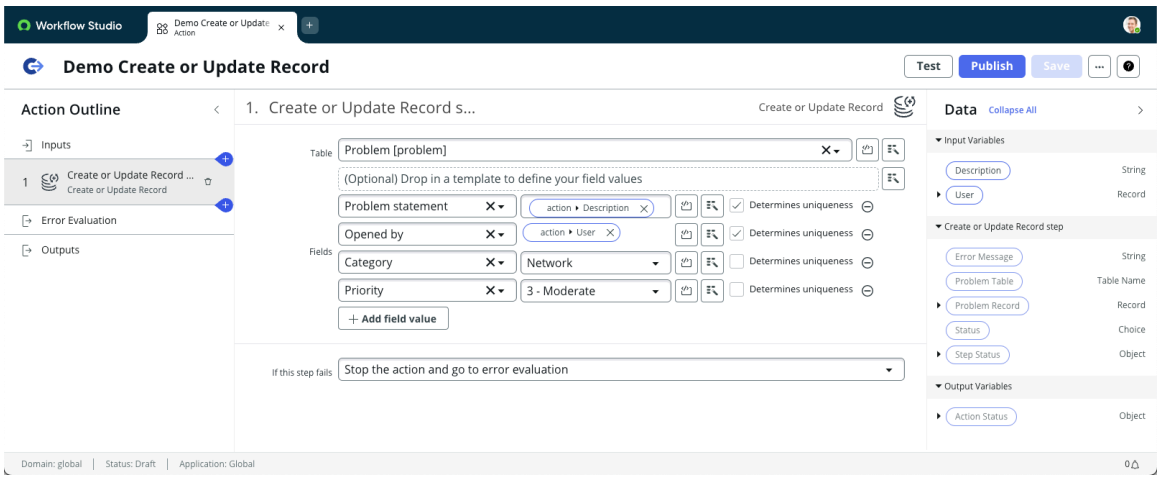
- Success [0]: The step succeeded.
- Error [1]: The step produced an error.

Step Status Message

Data type: *String*

String data pill containing the error message produced by the step or system operation. You cannot customize the step status message.

Example: Create or update a problem record



This action has two inputs. There is a string input for a description, and a reference input for a User record. The Create or Update Record step uses these inputs to create or update a problem record. The Problem statement and Opened by fields determine uniqueness. When the input values match, the step performs an update operation instead of a create operation.

Create Task step

Create a task record in an extension of the Task table. After you choose the task table, you can dynamically select the fields to configure the action. Defining the Parent field associates the task to a parent record.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Table

Data type: *Table Name*

Extension of the Task table in which to create a record. For example, Catalog Task [sc_task] or Incident Task [incident_task].

Field Values

Data type: *Template Values*

Field values to set for the record. For example, to set the short description to a certain value, select **Short description** and set the desired value.

Wait

Data type: *True/False*

Flag indicating whether to pause the flow until the Task record is no longer active. You can add a wait condition by dragging-and-dropping a True/False data pill into this input. The flow only waits for the Task record to complete when the condition field is true.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Table

Data type: *Table Name*

Table where the Task record was created.

Task

Data type: *Record*

Reference to the Task record created.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Delete Multiple Records step

Look up and delete multiple existing records as a single operation. This Workflow Studio step removes the need to first look up a list of records and then delete each record in the list.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Table	Select the table containing the records to delete.
Conditions	Define the filter condition used to look up records.

Field	Description
Order by	Select the field that you want to use to sort the records when more than one record matches the defined conditions.
Sort Type	Determine whether to sort the records alphabetically in ascending or descending order.
Run Business Rules and Workflow	Determine whether to call any business rules and workflows associated with the table.
Don't fail on error	Specify whether to continue running the flow when there is an error.

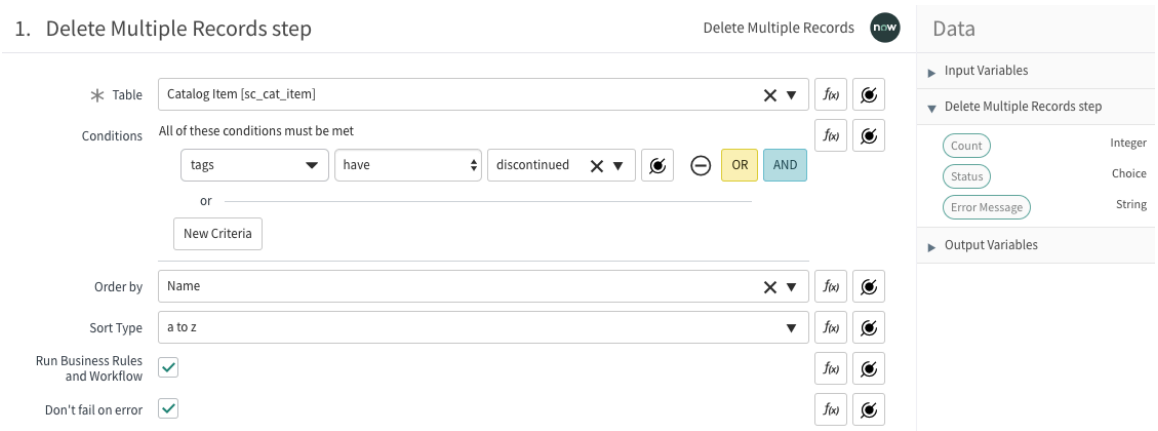
Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Example



Note:
The example is for illustration purposes only.

Outputs

Field	Description	Data Type
Count	Number of records deleted. If no records are deleted, the count is 0.	Integer
Error Message	Message that is displayed if the step produces an error.	String
Status	The completion status of the step as a numeric value. • 0 (success) • 1 (error)	Choice

Delete Record step

Deletes a record on any table.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Record	The record to be deleted. Drag-and-drop a record data pill or use the data pill picker to select a record.
Table	Read-only. Set to the table associated with the record.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Get Connection Info step

Provide the connection and credential details from another step, such as a REST step, to other steps in your action.

Note:
The Get Connection Info step is not available in the base system and requires the subscription to Integration Hub Starter Pack Installer (com.glide.hub.integrations) or later. For more information about the ServiceNow® Integration Hub subscription packages, see Integration Hub usage and subscription. After the required plugin is activated, the step is visible under Integrations.

Roles and availability

After setting up your Integration Hub Starter subscription, the Get Connection Info step is available as an Workflow Studio action step. Users with the action_designer role can create a custom action with the Get Connection Info step.

Inputs

Provide a value for each input that your step needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Input	Data type	Description
Type	Choice	Choose from: • Connection Alias - Associates to connection information required to connect to the remote system.

Input	Data type	Description
		• Credential Alias - Associates to credential data required to connect to the remote system. Depending on which option you choose, the following input requires that you choose the appropriate Connection Alias or Credential Alias.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your action.

Output	Data type	Description
Runtime Alias ID	String	Sys ID of the Connection Alias or Credential Alias record used to connect to the remote system.
Connection ID	String	Sys ID of the Connection record used to connect to the remote system.
Connection URL	String	URL used to connect to the remote system.
Credential ID	String	Sys ID of the Credential record used to connect to the remote system.
Credential Value	Password (2 Way Encrypted)	2-way encrypted password used to authenticate when connecting to the remote system.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Get Latest Response Text From Email step

Provide the most recent reply or forward message in an e-mail chain to other steps in your action.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Inputs

Provide a value for each input that your step needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Input	Data type	Description
Email Record	Record	Email record whose most recent reply or forward message you want to provide to other steps in your action. Select an Email [sys_email] record from the list, or add an Email [sys_email] record data pill from the Data panel.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your action.

Output	Data type	Description
Latest Response Text	String	Body text for the most recent reply or forward message in the Email [sys_email] record that you selected for the step's input. **Note:** If you select an Email [sys_email] record with a Type of New for this step's input, the Latest Response Text output will be the entire body text of the e-mail.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

JDBC step

Create a reusable action to send SQL commands to a relational database.

Note:

- This step requires an Integration Hub subscription. For more information, see Legal schedules - Integration Hub overview. After the required plugin is activated, the step is visible under Integrations.
- The JDBC step runs only on a ServiceNow® MID Server with JDBC step capabilities. Activate the plugin, Integration Hub Standard Pack Installer (com.glide.hub.integrations.standard) or later to use the JDBC capability for the MID Server.

Roles and availability

The JDBC step is available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Sanitizing inputs

Escape all user inputs to eliminate the possibility of a malicious user from executing malicious SQL statements that can result in SQL injection on your target database. When you use data pills in JDBC step SQL statements, sanitize them first using Sanitize SQL transform functions. This transform function category automatically appears when a data pill is dropped into the SQL Statement input.

SQL operations inclusion list

By default, you can run the following SQL operations.

- SELECT
- INSERT
- UPDATE
- DELETE
- SHOW
- DESCRIBE

To enable only some of these SQL operations that the JDBC step can perform, create a MID Server property, *`mid.property.jdbc_operations`* and enter the SQL operations, separated by comma. To learn more about MID Server properties, see MID Server properties.

Note:
Multiple SQL statements are not allowed. Stored procedures with output parameters are not supported.

Fields

Field	Description
Connection Details	
Connection	Type of connection to use. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to

Field	Description
	configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Connection list.
Credential Alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Connection list.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Connection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Database Type	Database type for this connection. The choices are:

Field	Description
	• **MySQL** • **Oracle** • **SQLServer** • **Custom** The default choice is **Custom**. This field is available when **Define Connection Inline** is selected from the Connection list.
JDBC Driver	Driver to use for this connection when it's not a default database type such as DB2 Universal and Sybase. The database Type is **Custom**. This field is available when **Define Connection Inline** is selected from the Connection list and **Custom** is selected from the Database Type list.
Connection URL	URL that the MID Server uses to connect to the specified database. The URL is created automatically when you save the form, and is read-only for the default databases. This field is available when **Define Connection Inline** is selected from the Connection list and **Custom** is selected from the Database Type list.
MID Application	Application that the MID Server must support to be eligible for selection. This field is available when **Define Connection Inline** is selected from the Connection list.
Capabilities	Capability of the MID Server. Select**JDBC**. This field is available when **Define Connection Inline** is selected from the Connection list.
Connection Timeout	Maximum elapsed time, in seconds, for the activity to wait while attempting to connect to the target database. This field is available when **Define Connection Inline** is selected from the Connection list. **Note:** Avoid setting the **Connection Timeout** value to zero, as this may cause a stale connection.
Query Timeout	Maximum elapsed time, in seconds, that the query is allowed to run without a response. This field is available when **Define Connection Inline** is selected from the Connection list.
JDBC Configuration	
SQL Statement	SQL statement that the step executes. **Note:** When you use data pills in step SQL statements, sanitize them first using a preprocessing Script step. For more information, see Sanitizing inputs using the escape functions.
Maximum Rows	Maximum number of rows to be returned from the SQL statement. The default value is **1000**.
Maximum Payload Size (KB)	Maximum allowable payload size, in KB, to be returned from the SQL statement. The default payload size is **5120 KB**. The maximum payload size is **10 MB**.

Field	Description
Test JDBC Step	Test the JDBC step. View test results directly in the Test JDBC Step window. For more information, see Test JDBC step.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Test JDBC step

Test the JDBC step before testing or publishing an action that contains the JDBC step.

Before you begin

Role required: admin

About this task

It is mandatory that you test the JDBC step before testing the action. Testing ensures that the relevant complex object output schema is created from table columns, which can be used as data pills in subsequent steps.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Create an action with a JDBC step.
3. Click **Test JDBC Step**.
   The **Test JDBC Step** pop-up window is displayed.
4. If the JDBC step takes an action input or output of the previous step as its input, provide required input values in the**Step input pills** field to test the JDBC step.

   Note:
   Input values in the **Step input pills** fields are not needed when records are updated, inserted, or deleted.

5. Click **Run Test**.

- When a SELECT query is executed, **Sample Result** is displayed in the **Test JDBC Step** pop-up window. **Sample Result** includes column names, columns types, and the values of the first row.
- When an UPDATE, INSERT, or DELETE query is executed, a message is displayed mentioning the number of rows affected.

6. To use the sample result as the JDBC step output, click **Use Result**.

> **Note:**
> **Use Result** is not displayed when records are updated, inserted, or deleted.

7. To retrieve schema of a different table when a SELECT query is executed in the JDBC step, enter the required value in the **Step input pills** field and click **Run Test**.

Result

When a SELECT query is executed in the JDBC step, **ResultSet** is displayed under **Outputs**. The relevant complex object output is populated. To learn more about complex objects, see Complex data.

What to do next

Test and publish the action.

JSON Builder step

Create a JSON payload to use in another step. Enter values or use data pills to produce a dynamic payload. This step supports several data types, including objects and arrays for nested structures.

Roles and availability

This step requires an Integration Hub subscription. For more information, see Legal schedules - Integration Hub overview. After the required plugin is activated, the step is visible under Integrations.

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Inputs

Provide a value for each input that your action needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Name

Data type: *String*

Name part of a name-value pair. Enter a value or use a data pill to produce a dynamic payload.

Value

Data type: *String*

Value part of a name-value pair. Enter a value or use a data pill to produce a dynamic payload. You can use a complex object pill to populate a row with an object data type, but doing so deletes any children the complex object pill already has.

Type

Data type: *Choice*

Data type for the name-value pair. If you're using a data pill for the name or value, make sure the data type for the pill matches the data type selected here. The options are:

- **String**
- **Object**
- **Number**
- **Boolean**
- **Array**

For arrays and objects, use the plus icon () to add name-value pairs to the array or object.

In case of empty value

Data type: *Choice*

Option to specify what to do if a name-value pair has an empty or null value.

- **Leave as is**: Keeps the empty or null value as an empty string.
- **Omit property**: Excludes a name-value pair if the value is empty or null.
- **Set as null**: Returns an empty or null value as a null data type.
- **Throw error**: Returns an error if a name-value pair has an empty or null value.

Include Outer Structure

Data type: *True/False*

Option to include the curly braces for a top-level JSON container.

Omit Empty Structure

Data type: *True/False*

Option to omit an empty payload. Empty payloads can occur when you select **Omit property** for the **In case of empty value** input for every name-value pair and all name-value pairs in the payload produce empty values.

Structure

Data type: *Structure*

Read-only payload the step produces.

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your action.

Output

Data type: *String*

JSON payload as a string. If you want to use the payload as an object, you can use the JSON parser step to parse the string.

Step Status

Data type: *Object*

Object data pill containing runtime details about the step. Each step in an action returns a Step Status.

Step Status > Code

Data type: *Integer*

Integer data pill indicating whether the step produced an error. By default, a value of 1 indicates that the step produced an error. A value of 0 indicates that the step ran successfully. You can't customize these codes.

Step Status > Message

Data type: *String*

String data pill containing the error message produced by the step or system operation. You can't customize this message.

Generate a JSON Payload

Automatically generate a JSON payload with the **Add JSON for Payload** button. This button opens the Add Payload script editor where you can enter a JSON payload to be automatically de-serialized into structured input.

There are a few details to be aware of when using the JSON payload generator.

- If the JSON has empty keys, the payload is still generated.
- If there are duplicate keys in the script editor, the last key entry in the object overrides the value of any previous keys with the same name.
- If the root object is an array, the root array is wrapped into a root object.
- If an array contains multiple objects, all the keys in the objects are coalesced into one parent object.

The JSON payload generator doesn’t support the following.

- empty payloads
- non-complex object types
- invalid JSON
- empty root objects, such as an empty array
- payloads larger than 65,000 bytes

Note:
Generating a JSON payload overrides any existing structures in the JSON Builder step.

Kafka Producer step

Create an action that publishes events to a topic in your Kafka environment.

Roles and availability

The Kafka Producer step is an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

This step is requires a Stream Connect subscription. For more information, see https://www.servicenow.com/products/automation-engine.html.

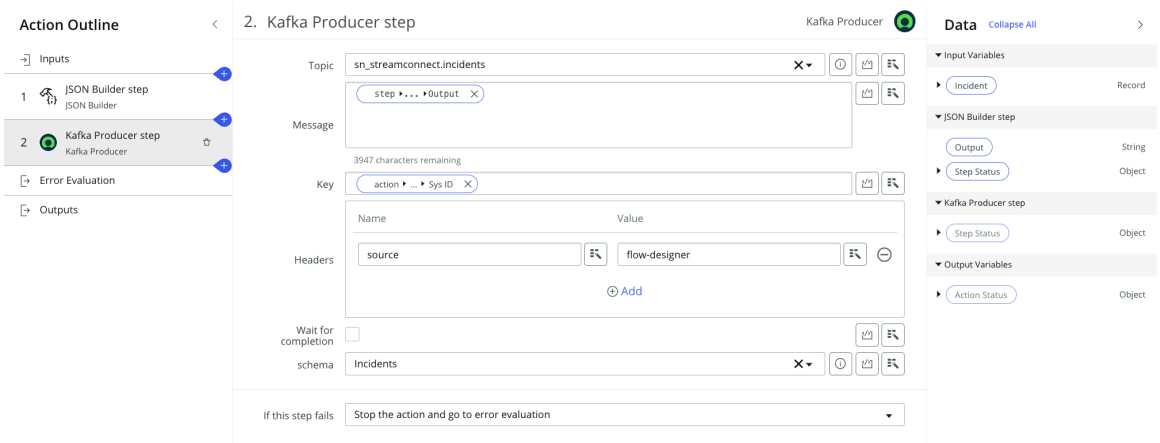
This step requires the ServiceNow Stream Connect Installer [com.glide.hub.stream_connect.installer] plugin.

Fields

Field	Description
Topic	Name of the topic to publish the message to. A topic stores messages of the same type. For example, a topic named Payments might store messages about recent payments. Select a topic from the drop-down list.
Message	Text of the message.
Key	Name of the key for a specific partition. Topics can be partitioned. Messages with the same key are stored in the same partition. For example, payment messages with a key of June would all be stored in the same partition of the Payments topic.
Headers	Headers for the message, in name-value pairs. For both the **Name** and **Value** fields, you can enter a value or use a data pill.
Wait For Completion	Option to require the flow to wait for the step to complete before continuing.
Schema	Reference to a schema table. Select a schema from the list. For information on schemas, see Schema management in Stream Connect. **Note:** The message you're sending in the **Message** field must adhere to the structure of the selected schema.
If this step fails	Option to go to error evaluation or continue running the next step. This option has no effect on the Step Status. Select one of the following. • **Stop the action and go to error evaluation**: Stop running the action at the current step and go to error evaluation. The Step Status object contains the error information returned by the step. • **Don't stop the action and go to the next step**: Ignore the failure and continue running the action from the next step. The Step Status object contains the error information returned by the step. Action error evaluation runs regardless of whether the action continues running.

Example

In this example, the step sends a message recording the creation of an incident. The message includes an incident identifier, a description, and a message header. The message is sent to the incidents topic, and stored in the partition with the name of the incident identifier.



Outputs

Field	Description	Data Type
Step Status	Object data pill containing runtime details about the step. Each step in an action returns a Step Status.	Object
Step Status > Code	Integer data pill indicating whether the step produced an error. By default, a value of 1 indicates that the step produced an error. A value of 0 indicates that the step ran successfully. You can't customize these codes.	Integer
Step Status > Message	String data pill containing the error message produced by the step or system operation. You can't customize this message.	String

Log step

Logs a message in the Workflow Studio log table sys_flow_log.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Log level	Level of importance of the log message. • Error • Warn • Info
Log message	Message to display in the Flow log [sys_flow_log] table. Enter text or drag data pills into the field.

Field	Description
	Note: The Workflow Studio design environment only supports entering 255 characters of text for a log message. The length limitation only applies to text entered directly into the input. Data pill values can exceed 255 characters in length. You can log values greater than 255 characters long by using either a data pill value or calling the GlideSystem - log(String message, String source) method from a script.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Look Up Record step

Look up a record from any table based on defined conditions.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Table	Select a table from the list.
Conditions	Set static or dynamic conditions to filter records. To define a static condition applied each time the action runs, define the conditions with the condition builder. To enable flow designers to dynamically apply conditions, define an input of type Conditions and drag-and-drop the input data pill into the **Conditions** field. When building a condition that looks up the value of a reference field, use a data pill that explicitly provides the Sys ID value. Ensure the condition has the format **[reference field][is][Reference type data pill->Sys ID]**. For example, both the Change and Incident tables contain a reference field to the User table. To look up change records where the requester is the caller from an incident record, create the condition **[Requested by][is][Trigger->incident record->Caller->Sys ID]**. **[Requested by][is][action->incident->Caller->Sys ID]** where **incident** is an input variable for an incident record.
Order by	Determines how to sort results when more than one record matches the defined conditions. Select the field you want to use to sort results.

Field	Description
Sort Type	Select whether to sort alphabetically in ascending or descending order.
If multiple records are found	Determines what is returned if more than one record matches the defined conditions. • Return only the first record • Fail the step
Don't fail on error	Determines whether to fail the flow if a record can't be found.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Record

Data type: *Record*

Record found based on the conditions you specified in the *Conditions* input.

Table

Data type: *Table*

Name of the table associated with the returned record.

Status

Data type: *Choice*

1 if a record was found successfully, and 0 if there was an error.

Error Message

Data type: *String*

Message containing details about why the record could not be found.

> **Note:**
> This output's value is only populated if the *Status* output's value is 0.

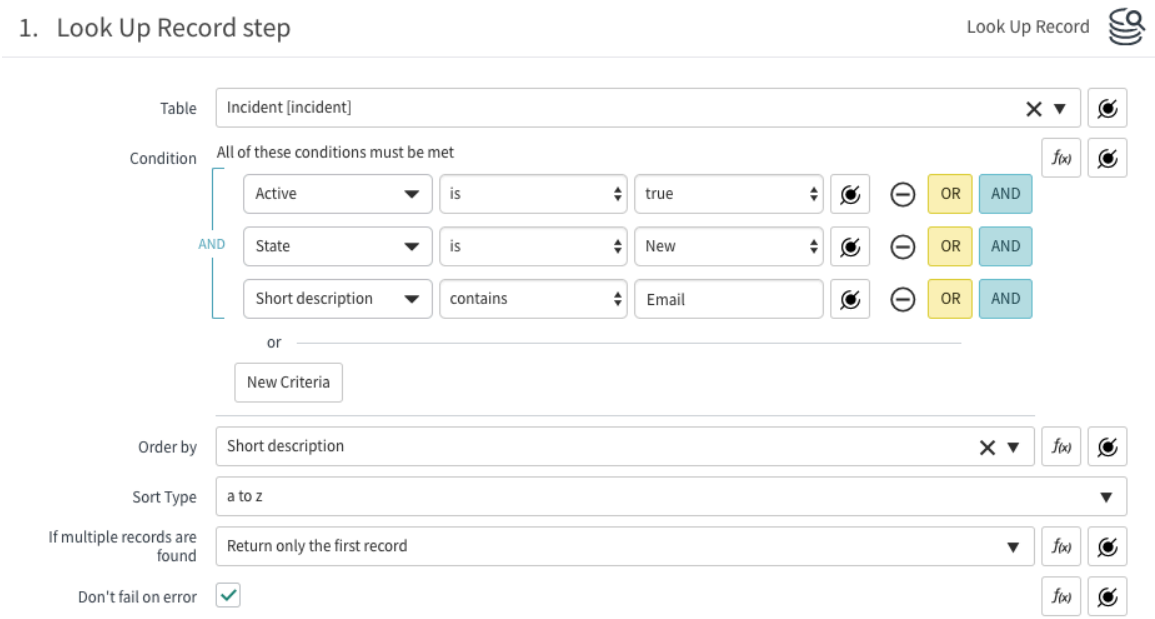
Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Example



Look Up Records step

Look up multiple records on any table using defined conditions.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Inputs

Provide a value for each input that your action needs. To add dynamic values, you can also drag pills from the Data panel or select them from the pill picker.

Table

Data type: *`Table Name`*

Table name containing the records you want to look up.

Conditions

Data type: *`Conditions`*

Field names and field values that you want to use to search for records. To use an inline script to specify conditions, consider using the GlideRecord and GlideQueryCondition classes to build your query. See GlideRecord - Global and GlideQueryCondition - Global.

Order by

Data type: *`Field Name`*

Field you want to use to sort results.

Sort Type

Data type: *`Choice`*

Option to sort alphabetically in ascending or descending order.

Max Results

Data type: *Integer*

The maximum number of record results the step can return.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your flow.

Records

Data type: *Records*

List of record Sys IDs found based on the lookup criteria that you provided. For more information, see Records.[Table] data type.

Table

Data type: *Table Name*

Table that contains the list of records.

Count

Data type: *Integer*

Number of records that the step returned.

General guidelines

Use these general guidelines when working with the Look Up Records action.

Process records with For Each flow logic

Use For Each flow logic to iterate through a list of records. For more information about using For Each flow logic, see For Each flow logic. Alternatively, you can use a Script step to process a list of records within a custom action.

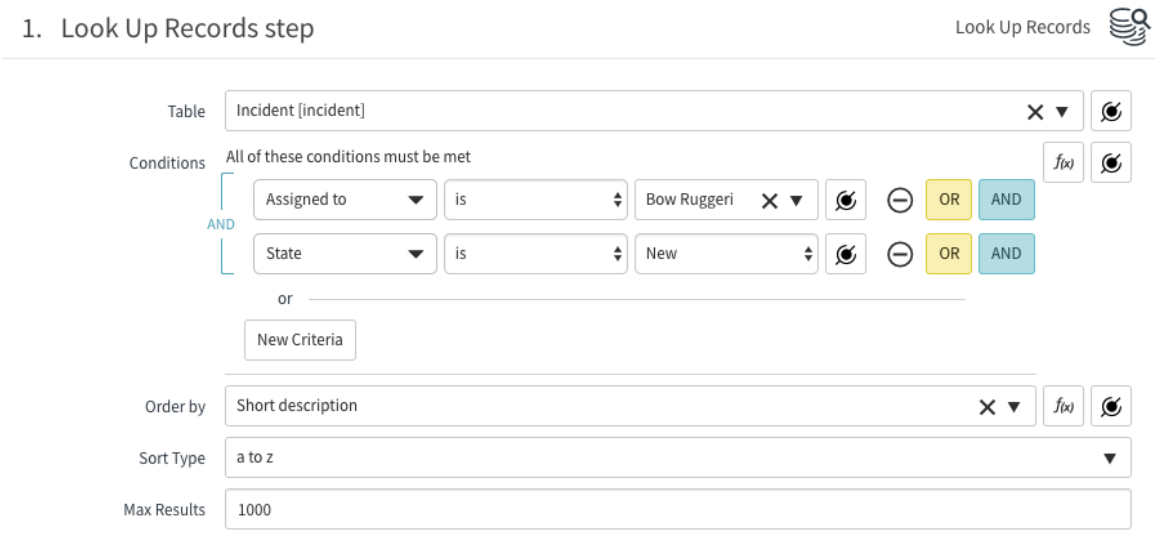
Set Max Results to improve performance

Set the Max Results input to 1000 records or lower to improve the performance of your action. The more records that the system has to look up, the more system resources it takes to identify and process them.

Use conditions to filter records

Use conditions to limit the number of records the step returns. The more specific conditions that you can provide, the better performance your action has.

Example



Notification step

Trigger a notification as a step within an action by selecting a record (such as an incident, change request, problem, or user record) to trigger a notification and defining the associated notification.

Notifications is a platform feature. Before triggering a notification as an action step in Workflow Studio, ensure that the notification is set up for use in the platform.

- When you Create an email notification, set the **Send when** field in the **When to send** tab of the Notification form to Triggered.
- Verify that your users have an active primary email channel and that all their notifications are active.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Record	Drag-and-drop an input record or a record from a previous step. This is the record that will trigger a notification. **Note:** Some notifications are not associated with a specific record or table, such as the **Passwords Require Updating** notification. If configuring such a notification, leave this field blank.
Table name	Read-only. Set to the table of the triggering record.
Notification	Select the notification to be triggered. The notifications that can be selected are associated with the table of the specified record. If no record was selected, you can select a notification that does not have an associated record or table. To create notifications, see Create an email notification

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Example

Payload Builder step

Enable action designers to easily create name-value pairs in JSON and XML payloads using dynamic data.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Note:
For JSON, this step is deprecated and replaced by the JSON Builder step.

Fields

Fields	Description
Name Value Pairs	The name-value pairs to include in the payload. Click the plus icon to add name-value pairs. Drag data pills into either field to produce dynamic payloads. The **Name** becomes a key in JSON and an element in XML. For example, suppose you create this name-value pair. • **Name**: `short_description` • **Value**: `[action]->[short_description]` When the system formats the name-value pair as JSON: `"short_description": "[action]->[short_description]"` When the system formats the name-value pair as XML: `<short_description>[action]->[short_description]</short_description>`
Omit if empty	The option to exclude a name-value pair if the value is empty or null.

Fields	Description
	Note: This field is only visible after clicking the down arrow to display advanced options.
Output Format	The payload file format. • **JSON**: Select to format the payload as a JSON document. • **XML**: Select to format the payload as an XML document.
Namespace	The XML namespace to apply to each element. For example, when the namespace is set to `incident`: `<incident:short_description>[action]->[short_description]</incident:short_description>` **Note:** This field is only visible when the **Output Format** is set to **XML**.
Include Outer Structure	The option to include or exclude a top level container appropriate to the output format. When the **Output Format** is JSON, curly braces contain the name-value pairs. When the **Output Format** is XML, a specified XML element contains the name-value pairs. For example, when the system formats the name-value pair as JSON: `{ "short_description": "[action]->[short_description]" }` When the system formats the name-value pair as XML: `<xml> <short_description>[action]->[short_description]</short_description> </xml>`
Send Empty Structure	The option to send valid JSON or XML structures when the payload is empty. Enable this option to include JSON or XML structural text in the payload. For example, when the system formats an empty structure as JSON: `{}` When the system formats an empty structure as XML: `<xml></xml>` Disable this option to produce an empty payload. Empty payloads can occur when you select the **Omit if empty** option for every name-value pair, and all name-value pairs in the payload produce empty values.

Fields	Description
Parent Node	The name of the XML element that contains the name-value pairs. The default parent node element is `xml`. **Note:** This field is only visible when the **Output Format** is set to **XML** and the option to **Include Outer Structure** is enabled.
Preview	The read-only payload the step produces.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

PowerShell step

Run PowerShell scripts on remote machines from your ServiceNow instance through a MID Server.

PowerShell is built on the Windows .NET Framework and is designed to control and automate the administration of Windows machines and applications. ServiceNow supports PowerShell 3.0 to 5.1. PowerShell 3.0 does not support Windows 2003 Server.

Note:
This step requires an Integration Hub subscription. For more information, see Legal schedules - Integration Hub overview.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Connection	Type of connection to use. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.

Field	Description
Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. **Note:** This field is available when **Use Connection Alias** is selected from the Connection list.
Credential Alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. **Note:** This field is available when **Define Connection Inline** is selected from the Connection list.
Host	Specify the fully-qualified domain name of the target host where the system runs the action step. For example, host.domain.com. **Note:** This field is only visible when the **Connection** is **Define Connection Inline**.
Port	Specify the communications port on which the target host listens for connections. For example, 5985. Leave blank to use the default port. **Note:** This field is only visible when the **Connection** is **Define Connection Inline**.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify.

Field	Description
	A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Connection list.
MID Application	Specify the application the MID Server must support to be eligible for selection. The system runs the action step from a MID Server that supports the selected application. If you use a data pill for this field, the pill must reference the MID Application's name, not the MID Application record. This field is available when **Define Connection Inline** is selected from the Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capabilities the MID Server must support to be eligible for selection. The system runs the action step from a MID Server that supports the selected capabilities. This field is available when **Define Connection Inline** is selected from the Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill containing a sys_id reference to a MID Server [ecc_agent_list] record. This field is available when **Define Connection Inline** is selected from the Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Remoting Type	The location where the PowerShell script runs such as the MID or a remote server. • **Explicit Remoting (Most Common)**: Establish a connection with and run the script on a remote server. • **Implicit Remoting (Advanced)**: Run a script on a MID Server while importing necessary modules from a remote server. If selected, define the **Remote name prefix** and **Modules to import** fields. For optimal performance, only import modules necessary to the step. If blank, all available modules are imported from the server. • **Run on a MID Server or have your script establish a remote session**: Run a script directly on a MID Server, or define remoting specifications within the script. This value is the default. **Note:** To invoke a function in a PowerShell script command or PowerShell script file, the command must define the function param block if the function has input parameters. This requirement applies to explicit and implicit remoting. For additional information on param block, see Microsoft's documentation on Windows Powershell parameters at https://technet.microsoft.com/.

Field	Description
Remote name prefix	The file path, excluding file names, to the modules to load from the remote server. **Note:** This field is only visible when the **Remoting Type** is **Implicit Remoting (Advanced)**.
Modules to import	The comma-separated list of modules to import from the remote server at the defined file path. **Note:** This field is only visible when the **Remoting Type** is **Explicit Remoting (Most Common)** or **Implicit Remoting (Advanced)**.
Test PowerShell Step	Button to test the configured credential for the PowerShell step. For more information, see Test a credential for the PowerShell step.
Script type	The type of script to run on the PowerShell host. • **Inline script**: Enter the script to run in the **Command** field of the step. • **MID Server Script File**: Select the PowerShell script to run from the MID Server Script Files [ecc_agent_script_file] table. This is the default value and separates scripting logic from the action, enabling you to update the script without having to modify and redeploy the action.
MID Server Script	Pre-defined PowerShell script from the MID Server Script Files table [ecc_agent_script_file]. **Note:** This field is only available if the **Script type** is **MID Server Script File**.
Script path	Read-only path to the selected MID Server script. **Note:** This field is only visible when the **Script type** is **MID Server Script File**.
Input variables	The optional name-value pairs that represent the values of PowerShell script variables. You can use action inputs and data from other steps within the PowerShell script. Define the following fields for each variable: • **Name**: The name of the script variable to pass a value to. The name cannot match a reserved or prohibited PowerShell variable. Some variable names are reserved for internal processing and should not be used as input variables. See **Reserved variables** in **PowerShell scripting variables**.

Field	Description
	• **Type**: The type of PowerShell variable. Select plain text, encrypted, or boolean. If encrypted is selected, the value appears in plain text in this field and is only encrypted when it passes to the ECC Queue. • **Value**: The value to map to the variable. Manually enter a value, or drag a data pill into the field.
Command	The inline PowerShell script to run on the target host. **Note:** This field is only visible when the **Script type** is **Inline script**.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

PowerShell scripting variables

To access input variables from the **Command** field, you must call them using special syntax. The syntax you use depends on the value of a system property. If the **Remoting Type** is **Run on a MID Server or have your script establish a remote session**, some reserved variables are available in addition to input variables.

Input variable syntax

By default, prefix variable names with a *$* character. For example, if an input variable is named **message**, use *$message* to access the variable in script.

If the *mid.powershell.command.script.parameter_passing* parameter is set to false, prefix the variable name with *$env:SNC_*. For example, if an input variable is named **message**, use *$env:SNC_message* to access the variable in script. To learn more about the *mid.powershell.command.script.parameter_passing* parameter, see MID Server parameters.

Reserved variables

When the **Remoting Type** is **Run on a MID Server or have your script establish a remote session**, the following variables are available for use in script. Reserved variables cannot be used as custom input variable names.

Reserved variable	Description
$computer	Host IP address defined in the Connection alias record.
$cred	Credential object that contains the credentials defined in the connection record. Use this variable with any PowerShell cmdlet that supports the credential parameter.

Reserved variable	Description
	For example, `New-PSSession -credential $cred`.
`$log_info`	If the *`mid.property.powershell.log_info`* property is set to true, adds logging information to a PowerShell script.

The following variable names are reserved for internal processing and should not be used as input variables.

- *`script`*
- *`useCred`*
- *`isMid`*
- *`isDiscovery`*
- *`debug`*
- *`user`*
- *`password`*
- *`executingScriptDirectory`*
- *`midScriptDirectory`*
- *`hresult`*

REST step

Send an outbound REST web service request to an external system.

Note:
REST step is not available in the base system and requires the ServiceNow® Integration Hub subscription. After the required plugin is activated, the step is visible under Integrations.

Outbound REST web service is a platform feature that enables you to retrieve, create, update, or delete data on a web services server that supports the REST architecture.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Connection	Type of connection to use. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information

Field	Description
	profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. **Note:** This field is available when **Use Connection Alias** is selected from the Connection list.
Credential Alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. **Note:** This field is available when **Define Connection Inline** is selected from the Connection list.
Use MID	Option to use a ServiceNow® MID Server to run the REST step. Select this check box to display the **MID Application** and **Capabilities** fields. **Note:** The system doesn't log REST request, response, and parameter runtime data sent via a MID server in the same way that Outbound web services logging occurs. Instead, you can view this data in the Flow execution details.
Base URL	Base URL for the REST request.

Field	Description
	• If **Use Connection Alias** is selected, this field displays the base URL associated with the alias. You can override the base URL by clicking the lock icon () and entering your own. • If **Define Connection Inline** is selected, enter a base URL for the connection.
Test REST Step	Button to test the REST step. To test, select the **Test REST Step** button. Enter any required input values and select the **Run Test** button. After the test runs, any step outputs or error messages are displayed in the Test Results section of the test window.
Connection Timeout	Number of milliseconds the system waits for a successful host connection. If the step does not make a successful connection during this time, the connection request times out. If **Define Connection Inline** is selected, enter a timeout value for the connection. Leave this field empty to use the system default connection timeout value. **Note:** Avoid setting the **Connection Timeout** value to zero, as this may cause a stale connection.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Connection list, and **Use MID** is checked.
MID Application	Capabilities the MID Server must support to be eligible for selection. The system runs the action step from a MID Server that supports the selected capabilities. This field is available when **Define Connection Inline** is selected from the Connection list, **Use MID** check box is enabled, and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capabilities the MID Server must support to be eligible for selection. The system runs the action step from a MID Server that supports the selected capabilities. This field is available when **Define Connection Inline** is selected from the Connection list, **Use MID** check box is enabled, and **Auto-Select MID Server** is selected from the MID Selection list.

Field	Description
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Connection list, **Use MID** check box is enabled, and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Connection list, **Use MID** is checked, and **Specific MID Cluster** is selected from the MID Selection list.
Build Request	Option to create the request manually, import an OpenAPI Specification, or import a REST message. • **Manually**: Create action inputs and complete the REST step form manually. • **From OpenAPI specification**: Import an OpenAPI Specification to generate action inputs and complete the REST step form. For more information, see OpenAPI support in the REST step. • **From REST Message**: Import a Platform REST message. For more information, see Import a REST message into a REST step.
API Source	Option to select an OpenAPI Specification used to construct the request, or select **Import OpenAPI** to import a new OpenAPI Specification. You can import specifications by providing a URL to the YAML or JSON, or copying and pasting content. **Note:** This field is available when you select **From OpenAPI specification** from the Build Request list.
API Operation	Option to select an operation from the list. Available operations are provided by the OpenAPI Specification in the **API Source** field. **Note:** This field is available when you select **From OpenAPI specification** from the Build Request list.
REST Message	Name of the REST message to import. Select a REST message from the list. **Note:** This field is available when you select **From REST Message** from the Build Request list.
REST Message Function	Name of the function to import from REST message. The available options are determined by the HTTP methods associated with the selected REST message.

Field	Description
	Note: This field is available when you select **From REST Message** from the Build Request list.
Import REST Message	Button to import the REST message. **Note:** This field is available when you select a REST message from the **REST Message** field.
Resource Path	Path for the resource.
HTTP Method	HTTP method used to process the request. • **GET** • **POST** • **PUT** • **PATCH** • **DELETE**
Query Parameters	Name-value pairs to pass to the REST endpoint. You can create these parameters manually, or drag input variables into the parameter fields, and then assign a value. Support REST step requests that contain duplicate query parameter names. If you create a REST request that contains duplicate query parameter names, Workflow Studio adds the query parameters to the request in the same order as you defined them. **Note:** When importing an OpenAPI Specification, the system adds all parameters and headers present in the specification to the REST step. Review the final REST step values and remove parameters you do not want to send in the request. For example, if the API accepts content type headers for both JSON and XML, the system adds both headers to the REST step. Remove one of the headers depending on the content type you want to receive in the response.
Headers	Headers to send with the request. You can create headers manually, or drag input variables into the parameter fields, and then assign a value. Support REST step requests that contain duplicate request headers. If you create a REST request that contains duplicate request headers, the headers are sent in the same order as you defined them.

Field	Description
	Note: When importing an OpenAPI Specification, the system adds all parameters and headers present in the specification to the REST step. Review the final REST step values and remove parameters you do not want to send in the request. For example, if the API accepts content type headers for both JSON and XML, the system adds both headers to the REST step. Remove one of the headers depending on the content type you want to receive in the response.
Request Type	Format of the request. Options include. • **Text**: A request in JSON, XML, or other text format. • **Binary**: A request in a binary file format. • **Multipart**: A request consisting of multiple content types. • **Form URL-Encoded**: A request in a URL-encoded query. **Note:** This field is editable when the **HTTP Method** is **POST**, **PUT**, **PATCH**, or **DELETE**.
Request Body [Text]	Body of the request in JSON or XML format. The flow execution details display the response body as either a link to the embedded text viewer or the sys_id of the attachment record containing the response. **Note:** This field is editable if you select **Text** from the Request Type list.
Attachment	Attachment record that contains the request. You can look up or create this record in a prior step and define it as an input variable. Create it by using the JSONStreamingBuilder and XMLStreamingBuilder APIs in the Script step. **Note:** This field is available when you select **Binary** from the Request Type list.
Name, Part Type, Value	Content of a multipart request. For each part, specify its name, part type, and value either using the fields individually or using an inline script for all the parts. You can specify the multipart values by clicking the toggle scripting icon ( ) and editing the script. For more information about inline scripting, see Inline scripts. • **Name**:The name of the part. It can be any valid string. • **Part Type**: The type of the part. Select either **Text** or **File**.

Field	Description
	○ **Text**:The text for the part. Once **Text** is selected, you can specify the content type. ○ **File**:The file for the part. When **File** is selected, the **Value** must be the sys_id of the attachment record containing the content. You can look up this record in a prior step or define it as an input value. Once **File** is selected, you can specify the file name and content type. ▪ For **Set Filename**, select **From Attachment** to use the attached record's file name, or select **From Filename input** to enter your own. ▪ For **Set Content Type**, select **From Attachment** to use the attached record's content type, or select **From Content Type input** to enter your own. • **Value**: The content of the part. For text, the value is the text content. For a file, the value is the sys_id of the attachment record containing the content. **Note:** These fields are available when you select **Multipart** from the Request Type list.
Name, Value	Content of a form URL-encoded request. Specify each part of the URL-encoded request with a name-value pair using the fields individually or using an inline script for all parts. You can specify the form URL-encoded values inline script by clicking the toggle scripting icon( ) and editing the script. For more information about inline scripting, see Inline scripts. **Note:** This field is available when you select **Form URL-Encoded** from the Request Type list.
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This check box is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.
Save As Attachment	Option to specify whether to save the response as a record in the Attachment [sys_attachment] table.
Attachment File Name	Name of the attachment created by the REST response. For example, `rest-response.txt`. **Note:** This field is available when **Save As Attachment** is selected.

Field	Description
Attachment File Record	Target record to which the attachment is associated. The target record must be a data pill of type **Record**. For example, a specific incident record. You can look up this record in a prior step or define it as an input variable. The flow execution details display the sys_id of the associated record. **Note:** This field is available when **Save As Attachment** is selected.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

REST response size limits

By default, the system limits the size of REST responses that are not saved as attachments to 5 MB. Direct REST responses that exceed this limit generate an error. To support larger response sizes, either save the response as an attachment or increase the response size limit with the *glide.pf.rest.response_payload_max_size* system property. This system property supports a maximum value of 10240 KB, which limits the REST response to 10 MB in size.

Parsing a REST response

REST API calls return data in the Response Body. The response body data is usually structured in JSON or XML format. You can use a Script step to parse the structured data into variables to use elsewhere in the action or in a flow. There is also an XML parser step to parse a response body that's in an XML format.

The general strategy to get extracted data from the response is to do the following.

1. Review the response body to select the data to return.
2. Create input and output variables in the Script step.
   - Create an input variable to pass in the Response Body from the REST step.
   - Create output variables to return data from the response.
3. Create a script to parse and map data.
   - Use the `JSON.parse()` method in a Script step to parse a JSON response body.
   - Map the parsed data to the output variables.
4. Create action outputs for the output variables to make the data available to a flow.

For a detailed example, see the section on Parsing a REST Response in the REST in IntegrationHub developer training (Xanadu).

Script step

Add custom JavaScript to execute within a reusable action. While most core actions and steps fit common use cases, you can build a Script step to execute behavior not satisfied by the core steps.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Note:
Integration Hub See Request Integration Hub for information about Integration Hub usage and subscriptions.

Fields

The Script step includes separate input and output variables that enable you to map JavaScript data to Workflow Studio data. By defining input and output variables within the step, you can define what Workflow Studio data is available within your script, and which scripting variables are available to other steps in your action.

Field	Description
Required Runtime	The runtime environment required to support the script. Choices include: • **Instance**: The action step runs the script from instance. Select this option when the script needs access to the ServiceNow API or instance data. This is the default value. • **MID**: The action step runs the script from the MID Server. Select this option when the script needs access to MID Server script files and APIs. Selecting this option displays the **Select MID Server Using** field. • **Vanilla (Core JavaScript)**: The action step runs the script from either the instance or MID Server. Select this option when the script only needs the core JavaScript APIs and not the ServiceNow API or instance data. The runtime you select determines the JavaScript objects and methods displayed in the Context-sensitive help. **Note:** This field is only visible when Integration Hub is activated.
Select MID Server Using	Specify the MID Server selection process to use. Choices include:

Field	Description
	• **Any MID**. The system runs the action step from any available MID Server. • **Use Connection Alias**. The system runs the action using the connection alias you specify. Selecting this option displays the **Connection Alias** field. • **Use Inline Selection**. The system runs the action using the connection details you specify in the action. Selecting this option displays the **Host**, **MID Application**, and **Capabilities** fields. **Note:** This field is only visible when Integration Hub is activated, and you select **MID** from **Required Runtime**.
Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. Only aliases of connection type Basic are supported. **Note:** This field is only visible when Integration Hub is activated, and you select **Use Connection Alias** from **Select MID Server Using**.
Host	The fully-qualified domain name of the MID Server where the system runs the action step. For example, mid-server.domain.com. **Note:** This field is only visible when Integration Hub is activated, and you select **Use Inline Selection** from **Select MID Server Using**.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **MID** is selected from the Required Runtime list, and **Use Inline Selection** is selected from the Select MID Server Using list.

Field	Description
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **MID** is selected from the Required Runtime list, and **Use Inline Selection** is selected from the Select MID Server Using list.
MID Application	Specify the application the MID Server must support to be eligible for selection. The system runs the action step from a MID Server that supports the selected application. This field is only visible when Integration Hub is activated, **Auto-Select MID Server** is selected from the MID Selection list, and you select **Use Inline Selection** from **Select MID Server Using**.
Capabilities	Capabilities the MID Server must support to be eligible for selection. The system runs the action step from a MID Server that supports the selected capabilities. This field is only visible when Integration Hub is activated, **Auto-Select MID Server** is selected from the MID Selection list, and you select **Use Inline Selection** from **Select MID Server Using**.
Specific MID Server	Data pill of the required MID Server. This field is only visible when Integration Hub is activated, **Specific MID Server** is selected from the MID Selection list, and you select **Use Inline Selection** from **Select MID Server Using**.
Input variables	Name-value pairs that represent data from the action, enabling you to use action inputs and data from other steps within a script.
Script	Script that executes within the action. To access input and output variables in your script, use the global objects `inputs` and `outputs`. For example, `inputs.myVariable`. **Note:** Script step input and output names can't include any of the following reserved system names: • `sys_id` • `sys_created_by` • `sys_created_on` • `sys_updated_on` • `sys_updated_by` • `sys_mod_count` In general, don't create a variable that has the same name as a system field. The Script step may confuse such an input variable for a field name and use the wrong value. The Script step always converts data stored in the `inputs` and `outputs` global objects into strings. If your Script step needs to work with JSON data, you can use the `inputs` global object to convert the JSON data into a string. Alternatively, you can define a JavaScript variable as a string rather than a JavaScript object. For example, this script illustrates two ways you can output JSON data. `(function execute(inputs, outputs) {` `  outputs.json_object_1 = inputs.json_input;`

Field	Description
	`` var array_of_objs = '[{"name1":"value1"}, {"id":"abcd"}]'; outputs.json_object_2 = array_of_objs; })(inputs, outputs); ``` By default, Workflow Studio run scripts on the instance. To run script from a MID Server requires an Integration Hub subscription. Workflow Studio runs script from the domain from which it is triggered or initiated. See Domain separation and Workflow Studio. For available classes and methods, see the JavaScript API context-sensitive help or the API reference.
Output variables	Map JavaScript output to Workflow Studio data pills. Define output variables when you want other steps in the action to use the script output.

Action error evaluation

If this step fails

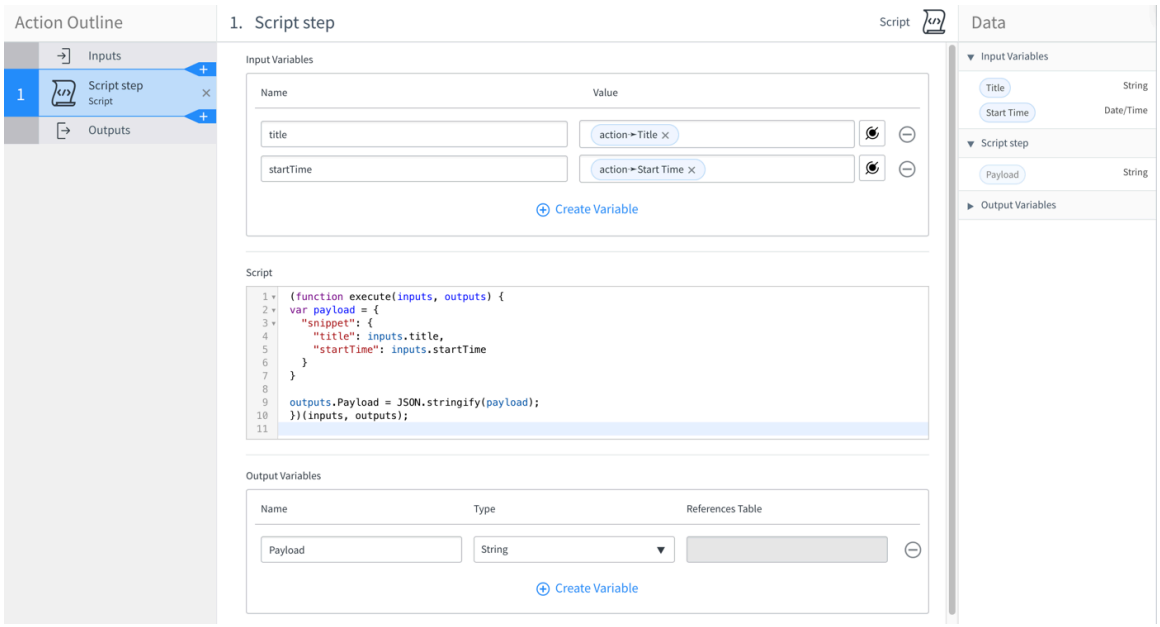
Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Example

This example builds a JSON payload that can be easily updated or changed and added to a subsequent REST step.

Note:
REST step is not available in the base system and requires the ServiceNow® Integration Hub subscription.



By creating an output variable that represents the payload, you can drag the **[Payload]** data pill into the REST step **Body** field.

Send Email step

Send an email to specified users or groups as an action in a flow.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Inputs

Provide a value for each input that your flow needs. To add dynamic values, you can also drag and drop pills from the Data panel or select them from the pill picker.

Target Record

Data type: *Document ID*

Record that the email is associated to. When a user sends a reply to your email, the target record is updated with the reply email content.

Table

Data type: *Table Name*

Table containing the target record.

To

Data type: *String*

The main recipients of the email. Enter a list of user email addresses separated by commas or white spaces. You can also drag data pills that contain email addresses into the field, such as a User or Group record. For example, if you want to send an email to the group assigned to the incident, drag the **[Assignment group]** data pill from the data panel.

To send email to a group, you must provide a **Group email** address. To send email to group members, the group must have the **Include members** option enabled.

> **Note:**
> The number of email recipients must be equal to or less than the maximum number set by the glide.email.smtp.max_recipients system property. See Minimize SMTP Recipient Quantity [Updated in Security Center 1.3] for information about setting this value.

CC

Data type: *String*

Additional recipients copied on this email. Enter a list of user email addresses separated by commas or white spaces. You can also drag data pills that contain email addresses into the field.

BCC

Data type: *String*

Additional recipients of this email, who are visible only to the sender (blind copied). Enter a list of user email addresses separated by commas or white spaces. You can also drag data pills that contain email addresses into the field.

Subject

Data type: *String*

Subject of the email. You can enter text or drag data pills into the field.

Body

The content of the message body. You can enter text or drag data pills into the field. The editor formatting options add inline styles. There is no style sheet associated with an email body. You can add your own inline style sheet to this HTML input.

> **Note:**
> You can partially reuse content from the Email Template [sysevent_email_template] table with some manual editing. The Send Email action can't process Email template variables, which are strings that start with the dollar sign character and include content in curly braces. You must replace email variables with references to data pills. For example, replace the string `${number}` with a data pill reference to an incident record number.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Outputs

These outputs appear in the Data panel. You can use them as inputs elsewhere in your action or flow.

Email

Data type: *Record*

Email record created.

Testing the email step

To verify that the email was generated when testing the action, review the email record in the Email [sys_email] table. The **Headers** field indicates whether the email was successfully generated. For

example:
ACL restrictions apply to the Send Email action. If you configured your flow to run as the user who initiates the session, ensure that the user can access email. To test access controls for a Send Email action, impersonate a typical email sender and manually trigger the flow.

SFTP step

Create a reusable action to manage files and directories on an SFTP server and to move files from one SFTP server to another.

Note:

- This step requires an Integration Hub subscription. For more information, see Legal schedules - IntegrationHub overview. For more information about the ServiceNow® Integration Hub subscription packages, see Integration Hub usage and subscription. After the required plugin is activated, the step is visible under Integrations.
- The SFTP step runs only on a ServiceNow® MID Server with SSH capabilities. Activate the plugin, Integration Hub in Workflow Data Fabric Professional (com.glide.hub.integrations.professional) to use the JDBC capability for the MID Server. For more information, see Request an Integration Hub plugin.
- The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.

Roles and availability

The SFTP step is available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Prerequisite

Activate the Managed File Transfer Extensions for the SFTP Step (com.glide.hub.action_step.sftp_mft) plugin.

SFTP commands

- Copy File
- Copy Directory
- Create Directory
- Get File List
- Remove File
- Remove Files
- Delete Directory
- Rename File or Directory
- Set File Attributes
- Copy Attachments To SFTP Server
- Copy Files To This Instance

Note:
The SFTP commands can be performed on a maximum of 10,000 files at a time.

Copy File

Copies a file from the source SFTP server to target SFTP server.

Fields

Field	Description
Connections	

Field	Description
Source Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Source Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Source Connection list.
Target Connection	Type of connection to use to connect to the target SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Target Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases.
Source Credential alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Source Connection list.

Field	Description
	Note: The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server that contains the files you wish to copy.
Port	Port number to communicate with the server.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Command Details	
Source Path	Full path to the file in the source server you wish to copy. For example, `/root/doc/tempwsdl.rtf`.
Target Path	Full path to the file in the target server you wish to copy the contents. For example, `/root/doc/attribute.rtf`. In this case, the contents of the tempwsdl.rtf are copied to the file, attribute.rtf.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.

Field	Description
Policy for Alias	
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Copy Directory

Copies a directory from the source SFTP server to the target SFTP server.

Note:
If you wish to use the Managed File Transfer feature while copying a directory, activate the ServiceNow IntegrationHub Action Step - MFT (com.glide.hub.action_step.mft) plugin.

Fields

Field	Description
Connections	
Source Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Source Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Source Connection list.
Target Connection	Type of connection to use to connect to the target SFTP server.

Field	Description
	• **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Target Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases.
Source Credential alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Source Connection list. **Note:** The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server that contains the files you wish to copy.
Port	Port number to communicate with the server.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.

Field	Description
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Command Details	
Source Path	Full path to the file in the source server you wish to copy. For example, `/root/doc/tempwsdl.rtf`.
Target Path	Full path to the file in the target server you wish to copy the contents. For example, `/root/doc/attribute.rtf`. In this case, the contents of the tempwsdl.rtf are copied to the file, attribute.rtf.
Include Files	List of target files to remove. This is a semicolon separated list that accepts wild cards, such as `*.txt`. **Note:** • If no value is provided, subfolders in the specified directory are deleted. • If a value is provided, subfolders are not deleted even if they are empty.
Exclude Files	List of target files that should not be removed. This is a semicolon separated list that accepts wild cards, such as `*.txt`. **Note:** • If no value is provided, subfolders in the specified directory are deleted. • If a value is provided, subfolders are not deleted even if they are empty.
Include Subfolders	Option to copy subfolders in the source directory.
Managed File Transfer	
Target File Name	Name of the target file.
Target Directory Name	Name of the target directory.
Datetime Format ${DateTime}	Format in which the date and time should be appended to the file name upon copying to the target server.

Field	Description
Preserve File Attributes	Option to preserve the file attributes upon copying files to the target directory.
Apply Move Conditions	Option to specify conditions while moving files.
Minimum File Size (Bytes)	Minimum size requirements to move files. **Note:** This field is available when **Apply Move Conditions** is enabled.
Maximum File Size (Bytes)	Maximum size requirements to move files. **Note:** This field is available when **Apply Move Conditions** is enabled.
File is Newer than	Files created after this date are moved. **Note:** This field is available when **Apply Move Conditions** is enabled.
File is Older than	Files created before this date are moved. **Note:** This field is available when **Apply Move Conditions** is enabled.
Move Order	Order in which the files should be moved. **Note:** This field is available when **Apply Move Conditions** is enabled.
Sort Order	Order in which the files should be sorted. **Note:** This field is available when **Apply Move Conditions** is enabled.
Duplicate File Action	Action to be performed when a duplicate file exists in the target directory. **Note:** This field is available when **Apply Move Conditions** is enabled.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.

Field	Description
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.
Managed File Transfer Error Cleanup	
Upon Failure, Remove Files on Target	Option to remove files from the target SFTP server when the copy command fails.
Upon Success, Remove Files from Source	Option to remove files from the source SFTP server when the copy command is executed successfully.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Create Directory

Creates a new directory on an SFTP server.

Fields

Field	Description
Connections	
Source Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Source Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as

Field	Description
	a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Source Connection list.
Source Credential alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Source Connection list. **Note:** The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server where you wish to create directory.
Port	Port number to communicate with the server.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Command Details	

Field	Description
Source Path	Path of the directory you wish to create.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Get File List

Returns a list of files from a given directory and its subfolders on an SFTP server.

Note:
This SFTP command runs only on a ServiceNow® MID Server.

Fields

Field	Description
Connections	
Source Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Source Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials,

Field	Description
	see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Source Connection list.
Source Credential alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Source Connection list. **Note:** The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server that contains the files you wish to list.
Port	Port number to communicate with the server.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Command Details	

Field	Description
Source Path	Path of the directory that contains the files you wish to list.
Include Files	List of target files to include. This is a semicolon separated list that accepts wild cards, such as `*.txt`.
Exclude Files	List of target files to exclude. This is a semicolon separated list that accepts wild cards, such as `*.txt`.
Include Subfolders	Option to specify if files from subfolders are included in the list.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Remove File

Removes a file on an SFTP server, including subfolders, when configured.

Fields

Field	Description
Connections	
Source Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.

Field	Description
Source Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Source Connection list.
Source Credential alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Source Connection list. **Note:** The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server that contains the file you wish to remove.
Port	Port number to communicate with the server.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.

Field	Description
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Command Details	
Source Path	Path of the directory that contains the file you wish to remove.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Remove Files

Remove files on an SFTP server, including subfolders, when configured.

Fields

Field	Description
Connections	
Source Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.

Field	Description
Source Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Source Connection list.
Source Credential alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Source Connection list. **Note:** The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server that contains the file you wish to remove.
Port	Port number to communicate with the server.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.

Field	Description
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Command Details	
Source Path	Path of the directory that contains the files you wish to remove.
Apply Remove Conditions	Option to specify conditions to remove files.
Include Files	List of target files to remove. This is a semicolon separated list that accepts wild cards, such as `*.txt`. **Note:** • If no value is provided, subfolders in the specified directory are deleted. • If a value is provided, subfolders are not deleted even if they are empty.
Exclude Files	List of target files that should not be removed. This is a semicolon separated list that accepts wild cards, such as `*.txt`. **Note:** • If no value is provided, subfolders in the specified directory are deleted. • If a value is provided, subfolders are not deleted even if they are empty.
Include Subfolders	Option to specify if files from the subfolders should be removed.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Delete Directory

Deletes directory on an SFTP server, including subfolders, when configured.

Fields

Field	Description
Connections	
Source Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Source Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Source Connection list.
Source Credential alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Source Connection list. **Note:** The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server that contains the directory you wish to delete.
Port	Port number to communicate with the server.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options.

Field	Description
	• **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Command Details	
Source Path	Path of the directory you wish to remove.
Include Subfolders	Option to specify if the subfolders should be deleted. **Note:** If this option is selected, all subfolders are deleted. Else, only empty subfolders are deleted.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Rename File or Directory

Renames a file or directory on an SFTP server.

Fields

Field	Description
Connections	
Source Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Source Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Source Connection list.
Source Credential alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Source Connection list. **Note:** The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server that contains the file or directory you wish to rename.
Port	Port number to communicate with the server.

Field	Description
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Command Details	
Source Path	Full path to the file or directory.
Target Path	Full path with the renamed file or directory.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Set File Attributes

Sets common file attributes, such as timestamps, size, permissions, and UID/GID, for a file or directory on an SFTP server.

A good practice is to use the Get File List command to return a list of files and their attributes first. Then, after you have moved the files from a source host to a target host, use the Set File Attributes command to set the source file attributes on the target file.

Fields

Field	Description
Connections	
Source Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Source Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Source Connection list.
Source Credential alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Source Connection list.

Field	Description
	Note: The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server that contains the file you wish to remove.
Port	Port number to communicate with the server.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Source Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
Command Details	
Source Path	Path of the directory that contains the files you wish to remove.
User ID	User ID attribute to apply to the file or directory. The UID and GUID variables must be set together as a pair.
Group ID	Group ID attribute to apply to the file or directory. The UID and GUID variables must be set together as a pair.
Permissions (chmod)	File or directory permissions to set for the user and group specified. This value must be specified in Octal notation only. For example, `755`.

Field	Description
	Note: The permissions number is an internal value returned by the Get File List command.
Modified Timestamp (epoch)	Override the timestamp when the file or directory was last modified. **Note:** • Timestamp must be in epoch format. • Access and modification timestamps must be set together as a pair.
Accessed Timestamp (epoch)	Override the timestamp when the file or directory was last accessed. **Note:** • Timestamp must be in epoch format. • Access and modification timestamps must be set together as a pair.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Copy Attachments To SFTP Server

Copies the specified attachments from ServiceNow instance to an SFTP server.

Fields

Field	Description
Connections	
Connection	Type of connection to use to connect to the source SFTP server.

Field	Description
	• **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Connection list.
Credential Alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Connection list. **Note:** The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Name or IP address of the SFTP server to which the files you should be copied. This field is available when **Define Connection Inline** is selected from the Connection list.
Port	Port number to communicate with the server. This field is available when **Define Connection Inline** is selected from the Connection list.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify.

Field	Description
	• **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Server	Specific MID Server to run the SFTP step. This field is available when **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Connection list and **Specific MID Server** is selected from the MID Selection list.
Attachments to Copy	
Attachment Records	Records in your ServiceNow instance that you want to copy to SFTP server.
Copy to SFTP Target	
Target Path	Full path to the file in the target server you want to copy the contents. For example, `/root/doc/`. In this case, the attachments would be copied to this path on the server.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the **Connection** list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Copy Files To This Instance

Attaches the specified files in the SFTP server to the specified record in ServiceNow instance.

Fields

Field	Description
Connections	
Connection	Type of connection to use to connect to the source SFTP server. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Connection list.
Credential Alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Define Connection Inline** is selected from the Connection list. **Note:** The SFTP step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.

Fields (continued)

Field	Description
Host	Name or IP address of the SFTP server to which the files you should be copied. This field is available when **Define Connection Inline** is selected from the Connection list.
Port	Port number to communicate with the server. This field is available when **Define Connection Inline** is selected from the Connection list.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Source Connection list.
MID Server	Specific MID Server to run the SFTP step. This field is available when **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Source Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
MID Application	Option to use a MID Server to run the SFTP step. This field is available when **Define Connection Inline** is selected from the Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Connection list and **Specific MID Server** is selected from the MID Selection list.
SFTP Files To Copy	
Source Path	Path of the directory that contains the files you want to copy. For example, `/root/doc/`.
Include Files	List of target files to copy. This is a semicolon separated list that accepts wild cards, such as `*.txt`. **Note:** If no value is provided, subfolders in the specified directory are copied.

Fields (continued)

Field	Description
Exclude Files	List of target files that should not be copied. This is a semicolon separated list that accepts wild cards, such as `*.txt`.
Maximum File Size (KB)	Maximum size of the file that can be copied.
Maximum Number of Files	Maximum number of files that can be copied in a request.
Attach Files To This Target Record	
Target Record	Record in ServiceNow instance to which you want to attach the files.
Table	ServiceNow table in which the target record is saved.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This option is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy for Alias** is selected, you can override the default retry policy and select another existing retry policy based on your requirement.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

SSH step

The SSH step executes SSH commands on an external *nix system through a ServiceNow® MID Server. The step also stores scripts and commands for the *nix systems.

Note:

- This step requires an Integration Hub subscription. For more information, see Legal schedules - Integration Hub overview.
- Integration Hub supports ServiceNow SSH only.

Roles and availability

The SSH step is available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Sanitizing inputs

Escape all user inputs to eliminate the possibility of a malicious user executing arbitrary commands on your target server. Escape and validate data pills before the command field uses

them by sanitizing arguments using Sanitize shell arguments transform functions. This transform function category automatically appears when a data pill is dropped into the **Command** input.

Fields

Field	Description
Connection Details	
Connection	Type of connection to use. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This field is available when **Use Connection Alias** is selected from the Connection list.
Credential Alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. This

Field	Description
	field is available when **Define Connection Inline** is selected from the Connection list. **Note:** The SSH step requires a credential record that supports either SSH private key credentials or SSH credentials. This step does not support Windows credentials where the **Use MID Server service account** option is selected.
Host	Host name or IP address of the target server. This field is available when **Define Connection Inline** is selected from the Connection list.
Port	Port number to communicate with the server. This field is available when **Define Connection Inline** is selected from the Connection list. **Note:** Provide valid value between `1` and `65535`. If you provide any other value, the value will be set `22` by default.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify. • **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Connection list.
MID Application	Option to use a MID Server to run the SSH step. This field is available when **Define Connection Inline** is selected from the Connection list and **Auto-Select MID Server** is selected from the MID Selection list.
Capabilities	Capability of the MID Server. Select **SSH**. This field is available when **Define Connection Inline** is selected from the Connection list and

Field	Description
	Auto-Select MID Server is selected from the MID Selection list.
MID Server	Data pill of the required MID Server. This field is available when **Define Connection Inline** is selected from the Connection list and **Specific MID Server** is selected from the MID Selection list.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Connection list, and **Specific MID Cluster** is selected from the MID Selection list.
SSH Configuration	
Working Directory	Optional target directory on the target host where the command is run.
Command	Command that runs on the target directory. The command can also include MID Server scripts. See Advanced SSH script options for more information. **Note:** Escape and validate data pills before the command field uses them by sanitizing arguments using a preprocessing Script step. For more information, see Sanitizing arguments using the escape class and function.
Long Running	Option to disable the SSH connection timeout for commands that might take longer than the default time of 120 seconds to run. When selected, the engine detaches from the execution thread until completion.
Sudo Mode	Option to elevate privileges to execute the script.

For more information, see SSH credentials.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Advanced SSH script options

To run a MID Server script on the target host, specify the script type and pass the name of the script into the `${syncFile()}` parameter. The system uses this parameter to locate the named

script in the MID Server Script File [ecc_agent_script_file] table and run it on the target host. For example, a bash script can be expressed as:

```
bash ${syncFile("<MID script name>")} argument1 argument2
 argument3
```

A base script (`main_script.bash`) can reference another script (`my_include.bash`) as well as a separate file (`.my_profile`) located on the target host. Both scripts and the file referenced must be synced to the MID Server, using the `${syncFile()}` parameter, to execute properly.

```
source ${syncFile(".my_profile")}
cp ${syncFile("my_include.bash")} /usr/ssmith/my_include.bash
bash ${syncFile("main_script.bash")} one two three four five six
rm /usr/ssmith/my_include.bash
```

A Python example with inline comments might look like this:

```
set $LIB_DIR=/usr/bin;.
# Sync a file that is referenced inside myF5CreateLBPool.py
cp ${syncFile("specialFunctions.py")} ~/specialFunctions.py
# set up environment variables
source ${syncFile(".python_profile")}
# call script that sets up dependencies on the box from remote
 package repos
python ${syncFile("setupPythonDependencies.py")} pycontrol
# call a script that requires functions from the package as well
 as a function from myIncludedFile
python ${syncFile("myF5CreateLBPool.py")} snow_pool
 myActualValue
# user is responsible for their own cleanup
rm ~/specialFunctions.py
```

To see the list of available MID Server scripts, navigate to **MID Server** > **Script Files**.

SOAP step

Enable action designers to send outbound SOAP web service requests to external systems.

Note:
SOAP step is not available in the base system and requires the ServiceNow® Integration Hub subscription. After the required plugin is activated, the step is visible under Integrations.

Roles and availability

- Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.
- Action designers need the web_service_admin role to perform these web services tasks.
  - Select WSDL
  - Load new WSDL
  - Select a WS-Security policy
- The ServiceNow® MID Server doesn't support WS-Security policies.

Fields

Field	Description
Connection Details	
Connection	The type of connection to use. • **Define Connection Inline**: Define connection information within the action step. • **Use Connection Alias**: Define connection information using the Connection Alias table. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see Introduction to credentials, connections, and aliases.
Connection Alias	Connection & Credential alias record that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials and connection information profiles when using an action in multiple environments. Likewise, if the connection information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential value is displayed as a Password (2 Way Encrypted) data pill on the data panel. **Note:** This field is available when **Use Connection Alias** is selected from the Connection list.
Credential Alias	Credential alias that the system uses to run the action step. Users with the flow_designer or admin role can create or select an associated Connection record. Using an alias eliminates the need to configure multiple credentials when using an action in multiple environments. Likewise, if the credential information changes, you don't need to update your custom action. To learn more about connections and credentials, see credentials, connections, and aliases. The credential

Field	Description
	value is displayed as a Password (2 Way Encrypted) data pill on the data panel. **Note:** This field is available when **Define Connection Inline** is selected from the Connection list.
Use MID	Option to use a MID Server to run the SOAP step. Select this check box to display the **MID Selection**,**MID Application**, and **Capabilities** fields. **Note:** This field is available when **Use Connection Alias** is selected from the Connection list.
Endpoint	The URL endpoint for the SOAP request. If **Use Connection Alias** is selected, this field is read-only and displays the endpoint URL associated with alias. If **Define Connection Inline** is selected, enter an endpoint URL for the connection.
Test SOAP Step	Button to test the SOAP step. To test, select the **Test SOAP Step** button. Enter any required input values and select the **Run Test** button. After the test runs, any step outputs or error messages are displayed in the Test Results section of the test window.
Connection Timeout	Number of milliseconds the system waits for a successful host connection. If the step does not make a successful connection during this time, the connection request times out. If **Define Connection Inline** is selected, enter a timeout value for the connection. Leave this field empty to use the system default connection timeout value. **Note:** Avoid setting the **Connection Timeout** value to zero, as this may cause a stale connection.
MID Selection	Option to select a specific MID Server or MID Cluster. Choose any one of the following options. • **Auto-Select MID Server**: Your ServiceNow instance selects the MID Server without manual input. • **Specific MID Server**: Your ServiceNow instance uses MID Server that you specify.

Field	Description
	• **Specific MID Cluster**: Your ServiceNow instance uses the MID Cluster that you specify. A MID Cluster is a group of MID Servers that enables your ServiceNow instance to handle multiple integrations, and improve integration speed. For more information, see Configure a MID Server cluster. This field is available when **Define Connection Inline** is selected from the Connection list, and **Use MID** is checked.
MID Cluster	Data pill for the MID Cluster you want to use. This field is available when **Define Connection Inline** is selected from the Connection list, **Use MID** is checked, and **Specific MID Cluster** is selected from the MID Selection list.
Request Details	
Build Envelope	The method to use when building the SOAP envelope. • **From WSDL**: Select this option to display the **Select a WSDL** and **Operation** fields. • **Manually**: Select this option to manually enter or paste WSDL text.
Select a WSDL	The WSDL to use to build the SOAP envelope. Select an existing WSDL record or click **Load New WSDL** to download or manually enter a WSDL file. The selected WSDL populates the values of the **Operation**, **SOAP action**, and **SOAP Envelope** fields. **Note:** This field is available when you select **From WSDL** from the Build Envelope list.
Load New WSDL	Option to download or manually enter a WSDL file.
Operation	The operation to run from the selected WSDL. Each WSDL has its own list of available operations.
SOAP Action	The URL to run the SOAP action. If **Build Envelope** is set to **From WSDL**, this field is read-only and displays the URL to run SOAP action. If **Build Envelope** is set to **Manually**, enter a URL to run the SOAP action.
Request Type	Format of the request. Options include.

Field	Description
	• **Text**: A request in JSON, XML, or other text format. • **Binary**: A request in a binary file format.
SOAP Envelope	The XML text sent to the endpoint. If **Build Envelope** is set to **From WSDL**, the system adds the necessary XML for the **Operation** that you select. If **Build Envelope** is set to **Manually**, enter the XML text that you want to use. Enter record values in the appropriate SOAP envelope elements. For example, enter an incident short description in the `<short_description>` element. **Note:** This field is available when the **Request Type** is **Text**.
Attachment	Attachment record that contains the request. You can look up or create this record in a prior step and define it as an input variable. Create it by using the JSONStreamingBuilder and XMLStreamingBuilder APIs in the Script step. **Note:** This field is available when the **Request Type** is **Binary**.
Reset Envelope	Option to discard all manual changes that you made to the SOAP envelope. Select this check box to revert the SOAP envelope to its original state. **Note:** This field is available when you select **From WSDL** from the Build Envelope list.
New WSDL	
Name	The name of the WSDL record you want to create.
Import Method	The method to enter WSDL. • **Download from URL**: Select to display the **WSDL URL**, **User name**, and **Password** fields to retrieve the WSDL from an external source, typically the web service provider. • **Manually Populate WSDL Content**: Select to display the **WSDL Content** field to manually enter or paste WSDL text.
WSDL URL	The URL to the SOAP web service.

Field	Description
	Note: This field is available when you select **Download from URL** from the Import Method list.
User name	The user name to authenticate with the SOAP web service. **Note:** This field is available when you select **Download from URL** from the Import Method list.
Password	The password to authenticate with the SOAP web service. The system always masks passwords in the user interface and prevents exporting them as plain text. **Note:** This field is available when you select **Download from URL** from the Import Method list.
WSDL Content	The XML document describing the SOAP web service and its operations. **Note:** This field is available when you select **Manually Populate WSDL Content** from the Import Method list.
Import	Option to add the SOAP web service WSDL to the instance.
WS-Security	
Enable WS-Security Policy	Option to restrict the SOAP web service to a security policy. Select this check box to display the **Policy** field.
Policy	The policy record that you want to use to restrict web service connections. Select an existing policy record.
Retry Policy	
Enable Retry Policy	Option to enable the retry policy. For more information, see Retry policy.
Override Default Policy for Alias	Option to override the default retry policy. This check box is not applicable when **Define Connection Inline** is selected from the Connection list.
Retry Policy	Default retry policy associated with **Connection Alias**. If **Override Default Policy**

Field	Description
	for Alias is selected, you can override the default retry policy and select another existing retry policy based on your requirement.
Advanced Options	
Headers	The name-value pairs to include in the SOAP message as HTTP headers. Click the plus icon ⊕ to add headers. Add a **Name** and **Value** for each HTTP header.
Headers > **Omit if empty**	Option to exclude a header if the value is empty or null. **Note:** This check box is available after clicking the down arrow to display the advanced options.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

SOAP response size limit

The system limits the size of SOAP responses to 5 MB. Direct SOAP responses that exceed this limit generate an error. To support larger response sizes, increase the response size limit with the *glide.pf.soap.response_payload_max_size* system property. This system property supports a maximum value of 10 MB.

Update Multiple Records step

Look up and update multiple records as a single step. Using this step removes the need to separately look up a list of records and then process the list with a Script step. Set field values with a template or add and configure them using data pills.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Table	Select the table containing the records to look up and update.
Conditions	Define the filter conditions used to look up records.

Field	Description
Field Values	Set static or dynamic values of fields in the record. For example, to set the short description to a static value, select **Short description** and set the desired value. To add dynamic values, see Create a template value input. **Important:** The system does not support updating multiple journal fields such as the additional comments or work notes of a task record.
Order by	Select the field that you want to use to sort the records when more than one record matches the defined conditions.
Sort Type	Determine whether to sort the records alphabetically in ascending or descending order.
Run Business Rules and Workflow	Determine whether to call any business rules and workflows associated with the table.
Update System Fields	Select if you want to automatically update system fields such as **Updated by**.
Don't fail on error	Specify whether to continue running the flow when there is an error.

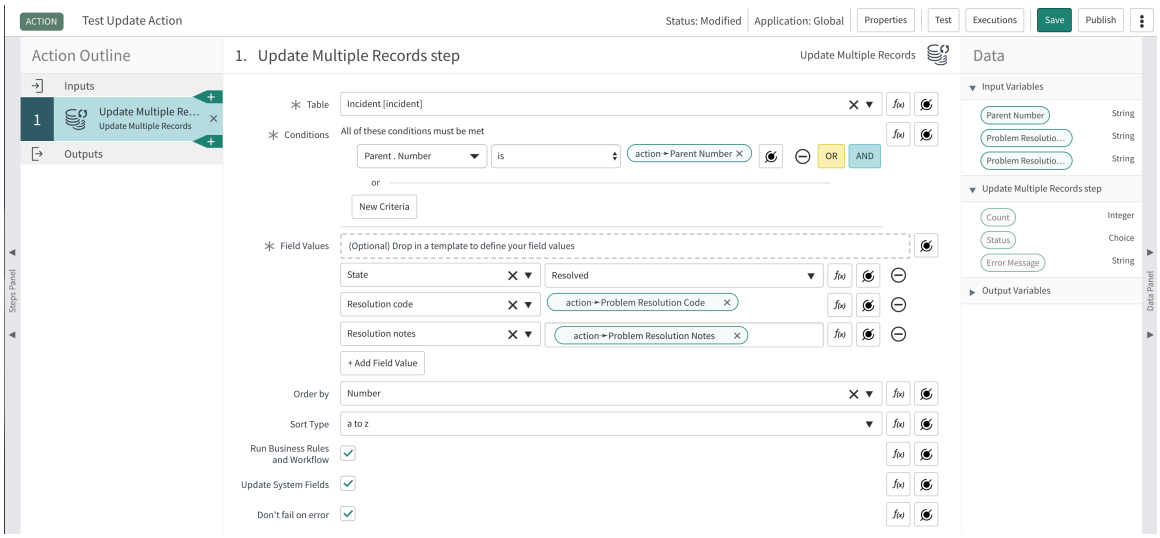
Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Example



Outputs

Field	Description	Data Type
Count	Number of records updated. If no records are updated, the count is 0.	Integer
Error Message	Message that is displayed if the step produces an error.	String
Status	The completion status of the step as a numeric value. • 0 (success) • 1 (error)	Choice

Update Record step

Update an existing record in a table. You can dynamically add and configure fields for the record, or use a template to set field values.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Record	The record to be updated. Drag-and-drop a record data pill or use the data pill picker to select a record. **Warning:** When using script to select a record, always add a condition to check for a matching record. Some GlideRecord methods return a list of records whenever the method query produces no results. Without a condition check, the action may update all records in a table. For example, this script uses an If condition to verify that a record exists. If the record exists, it returns a Sys ID value. If no record exists, it returns a null result. `var configurationItem = new GlideRecord('cmdb_ci'); if (configurationItem.get(fd_data.trigger.cmdb_ci.sys_id)) return configurationItem; else return null;`
Table	The table associated with the record. When you select a record, this field is automatically set to the table associated with the record.

Field	Description
Field Values	Set static or dynamic values of fields in the record. For example, to set the short description to a static value, select **Short description** and set the desired value. To add dynamic values, see Create a template value input. **Important:** The system does not support updating multiple journal fields such as the additional comments or work notes of a task record.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Wait For Condition step

Pause a flow until record values match a specific set of conditions.

Roles and availability

Available as an Workflow Studio action step. Users with the action_designer role can create a custom action with one or more action steps.

Fields

Field	Description
Record	Drag an input record or a record from a previous step. **Note:** If this record is deleted, the flow stops waiting and continues running.
Table	Read-only. Set to the table associated with the record. Confirm that the system supports Wait for Condition for your selected table. For a list of unsupported tables, see the Unsupported tables section.
Conditions	Select the record values necessary to resume running the flow. For example, if the condition is **[State] [is] [Closed]**, the flow pauses until the condition is met. Once met, the flow moves on to the next step or action. Set static or dynamic conditions to filter records. To define a static condition applied each time the action runs, define the conditions with the condition builder. To enable flow designers to dynamically apply conditions, define an input of type Conditions and drag-and-drop the input data pill into the **Conditions** field. **Note:** For conditions that depend on a specific duration, consider using Wait for a duration flow logic instead.

Field	Description
Enable timeout	Option to limit the amount of time that the flow waits for the action to be completed before continuing. **Note:** Use the **Enable timeout** option to prevent this action from continuing to run. If the condition to continue is never met, a timeout value specifies when the system skips the Wait for Condition action and go to the next item in the flow. You must set a Duration value to enable a timeout. You can also select a Schedule if you want to compute the duration end date based on a specific work schedule.
Duration	Amount of time that the flow waits before continuing when the **Enable timeout** option is selected. Enter the time to wait in hours, minutes, and seconds. If you leave this field empty, the flow does not wait.
Schedule	Schedule used to compute the timeout duration when the **Enable timeout** option is selected. For example, waiting for 10 hours as part of an 8-5 weekdays schedule causes the flow to wait for one or more business days. If you leave this field empty, the timeout runs without a schedule.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Unsupported tables

The system does not support Wait for Condition for the following tables.

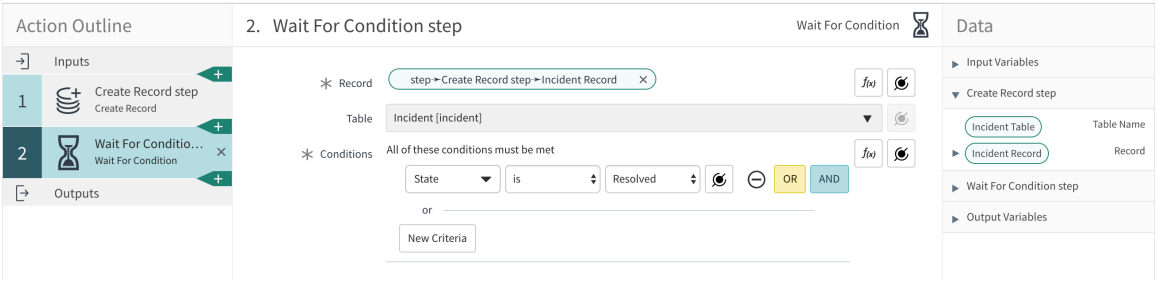
Table Category	Table Names
Audit	Sys Audit [sys_audit], Audit Deleted Record [sys_audit_delete], Audit Relationship Change [sys_audit_relation], Audit Roles [sys_audit_role], Audit Relationship Change [sys_audit_relation], Audit Deleted Record [sys_audit_delete]
Email	Email [sys_email], Email Account [sys_email_account], Email Log [sys_email_log]
Events	Event [sysevent], Notification [sysevent_email_action], Stationery [sysevent_email_style], Email Template [sysevent_email_template], Inbound Email Actions [sysevent_in_email_action], Slow Event [sysevent_pattern], Event Registration

Table Category	Table Names
	[sysevent_registration], Script Action [sysevent_script_action]
Import Sets	Import Set [sys_import_set], Import Set Row [sys_import_set_row], Import Set Row Error [sys_import_set_row_error], Transform History [sys_import_set_run], Computer [imp_computer], Notification [imp_notification], Location [imp_location], User [imp_user]
JRobin	JRobin Database [jrobin_database], JRobin Shard [jrobin_shard], Graph Line [jrobin_graph_line], JRobin Shard Fragments [jrobin_shard_location], Member [jrobin_graph_set_member], Round Robin Archive [jrobin_archive], Round Robin Data Source [jrobin_datasource], Round Robin Definition [jrobin_definition], Round Robin Graph [jrobin_graph], Round Robin Graph Set [jrobin_graph_set]
Logs	Log Entry [syslog], Service Portal Log Entry [sp_log]
MID Server	MID Server Property [ecc_agent_property], Mid Server Log [ecc_agent_log], Queue [ecc_queue], Configuration [ecc_queue_config], ECC Queue Statistics (by ECC Agent) [ecc_queue_stats_by_ecc_agent]
Performance Analytics	Job Log [pa_job_logs]
Record Watcher	Responders [sys_rw_action], Channel Responders [sys_rw_amb_action]
Reporting	Summary Set [sys_report_summary], Report Summary Line [sys_report_summary_line]
Scheduled Jobs	Schedule Item [sys_trigger], Broadcast Message [sys_broadcast_message], Broadcast Message Relationships [sys_broadcast_message_m2m], Progress Worker [sys_progress_worker], Progress Worker Domain [sys_progress_worker_domain]
SSO	SSO Properties [sso_properties], Digest Token Properties [digest_properties], SAML Update 1 Properties [saml2_update1_properties], SSO Federation [sso_federation]
System Cache	Cache Flush [sys_cache_flush], Cache Entry [sys_db_cache]
System Clone	ServiceNow Instance [instance], Clone Security Token [clone_token], Preserved Data [clone_preserved_data]
System Dictionary	Dictionary Entry Override [sys_dictionary_override]
System Events	Event Processor [sys_event_processor]

Table Category	Table Names
System Fields	Field Class [sys_glide_object]
System Performance	Component Status [sys_status], Cluster Message [sys_cluster_message], Node State [sys_cluster_state]
Text Index	Ts Attachment [ts_attachment], Text Index Attribute Map [ts_attribute_map], Ts Chain [ts_chain], Chain Summary [ts_chain_summary], Text Index Column Attribute Map [ts_column_attribute_map], Text Index Configuration [ts_configuration], Text Index Configuration Attribute [ts_configuration_attribute], Ts Delete Doc [ts_deleted_doc], Ts Document [ts_document], Ts Field [ts_field], Text Search Groups [ts_group], Japanese User Token [ts_japanese_token_dictionary], Ts Phrase [ts_phrase], Global Searches [ts_query], Knowledge Searches [ts_query_kb], Text Search Stat [ts_search_stats], Text Search Summaries [ts_search_summary], Stop Word [ts_stop], Synonym Dictionary [ts_synonym_dictionary], Synonym Set [ts_synonym_set], Text Search Table [ts_table], Text Index Table Attribute Map [ts_table_attribute_map], Service Catalog Searches [sc_ts_query], Ts Word [ts_word], Ts Word Roots [ts_word_roots]
Update Sets	Update Set [sys_update_set], Update Version [sys_update_version], Customer Update [sys_update_xml], Update Set Log [sys_update_set_log]
Upgrades	System Upgrades [sys_upgrade_history], Upgrade Details [sys_upgrade_history_log], System Upgrade Metric [sys_upgrade_metric], Upgrade Blame Log [sys_upgrade_blame], Upgrade Manifest [sys_upgrade_manifest], Upgrade State [sys_upgrade_state]
Usage Analytics	Usage Data for Applications [ua_app_usage], UsageAnalytics Count Configurations [usageanalytics_count_cfg], Application Metadata [ua_app_metadata], UsageAnalytics Count for Tables [usageanalytics_count], Subscription [license_details], Role for Subscription [role_has_license]
Users	User Session [sys_user_session], User Token [sys_user_token], User Preference [sys_user_preference], Navigator History [sys_ui_navigator_history]
Workflow	Workflow Execution [wf_workflow_execution], Workflow History [wf_history], Workflow Executing Activity [wf_executing], Workflow

Table Category	Table Names
	Queued Command [wf_command], Workflow Context [wf_context], Workflow Transition History [wf_transition_history]

Example



Output

Field	Description	Data Type
State	The completion status of the action as a numeric value. • 0 (success) • 1 (error)	Choice

ZIP step

Manage the attachments in a record by performing archive operations such as zip and unzip. You can also view the details of a zipped file.

Roles and availability

Users with the action_designer role can create a custom action with one or more action steps.

Important:
ZIP step is not available in the base system and requires the ServiceNow® Integration Hub subscription. After the required plugin is activated, the step is visible under Integrations.

Benefits

Use zip operations to achieve the following benefits:

- Zip the attachments in a record to reduce the file size.
- Unzip specific attachments from a zipped archive in a record.
- View the details of the attachments before unzipping a zipped archive.

Zip operations

Perform the following operations by using the zip step:

- For information about the zip operation, see Zip.
- For information about the Unzip operation, see Unzip.
- For more information about viewing the contents of a compressed zip archive, see Get Zip File Details.

Zip operation

Compress one or more record attachments into a single zip archive. You can also manage the contents of a zip archive by removing specific attachments from the zipped file.

Zip step

Field	Description
Operation	**Zip** operation that archives the attachments in the required record.
Source Attachment Records	Records that contain the attachments that you want to archive.
Target Zip File Name	Name of the zipped archive.
Target record	Record that you want to archive the attachments to.
Target Table	Table related to the target record. The table is automatically populated when you select the record from the **Target record** field.
Advanced Options	
Delete Source Files	Field used to remove the original attachments from the source record after the attachments are archived.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Unzip operation

Extract one or more compressed attachments from a zip archive. You can extract all the compressed attachments in an archive or choose specific attachments to unzip.

Zip step

Field	Description
Operation	**Unzip** operation that extracts the attachments from the zipped archive in the record.
Source Attachment Records	Record that contains that zipped archive that you want to unzip.
Target record	Record that you want to unzip the attachments in the zipped archive to.

Zip step (continued)

Field	Description
Target Table	Table related to the target record. The table is automatically populated when you select the record from the **Target record** field.
Advanced Options	
Files to Unzip	Option to select the files that you want to unzip. Use the following options: • **All Files**: Unzips all the attachments from the zipped archive. • **Selected Files**: Unzips only specific attachments that you select by using the **File Name Regular Expression** field. • **File Name Regular Expression**: Regular expression for selecting the required attachments from the zipped file in the source record. For example, the regular expression `*\.png` selects all the `.png` files in the archive.
Delete Source Zip Attachment	Field that removes the zipped archive in the source record after the unzip operation is completed.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

Get Zip file details

View the contents of a compressed zip archive.

Zip step

Field	Description
Operation	**Get Zip File Details** operation displays the details of the zip archive in the required record.
Zip Attachment	Zipped archive that you want to view the details for.

Action error evaluation

If this step fails

Data type: *Choice*

Option to continue running the next step or go to error evaluation. To use the step status code or message for a custom action error condition, see Action error evaluation.

User preferences for flows

Enable or disable user preferences for flow to change options available to Workflow Studio.

Flow preference	Description
Show triggered flows	Option to show flows with a trigger from the subflow picker. By default, this option is disabled, and the subflow picker hides triggered flows from the subflow picker.
Show store spokes	Option to show Workflow Studio content from the ServiceNow Store from the action picker. By default, this option is enabled, which displays Workflow Studio content from ServiceNow Store spokes you have already installed.
Show inline script toggle.	Option to add or edit inline scripts when configuring an action input. By default this options is enabled, and you can add or edit inline scripts.
Show advanced connection options	Option to show advanced connection properties for actions that use connection aliases or inline connections. By default, this option is disabled.
Show flows as diagrams	Option to show all flows in the flow diagramming view. By default this option is disabled, and all flows open in the descriptive text view.
Auto Refresh Tests	Option to create an Asynchronous Message Bus (AMB) channel for test runs of flows and actions. The AMB channel listens for state changes of execution details that are in a waiting state. When the execution details are done, the AMB channel closes and sets the state to complete. By default, this option is disabled, and you must manually update execution details.
Show recommendations	Option to show flow recommendations to build a flow. The system generates recommendations based on the current position in the flow and the flow component names listed before. By default, this option is enabled. For more information about flow recommendations, see Flow recommendations.

Related topics

Set flow user preferences

Supported Service Catalog variable types

Workflow Studio supports several Service Catalog variable types for both single-row and multi-row variable sets.

Supported variable types

Service Catalog variable type	Workflow Studio variable type for single-row variable sets	Workflow Studio variable type for multi-row variable sets
Check box	True/false	True/false
Date	Date	Date/time
Date and time	Date/time	Date/time
Duration	Duration	String
Email	Email	String
HTML	HTML	Not supported

Supported variable types (continued)

Service Catalog variable type	Workflow Studio variable type for single-row variable sets	Workflow Studio variable type for multi-row variable sets
IP Address	IP Address	String
Label	String	Not supported
List collector	List	Not supported
Lookup multiple choice	Choice	String
Lookup select box	Choice	String
Macro	String	Not supported
Macro with label	String	Not supported
Masked	Masked code	String
Multi-line text	Multiple line small text area	String
Multiple choice	Choice	Choice
Numeric scale	Integer	Integer
Reference	Reference	String
Select box	Choice	Choice
Single-line text	String	String
URL	URL	String
Wide single-line text	String	String
Yes/No	True/false	True/false

More information

For a list of Service Catalog variable types, see Types of service catalog variables.

Transform functions

Transform data pill values without the need to write a script. Use transform functions to reformat text, perform mathematical calculations, sanitize potentially unsafe SQL statements, and serialize complex objects to raw XML.

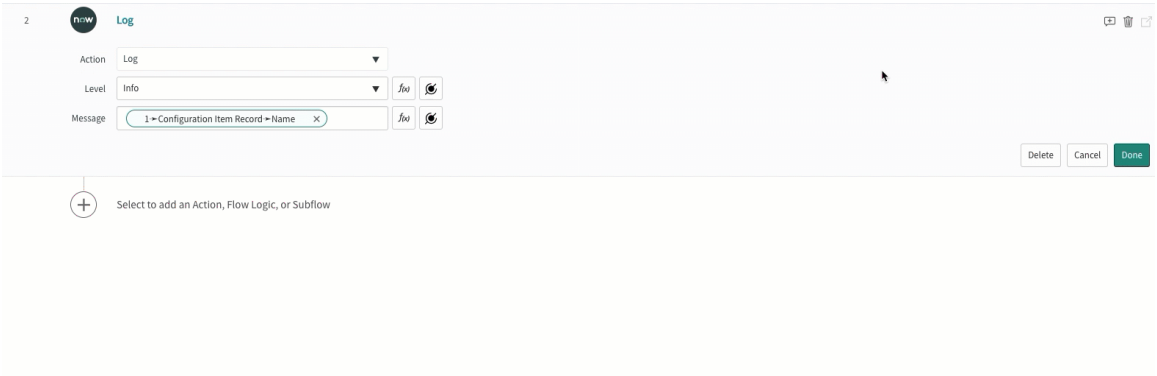
Available transform function categories include date and time, string, utilities, simple math, sanitize shell arguments, sanitize SQL, and complex data. Some examples of transform function uses include:

- Trimming white space from a string before integrating it into the CMDB.
- Adding days, hours, minutes, and seconds to a date or time to localize for a specific time zone.
- Sanitizing SQL values to prevent injection as part of a JDBC step for an Integration Hub spoke.
- Retrieving an appropriate value from a map of priorities that have equivalent values in a third-party database.
- Transforming a complex object into raw XML as part of a REST step **Request Body** field.

Note:
Custom transform functions are not currently supported. For information on creating custom functions to transform Workflow Studio data, see Inline scripts.

Applying a transform function

You can apply a transform function to a data pill when you are designing or creating a flow. To apply a transform function, point or tab to a data pill and select the **f(x)** ( ) icon that appears. Selecting the icon displays the Available Transforms list. Select the transform function that you want to apply to your data pill, enter information into any required fields, and select **Apply**. Your selected transform function appears in the Applied Transforms list.



Applying multiple transform functions

You can apply multiple transform functions to the same data pill. The system applies transform functions sequentially from top to bottom as listed in the Applied Transforms list. For example, you can apply a String to Date transform function followed by the Add Time transform function.

Viewing applied transform functions

You can find out which transform functions are applied to a data pill by looking at the Applied Transforms list. When you are designing or creating a flow or action, point to or tab into the data pill and select the **f(x)** ( ) icon. You can also view applied transform functions in the flow execution details.

Note:
The output values for applied transform functions are field values, not Display values.

When a data pill is dropped into certain types of inputs, the system automatically suggests a transform function category that is related to the input. For example, the system suggests transform functions to escape special characters in SQL statements, prevent script injection in shell scripts, and validate API or operation requests. Currently, the system suggests transform function categories for these inputs:

- For data pills dropped in the JDBC step's **SQL Statement** input, the sanitize SQL transform function category is displayed automatically.
- For data pills dropped in the SSH step's **Command** input, the sanitize shell arguments transform function category is displayed automatically.

General guidelines

Apply transform functions to valid types of data pills for the input

Be sure to check the type of data pill for the input before applying a transform function. Applying a transform function to an invalid data pill type results in the

system skipping the transform. An error also occurs if transform functions produce results that the system cannot parse. For example, when transforming a string into a date, the system throws an error if the transform does not produce a valid date.

Confirm applied transform functions for multiple inputs with the same data pill

A transform function creates a new value at runtime for a specific input, and does not change the original data pill. If you use the same data pill across multiple actions or steps, transform functions must therefore be applied to each individual input.

View final transformed values in the flow execution details

Only the final transformed value appears in the flow execution details, and not the value for each applied transform.

Test transform functions to verify they produce expected results

Make sure that your transform functions produce the expected runtime values for the data pills. For more information, see Test a flow and Test an action.

Date and time transform functions

Use date and time transform functions to recalculate or reformat data pills for Date/Time values.

Date and time transform functions require a Date/Time or String input data pill. Some functions also support Duration data pills. Make sure to use the correct input data pill type when applying date and time transform functions. If a date and time transform function is applied to an improper data pill type, the data is not transformed at runtime and the input value is returned instead. For more information on confirming your flow runtime values, see Test a flow.

Note:
Runtime Date/Time values are not localized and appear in the UTC (Coordinated Universal Time) time zone. For more information, see Time zones and time zone representation.

Add Time

Adds days, hours, minutes, or seconds to an input Date/Time, Date, Due Date, or Duration.

Input data pill	Parameters	Output data pill
Date/Time, Date, Due Date, or Duration **Note:** If the input is a Date or Due Date, the transform creates a full Date/Time value using time values of hour 0, minute 0, and second 0.	*Duration* - Amount of time to add in days, hours, minutes, and seconds	Date/Time - Transformed Date/Time value after adding the specified *Duration*

Example

- Input: `2019-09-12 11:00:00`
- Duration: `3 hours`
- Output: `2019-09-12 14:00:00`

Subtract Time

Subtracts days, hours, minutes, and/or seconds from the input Date/Time, Date, Due Date, or Duration.

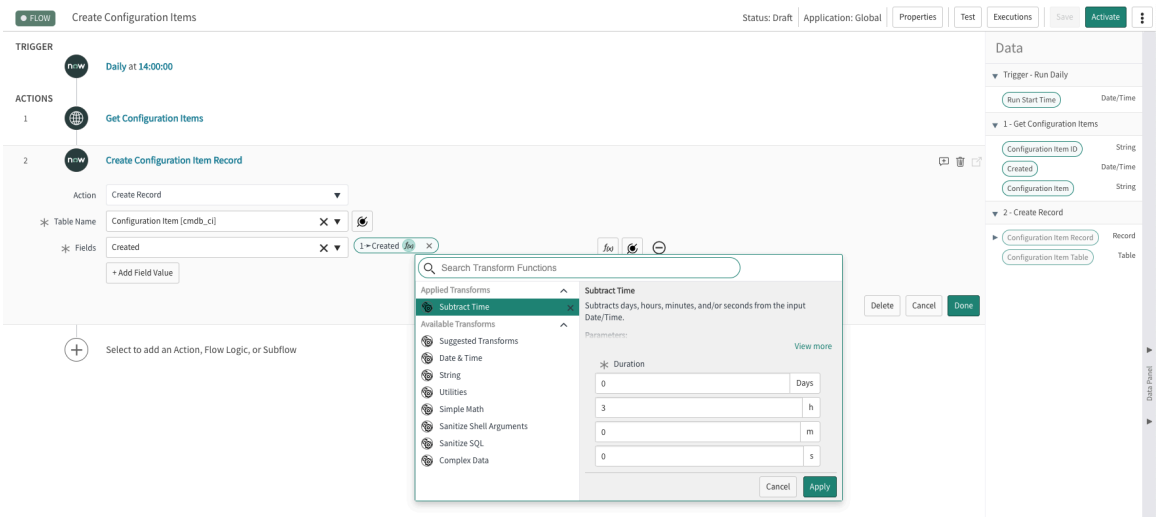
Input data pill	Parameters	Output data pill
Date/Time, Date, Due Date, or Duration **Note:** If the input is a Date or Due Date, the transform creates a full Date/Time value using time values of hour 0, minute 0, and second 0.	*Duration* - Enter an amount of time to subtract in days, hours, minutes, and seconds	Date/Time - Transformed Date/Time value after subtracting the specified *Duration*

Example

- Input: `2019-09-12 11:00:00`
- Duration: `2 days, 1 hours, 5 minutes, 10 seconds`
- Output: `2019-09-10 09:54:50`

In this example, the flow retrieves a CMDB Configuration Item record from a remote instance. The Subtract Time transform function then localizes the value of the **Created** field by subtracting three hours from the input Date/Time.

Localize a field value's time zone



String to Date

Converts the input String to a Date/Time.

Input data pill	Parameters	Output data pill
String formatted as specified in the *Input Date Format*	• *Input Date Format* - Date/Time or date format of the input String • *Custom Format* - Valid Date/Time or date format represented as a String. Required only if Custom Format is selected as the *Input Date Format*.	Date/Time

Note:

- If the Date/Time value for the **Custom Format** input is incomplete, the transform creates a full Date/Time value using default dates and times. In such a case, the transform defaults to the current year, the current month, day 1 of a month, hour 0, minute 0, and second 0. For example, an input data pill value of **Oct 2019** and a custom date format of **MMM yyyy** produces an output of `2019-10-01 00:00:00`.
- If you use an incorrect data pill type or invalid **Custom Format**, the flow cancels during runtime.

Example

- Input: `'1995-11-20'`
- Input Date Format: `ISO Date (2004-06-28)`
- Output: `1995-11-20 00:00:00`

Date to String

Converts the input Date/Time, Date, or Due Date to a String. Select a **Date Format** for the input Date/Time. Alternatively, enter a **Custom Format** for the input Date/Time.

Input data pill	Parameters	Output data pill
Date/Time, Date, or Due Date	• *`Output Date Format`* - Date/Time or date format of the output String • *`Custom Format`* - Valid Date/Time or date format represented as a String. Required only if selecting Custom Format as the **Date Format**.	String formatted as specified in the *`Output Date Format`*

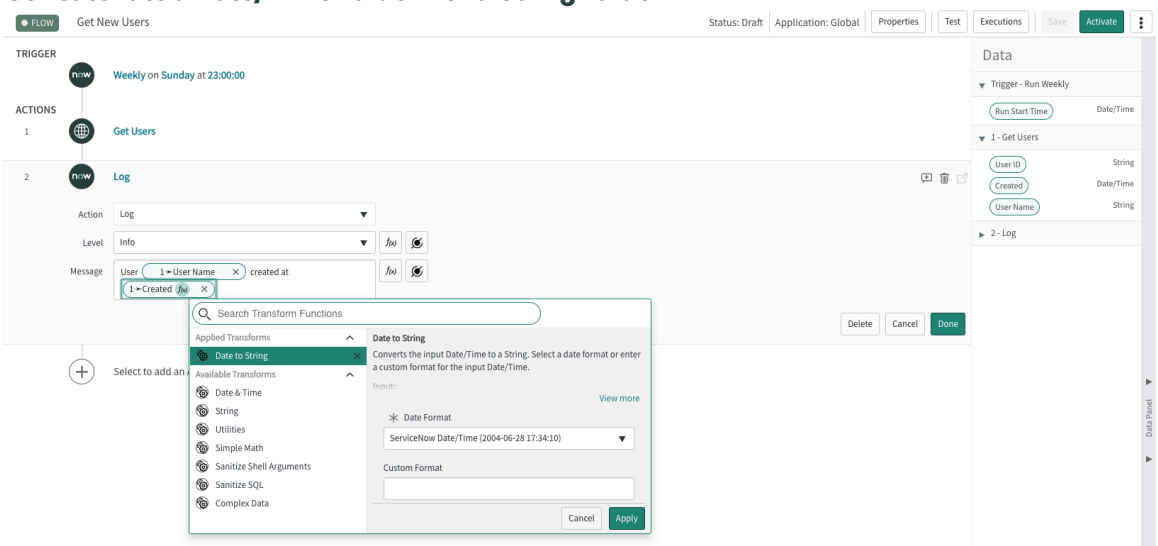
Note:
If you use an incorrect input data pill type or invalid **Custom Format**, the flow cancels during runtime.

Example

- Input: `1969-12-31 14:23:57`
- Output Date Format: `Custom (enter below)`
- Custom Format: `'On' MMM dd, yyyy 'at' hh:mm a`
- Output: `'On Dec 31, 1969 at 2:23 PM'`

In this example, the flow retrieves a User [sys_user] record from a third-party database. The Date to String transform function converts the format of the **Created** field and then logs the date, time, and name that is associated with the record.

Concatenate a Date/Time value with a String value



Custom date formats

You can specify a custom 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
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

String	Description	Output Format	Example
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

Day

Retrieves the day component from the specified Date/Time.

Input data pill	Output data pill
Date/Time	Integer - Day from the specified date.

Example

- Input: `2021-11-20 13:06:12`
- Input Date Format: `ISO Date (2004-06-28)`
- Output: `20`

Hour

Retrieves the hour component from the specified Date/Time.

Input data pill	Output data pill
Date/Time	Integer - Hour from the specified date.

Example

- Input: `2021-11-20 13:06:12`
- Input Date Format: `ISO Date (2004-06-28)`
- Output:`13`

Minute

Retrieves the minute component from the specified Date/Time.

Input data pill	Output data pill
Date/Time	Integer - Minute component from the specified date.

Example

- Input: `2021-11-20 13:06:12`
- Input Date Format: `ISO Date (2004-06-28)`
- Output: `06`

Second

Retrieves the second component from the specified Date/Time.

Input data pill	Output data pill
Date/Time	Integer - Second component from the specified date.

Example

- Input: `2021-11-20 13:06:12`
- Input Date Format: `ISO Date (2004-06-28)`
- Output: `12`

Month

Retrieves the month component from the specified Date/Time.

Input data pill	Output data pill
Date/Time	Integer - Month component from the specified date.

Example

- Input: `2021-11-20 13:06:12`
- Input Date Format: `ISO Date (2004-06-28)`
- Output: `11`

Week

Evaluates the week number for the specified Date/Time.

Input data pill	Output data pill
Date/Time	Integer - Week number for the specified date.

Example

- Input: `2021-04-07 12:01:12`
- Input Date Format: `ISO Date (2004-06-28)`
- Output: `15`

Year

Retrieves the year component from the specified Date/Time.

Input data pill	Output data pill
Date/Time	Integer - Year component from the specified date.

Example

- Input: `2021-04-07 12:01:12`
- Input Date Format: `ISO Date (2004-06-28)`
- Output: `2021`

Date Difference

Evaluates the time duration difference between the specified input date and the parameter date and then adds the difference time duration to the Epoch time (1970-01-01 00:00:00).

Input data pill	Parameters	Output data pill
Date/Time, Date, or Due Date	Date/Time, Date, or Due Date	Duration - Time duration difference added to the Epoch time (1970-01-01 00:00:00).

Example

- Input: `2021-05-02 09:10:12`
- Input Date Format: `ISO Date (2004-06-28)`
- Parameters: `2021-04-07 06:02:23`
- Output: `1970-01-26 03:07:49`

End of Month

Evaluates the last day of the month after adding the specified number of the months to the specified date.

Input data pill	Parameters	Output data pill
Date	Number of Months - Months to add to the specified date.	Integer - Last day of the month after adding the number of months to the specified input date.

Example

- Input: `2021-11-20`
- Input Date Format: `ISO Date (2004-06-28)`
- Number of Months: `3`
- Output: `2022-02-28`

String transform functions

Use string transform functions to reformat or perform calculations on String data pills.

String transform functions require a String input data pill. Make sure to use the correct input data pill type when applying string transform functions. If a string transform function is applied to an improper data type, the data is not transformed at runtime and the input value is returned instead. For more information on confirming your flow runtime values, see Test a flow.

Convert String to Number

Converts a string into a number.

Input data pill	Output data pill
String	Number - Number converted from a string.

Example

- Input: `"500"`
- Output: `500`

Contains

Returns **true** when the input string contains a given sequence of characters.

Input data pill	Parameters	Output data pill
String	Characters to search for.	Boolean indicating whether a sequence of characters exists in the input string

Example

- Input: `Cheese Pizza`
- Parameter: `Cheese`
- Output: `true`

Does not Contain

Returns **true** when the input string does not contain a given sequence of characters.

Input data pill	Parameters	Output data pill
String	Characters to search for.	Boolean indicating whether a sequence of characters does not exist in the input string

Example

- Input: `Cheese Pizza`
- Parameter: `Joey`
- Output: `true`

Ends With

Returns **true** when the input string ends with a given sequence of characters.

Input data pill	Parameters	Output data pill
String	Characters to search for.	Boolean indicating whether the input string ends with the given sequence of characters

Example

- Input: `Cheese Pizza`
- Parameter: `Pizza`
- Output: `true`

First Character

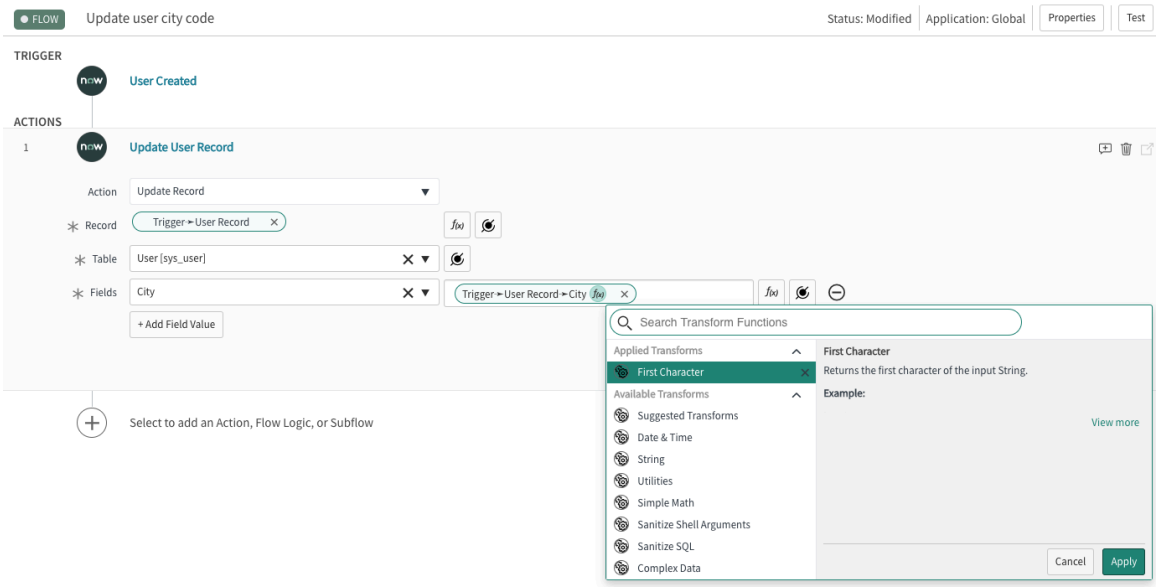
Returns the first character of the input String.

Input data pill	Output data pill
String	String - Transformed String as the first character of the input String

Example

- Input: `Madrid`
- Output: `M`

In this example, the flow triggers when a User [sys_user] record is created. The flow then updates the **City** field for the User [sys_user] record with a code that is represented as the first character of the city's name.



Last Character

Returns the last character of the input String.

Input data pill	Output data pill
String	String - Transformed String as the last character of the input String

Example

- Input: `Madrid`
- Output: `d`

Replace String

Returns a replaced string from the input string based on the provided regular expression (regex) and replacement string. Use the JavaScript regular expression format.

Input data pill	Parameters	Output data pill
String	• *`Regex`* - Regular expression to be matched for replacement • *`Replace String`*- Replacement string	Resulting string after replacement with given parameters

Example

- Input: `"Example input string."`
- Parameters:
  - Regex: `\"`
  - Replacement string: `\\\"`
- Output: `\"Example input string.\"`

Size

Returns the total number of characters in the input String.

Input data pill	Output data pill
String	Integer

Example

- Input: `Example input string.`
- Output: `21`

Split

Returns an Array.String based on a provided **Separator** that splits the input String. If the **Separator** field is left blank, the transformation is ignored and the system returns the input String. If entering any data type other than a String as the **Separator**, the system converts the provided value to a String.

Input data pill	Parameters	Output data pill
String	*`Separator`* - Enter a delimiter that specifies where the input String should be split. If left blank, the input String is not transformed at runtime.	Array.String - An array of substrings from the input String

Example

- Input: `Example, input, string.`
- Separator: `,`
- Output: `["Example", "input", "string."]`

Starts With

Returns **true** when the input string starts with a given sequence of characters.

Input data pill	Parameters	Output data pill
String	Characters to search for.	Boolean indicating whether the input string starts with the given sequence of characters

Example

- Input: `Cheese Pizza`
- Parameter: `Chees`
- Output: `true`

Substring

Returns a substring from the input String that is based on the provided **Start Index** and **End Index**. Input String index starts at 0.

Input data pill	Parameters	Output data pill
String	• *`Start Index`* - Index of the first character to include in the returned substring • *`End Index`* - Index of the last character to include in the returned substring	String - Transformed String as a substring of the input String

Note:
If the *`Start Index`* value is longer than the string's length, then the transform function returns the input string. This behavior is different from the JavaScript *`substring()`* method, which instead returns an empty string.

Example 1

- Input: `Example input string`
- Start Index: `3`
- End Index: `6`
- Output: `mple`

Example 2

- Input: `Example input string`
- Start Index: `30`
- End Index: `40`
- Output: `Example input string`

To Lower Case

Converts the input String to all lowercase characters.

Input data pill	Output data pill
String	String in all lowercase characters

Example

- Input: `ExamPle inpuT stRing`
- Output: `example input string`

To Proper Case

Changes the case of words in the input string. Capitalizes the first letter of each word and makes the remaining letters in the word lower case. A word is considered any string separated by a space, hyphen, backslash, or forward slash character. The transform function always evaluates words from left-to-right to determine the first letter.

Input data pill	Output data pill
String	String in proper case

Example

- Input: `exAMPle-input string/TEXT`
- Output: `Example-Input String/Text`

To Upper Case

Converts the input String to all uppercase characters.

Input data pill	Output data pill
String	String in all uppercase characters

Example

- Input: `ExamPle inpuT stRing`
- Output: `EXAMPLE INPUT STRING`

Trim

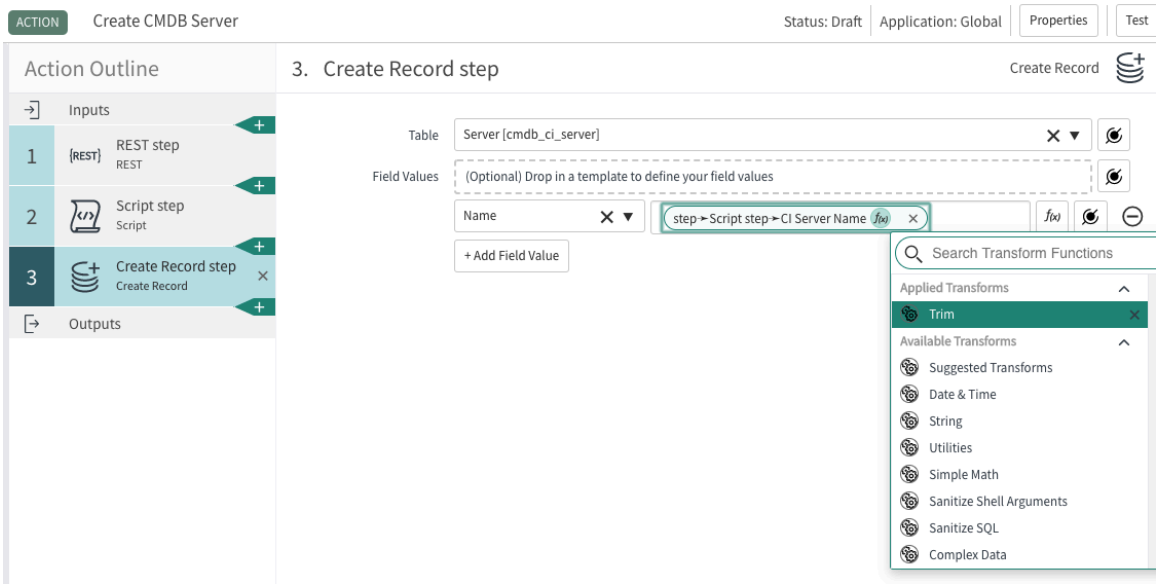
Removes white space from the beginning and end of the input String. Does not remove white space within the input String.

Input data pill	Output data pill
String	String - Transformed String with trimmed white space

Example

- Input: `SQL Server APAC 1`
- Output: `SQL Server APAC 1`

In this example, the action makes a REST call to a third-party system and GETs a response body containing data about a server. Then, the Trim transform function removes any unwanted white space before adding the server's name to a new record in the Server [cmdb_ci_server] table.



Utilities transform functions

Use utilities transform functions to return a Complex Object from an Array, or a value associated with a specific key.

Utilities transform functions require an Array, Name-Value Pair, String, Integer, or Choice input data pill. Make sure to use the correct input data pill type when applying utilities transform functions. If a utility transform function is applied to an improper data type, the data is not transformed at runtime and the input value is returned instead. For more information on confirming your flow runtime values, see Test a flow.

Get First Item from Array

Returns the first item from the input array as a complex object.

Input data pill	Output data pill
Array	*`Complex Object`* - First item found in the input array

Get Item from Array

Returns a Complex Object from the input Array. Enter a value for the *`Nth Item`* in the input Array that you want to return. The *`Nth Item`* represents the Array index, starting at 0.

Input data pill	Parameters	Output data pill
Array	*Nth Item* - Enter the index of the target object in the input Array. The Array index starts at 0.	Complex Object

Get Item from Name/Values

Returns a value that is associated with a matching key from a map of Name-Value Pairs.

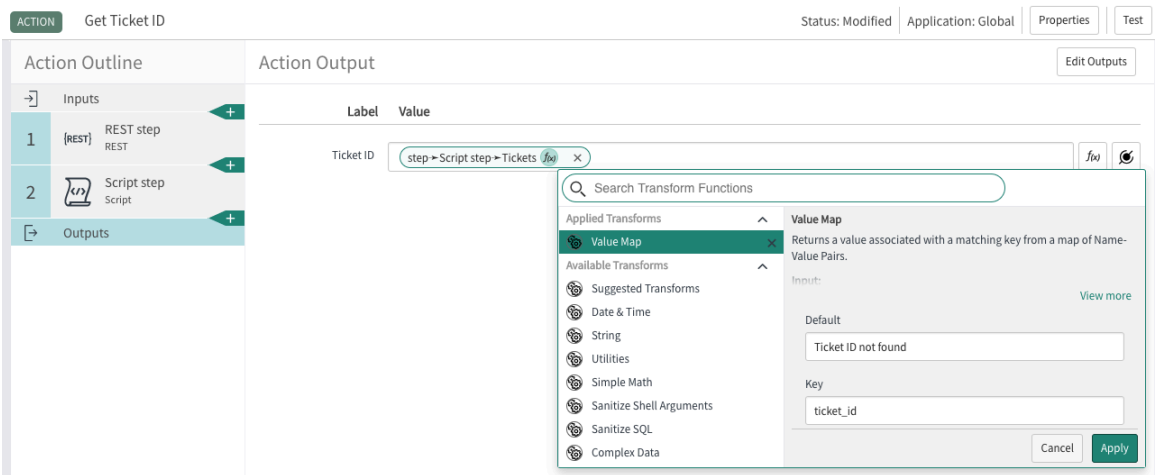
Input data pill	Parameters	Output data pill
Name-Value Pairs	• *Key* - Name of the key that is used to look up its corresponding value • *Default* - The value returned when there is no matching key	String associated with the matching key

Note:
When applying the Get Item from Name/Values transform function, consider that the runtime value might be the system value, not the display value. For example, if mapping the **Priority** field in the Incident table to a similar field in a remote table, the returned runtime value might be 2, not 2 - High.

Example usage:

- Input: `"username":"abel.tuter"`
- Key: `username`
- Default: `example.username`
- Output: `abel.tuter`

In this example, an action makes a REST call to a third-party system and GETs ticket data as a map of Name-Value Pairs. A Ticket ID is provided as an output for this action. The Get Item from Name/Values transform function returns either the value that is associated with the `ticket_id` key or `Ticket ID not found`.



Get Last Item from Array

Returns the last item from the input array as a complex object.

Input data pill	Output data pill
Array	*`Complex Object`* - Last item found in the input array

Is Blank

Returns **true** when the input is blank. A string input is blank when it is an empty string. An integer input is blank when it is zero. A Boolean input is blank when it is **false**.

Note:
This transform function doesn't support reference inputs.

Input data pill	Output
Any	Returns **true** or **false**

Example usage:

- Input: an integer data pill with `0`
- Output: `true`

Is False

Returns **true** when the input is false. A string is false when it is an empty string. An integer is false when it is zero. A Boolean input is false when it is **false**.

Input data pill	Output data pill
Any	Returns **true** or **false**

Example usage:

- Input: an integer data pill with `13`
- Output: `false`

Is Not Blank

Returns **true** when the input is not blank. A string is not blank when it is not an empty string. An integer is not blank when it is anything but zero. A Boolean input is not blank when it is **true**.

Note:
This transform function doesn't support reference inputs.

Input data pill	Output data pill
Any	Returns **true** or **false**

Example usage:

- Input: an integer data pill with `13`
- Output: `true`

Is Null

Returns **true** when the input value is null. An input is null if it is not initialized, or if it is a null object or reference.

Input data pill	Output
Any	Returns **true** or **false**

Example usage:

- Input: an integer data pill with `725`
- Output: `false`

Is True

Returns **true** when the input is true. A string is true when it is not an empty string. An integer is true when it is anything but zero. A Boolean input is true when it is **true**.

Input data pill	Output data pill
Any	Returns **true** or **false**

Example usage:

- Input: an integer data pill with `13`
- Output: `true`

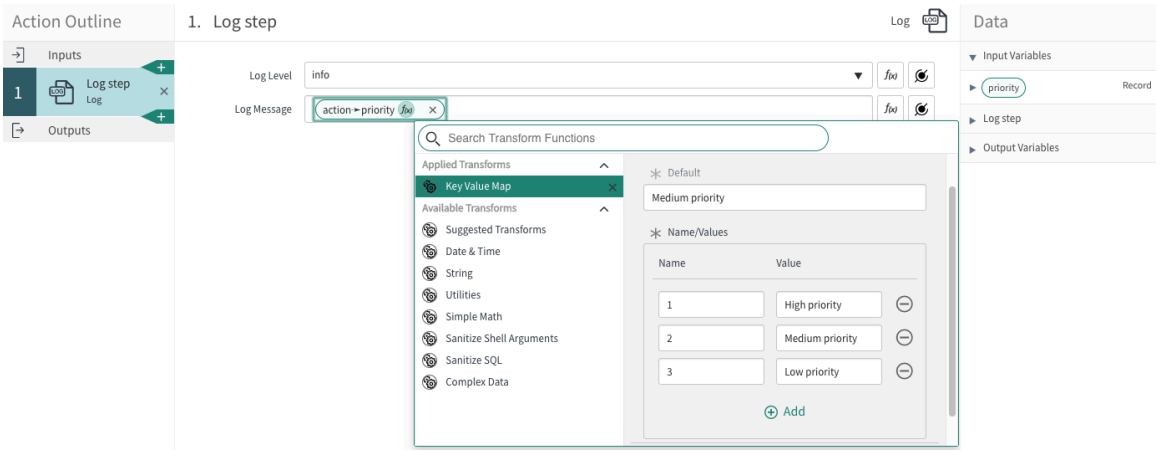
Key Value Map

Returns a value associated with a matching key, or a default value if there is not a match.

Input data pill	Parameters	Output data pill
String or Integer	• *`Name`* - The key that is used to find the corresponding value • *`Value`* - The value matched to a specific key • *`Default`* - The value returned when there is no matching name	String associated with the matching key

Example usage:

In this example, a log action stores a record's priority as a message. In the Name-Values section, each priority is mapped to a corresponding string value. When the flow runs, the record's priority number is transformed to a string, and the string is logged to a message.



Sort

Sorts the specified array in ascending or descending order.

Input data pill	Parameters	Output data pill
Array.String, Array.Integer, Array.Boolean, or Array.Datetime. - Unsorted array	*`Sort Order`* - Ascending or descending	Array - Sorted array

Note:
Sort function is case-sensitive for array.strings.

Example usage:

- Input: `[7, 2, 3, 1, 7, 9]`
- Parameter: Ascending
- Output: `[1, 2, 3, 7, 7, 9]`

Unique

Removes the duplicate elements from the specified array.

Input data pill	Output data pill
Array.String, Array.Integer, Array.Boolean, or Array.Datetime.	Array - After duplicated elements are removed

Example usage:

- Input: `[7, 2, 3, 2, 7, 9]`
- Output: `[7, 2, 3, 9]`

Join

Concatenates the individual elements of the specified array with the specified delimiter and returns the concatenated string.

Input data pill	Parameters	Output data pill
Array.String, Array.Integer, Array.Boolean, or Array.Datetime.	Delimiter - Character that separates the individual elements after concatenation.	String - String after a delimiter is added.

Example usage:

- Input: `[1, 2, 3]`
- Parameters: `<`
- Output: `1<2<3`

Simple math transform functions

Use simple math transform functions to perform basic mathematical calculations on Number data pills.

Simple math transform functions require an Array.Number, Array.Integer, or Array.Decimal input data pill. Make sure to use the correct input data pill type when applying simple math transform functions. If a simple math transform function is applied to an improper data type, the data is not transformed at runtime and the input value is returned instead. For more information on confirming your flow runtime values, see Test a flow.

Absolute Value

A mathematical function that returns the distance from zero for any real number. An absolute value is always a positive or zero value.

Input data pill	Output data pill
Number, Integer, or Decimal	Number as the absolute value of the input number

Example

- Input: `-3`
- Output: `3`

Add

Adds the given value to the input.

Input data pill	Parameters	Output data pill
Number, Integer, or Decimal	Number to be added.	Number as the addition of the input value by the parameter.

Example

- Input: `12`
- Parameter: `4`
- Output: `16`

Average

Returns the average value of the elements in the input array.

Input data pill	Output data pill
Array.Number, Array.Integer, or Array.Decimal	Number as the average value of the input array.

Example

- Input: `[10, 30, 20]`
- Output: `20`

Count

Returns the number of elements in the input array.

Input data pill	Output data pill
Array.Number, Array.Integer, Array.Decimal, Array.Object, Array.String, or Array.Boolean	Number of elements in the input array

Example

- Input: `[2, 10, 30]`
- Output: `3`

Divide

Divides the input value by a given value.

Input data pill	Parameters	Output data pill
Number, Integer, or Decimal	Number to divide the input value by.	Number as the division of the input value by the parameter.

Example

- Input: `12`
- Parameter: `4`
- Output: `3`

Max

Returns the highest value found in the input Array.

Input data pill	Output data pill
Array.Number, Array.Integer, or Array.Decimal	Number as the highest value in the input Array

Example

- Input: `[1, -5, 20, 6]`
- Output: `20`

Median

Returns the median value of elements in the input array.

Input data pill	Output data pill
Array.Number, Array.Integer, or Array.Decimal	Number as the median value of the input array.

Example

- Input: `[10, 30, 15]`
- Output: `15`

Min

Returns the lowest value found in the input Array.

Input data pill	Output data pill
Array.Number, Array.Integer, or Array.Decimal	Number as the lowest value in the input Array

Example

- Input: `[1, -5, 20, 6]`
- Output: `-5`

Multiply

Multiplies the input value by a given value.

Input data pill	Parameters	Output data pill
Number, Integer, or Decimal	Number to be multiplied.	Number as multiplication of the input value by the parameter.

Example

- Input: `12`
- Parameter: `4`
- Output: `48`

Power

Returns the value of the input value raised to the power of a given value.

Input data pill	Parameters	Output data pill
Number, Integer, or Decimal	Number as the exponent of the power.	Number as the power of input value to the parameter.

Example

- Input: `2`
- Parameter: `3`
- Output: `8`

Round

A mathematical function that approximates a numeric value based on rounding rules and a digit position. The function rounds up by adding one to the digit to be rounded and then replacing all digits to its right with zeroes.

Input data pill	Parameters	Output data pill
Number, Integer, or Decimal	*`Number of Digits`* - A positive integer specifying the position of the digit to be rounded starting on the left	Number as the rounded value of the input number

Example

- Input: `194`
- Parameter: `2`
- Output: `190`

Note:
The function uses the digit to the right of the parameter digit to round up or down. If the digit to the right has a value from zero through four, the function rounds down. If the digit to the right has a value from five through nine, then the function rounds up. If there is no digit to the right, then the function rounds down.

Square Root

A mathematical function that computes a positive number that when multiplied by itself produces the input value. The input value must be a positive real number.

Input data pill	Output data pill
Number, Integer, or Decimal	Number as the square root of the input number

Example

- Input: `16`
- Output: `4`

Subtract

Subtracts the given value from the input.

Input data pill	Parameters	Output data pill
Number, Integer, or Decimal	Number to subtract from the input value.	Number as the subtraction of the input value by the parameter.

Example

- Input: `12`
- Parameter: `4`
- Output: `8`

Sum

Returns the sum of all values in the input Array.

Input data pill	Output data pill
Array.Number, Array.Integer, or Array.Decimal	Number as a sum of all values in the input Array

Example

- Input: `[1, -5, 20, 6]`
- Output: `22`

To Fixed

Truncates a floating number to the specified number of decimal places.

Input data pill	Parameter	Output data pill
Number - Decimal number before truncation.	Number of Digits - Number that specifies the number of decimal places to truncate.	Number - Decimal number after truncation.

Example

- Input: `1.93456`
- Parameter: `2`
- Output: `1.93`

Sanitize shell arguments transform functions

Use sanitize shell arguments transform functions to remove any potentially unsafe command injections in String data pills to be used for Bash shell scripting.

Sanitize shell arguments transform functions require a String input data pill. Make sure to use the correct input data pill type when applying sanitize shell arguments transform functions. If a sanitize shell arguments transform function is applied to an improper data type, the data is not transformed at runtime and the input value is returned instead. For more information on confirming your flow runtime values, see Test a flow.

Note:
When a data pill is dropped into the **Command** input for an SSH step, the sanitize shell arguments transform function category automatically appears.

Sanitize Bash shell arguments

Returns a String free of any unsafe command injections in your Bash shell script. Wraps the input String with single quotes and escapes any existing single quotes so that you can pass the String directly to a shell function as a safe argument.

Input data pill	Output data pill
String	String - String with Bash shell arguments properly escaped

Sanitize SQL transform functions

Use sanitize SQL transform functions to escape special characters and prevent injection in String data pills to be used for SQL statements.

Sanitize SQL transform functions require a String input data pill. Make sure to use the correct input data pill type when applying sanitize SQL transform functions. If a sanitize SQL transform function is applied to an improper data type, the data is not transformed at runtime and the input value is returned instead. For more information on confirming your flow runtime values, see Test a flow.

Note:
When a data pill is dropped into the **SQL Statement** input for a JDBC step, the sanitize SQL transform function category automatically appears.

Sanitize SQL Identifier

Returns a String with escaped special characters/injected values for SQL identifiers (such as table, view, and column names). Wraps the input String in database-specific quotes.

Input data pill	Parameters	Output data pill
String	*Database* - The database-specific context in which characters are escaped. Choices include MySQL, Oracle, PostgreSQL, and Microsoft SQL Server.	String - String with SQL identifiers that are properly escaped based on the selected database

Example

- Input: `simple_column`
- Database: `MySQL`
- Output: `` `simple_column` ``

Note:

- If your input String contains a period character, **Sanitize SQL Identifier** returns an error. To join SQL identifiers using a period, use two data pills concatenated with a period and apply **Sanitize SQL Identifier** to both pills.
- Don't enclose the input data pill in quotes. The system automatically wraps the input value with the type of quotes or backticks that apply to your database type.

Sanitize SQL Value

Returns a String with escaped special characters or injected values for SQL values. Wraps the input String in database-specific quotes.

Input data pill	Parameters	Output data pill
String	*Database* - The database-specific context in which characters are escaped. Choices include MySQL, Oracle, PostgreSQL, and Microsoft SQL Server.	String - String with SQL values that are properly escaped based on the selected database

Example

- Input: `'1'='1`
- Database: `SQLServer`
- Output: `''1''=''1'`

Note:
Don't enclose the input data pill in quotes. The system automatically wraps the input value with the type of quotes or backticks that apply to your database type.

Complex data transform functions

Use complex data transform functions to serialize Complex Object data pills into an XML format.

Complex data functions require a Complex Object input data pill. Make sure to use the correct input data pill type when applying complex data transform functions. If a complex data transform function is applied to an improper data type, the data is not transformed at runtime and the input value is returned instead. For more information on confirming your flow runtime values, see Test a flow.

To XML

Serializes the input Complex Object to XML.

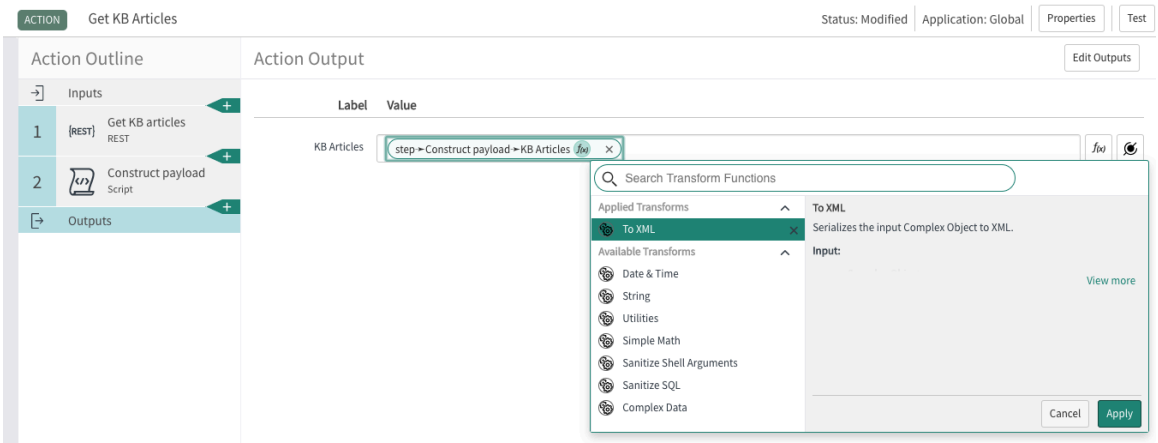
Input data pill	Output data pill
Complex Object	XML - XML document formatted as a String

Example

- Input: `{ "article_id": KB3843202, "article_description": "How do I reset my Active Directory password?" }`
- Output: `<article_id>KB3843202</article_id> <article_description>How do I reset my Active Directory password?</article_description>`

In this example, an action makes a REST call to a third-party knowledge base and retrieves KB article IDs and descriptions. The To XML transform function changes the response body's JSON text into XML format before it is integrated into the system's knowledge base.

Data transformed from JSON text to XML format



Types of flows and when to use them

A decision matrix and basic definitions help you determine what type of flows to create.

Types of flows

Flow

A flow consists of a trigger and one or more actions.

Subflow

A subflow consists of properties, one or more inputs, one or more outputs, a sequence of actions, and the data collected or created.

Contrary to the name, Dynamic Flow is a type of flow logic, not a type of flow.

When to use different flows

When to use different flows

If...	Then create...
You need a constant input to initiate a set of actions	A flow
You need a variable input to initiate a set of actions	A subflow
You want to start a flow by calling it from another flow or script	A subflow

When to use different flows (continued)

If...	Then create...
You want to reuse a set actions in other flows	A subflow
You want to configure different types of inputs for each call	A subflow
You want to specify the inputs available to a subflow when it starts	A subflow
You want to specify the outputs available to a parent flow after a subflow ends	A subflow
You have a large flow with 25 or more actions and want to improve its performance and readability	Subflows
There are interrelated outputs or some action must be taken when all are available	Parallel subflows
There are not interrelated outputs or some action must be taken when all are available	Multiple flows triggered by a single event
You want to correct certain errors in your record data automatically	A subflow
You want to avoid the limit of 10 items in the error-handling-process	A subflow
You want to use subflow outputs to trigger automation in other flows	A subflow

Decision tables reference

Reference topics provide additional information about decision tables properties and administration in Workflow Studio.

Decision Builder system properties

Use system properties to set limits on the maximum number of inputs and decisions for decision tables created in Decision Builder.

These properties are available for decision tables in Decision Builder.

Note:
To open the System Properties [sys_properties] table where these properties are located, enter `sys_properties.list` in the navigation filter.

Properties for decision tables

Property	Description
com.glide.decision_table.excel_hide_references	When set to true, decision tables exported to Excel will not include drop-down lists of records for condition or result columns with type = reference.

Properties for decision tables (continued)

Property	Description
	• Type: true/false • Default value: False
com.glide.decision_table.max_inputs	Maximum number of decision inputs that can be defined on a model (sys_decision) record. • Type: integer • Default value: Empty. There are no limits on inputs when this property is left empty.
com.glide.decision_table.max_questions	Maximum number of decision questions that can be defined on a decision table (sys_decision) record. • Type: integer • Default value: Empty. There are no limits on inputs when this property is left empty.

Note:

- A warning message is displayed when the inputs or decisions exceed the maximum limit set by the administrator.
- By default, decision tables in Decision Builder display the first 20 rows. To view 20 more rows, select **Show more**.

Domain separation and Decision Builder

This is an overview of domain separation as it pertains to Decision Builder. Domain separation allows you to separate data, processes, and administrative tasks into logical groupings called domains. You can then control several aspects of this separation, including which users can see and access data.

Support level: Standard

- Includes **Basic** level support.
- Business logic: Processes can be created or modified per customer by the service provider (SP). The use cases reflect proper use of the application by multiple SP customers in a single instance.
- The owner of the instance must be able to 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 mandatory when a record closes for one tenant but not for another.

Domain separation in Decision Builder

- Decision tables belong to the domain of the user who creates them. For example, when the customer in the TOP domain creates a decision table, it belongs to the TOP domain.
- While users in a parent domain can see decision tables in a child domain, they must edit them in the domain they belong to. For example, an administrator in the TOP domain can see decision tables from the ACME domain but must switch to the ACME domain to edit it.

Classic approvals

Classic approvals are a legacy process to require authorization on tasks before the work is done. In earlier releases, you could create approval records to define approval tasks and associate users or groups to approve or reject them.

Administrators can define classic approval logic by navigating to **All** > **System Policy** > **Rules** > **Approvals**.

Important:
Classic approval rules have been replaced by the Workflow Studio Ask for Approval action. Use Workflow Studio to create workflow-driven approval logic that is easier to maintain and provides better reporting information.

Administrators can see all approval requests by navigating to **All** > **Self-Service** > **My Approvals** and removing the list filter.

An approval record consists of these fields:

Approval record components

Field	Input value
Approver	A reference to the user who is responsible for approving the related record.
State	Choices are: • Not Yet Requested (This state indicates that you are not yet asking your approvers to approve this request. Until you set the status to **Requested** they will receive no email notifications about the request.) • Requested • Approved • Rejected
Approving	A document_id reference field to the record being approved, on any table.
Comments	A journal field for storing comments regarding the approval.
Approval Summarizer	A Create a formatter and add it to a form that displays key fields relevant to the approval from the referenced document. This summarizer will not display if there is no record referenced.

Approval engines

The differences in the way that companies handle their approvals, as well as the differences between approvals for the various applications (such as Service Catalog Requests and Change Management), calls for supporting flexibility in setting up approvals within applications.

Important:
Classic approval rules have been replaced by the Workflow Studio Ask for Approval action. Use Workflow Studio to create workflow-driven approval logic that is easier to maintain and provides better reporting information.

This flexibility is provided through the selection of an "approval engine" that is used to manage the approvals for each of the Task tables (that is, all tables that extend the Task table).

There are three different approval engine options available for each Task table.

Approval engine options

Option	Description
Approval Rules	A simple set of rules that are evaluated until one matches for the Task table. The matching approval rule is used to create the users that are to approve the task. Set up approval rules by navigating to **System Policy** > **Approvals**.
Process Guides	A sequence of approval steps over which you may control how approvals and rejections are handled. This option is deprecated and should not be used.
Turn off Engines	Turn off both approval engines for this Task table. This option should be selected and is made read-only when a workflow is used to manage the approval process for the table. **Note:** Not turning off the approval engines might have a performance or behavioral impact on your instance.

Set up an approval engine

To manage the approvals for each of the Task tables in the system, set up an approval engine.

Before you begin

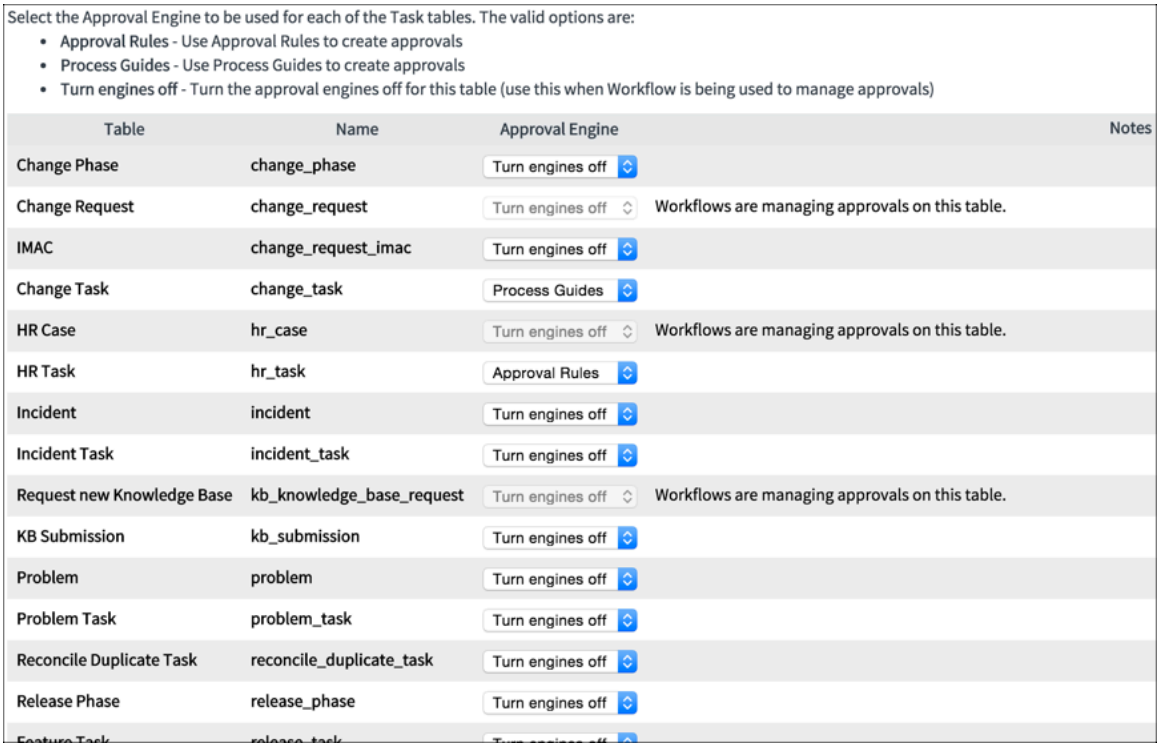
Role required: none

Important:
Classic approval rules have been replaced by the Workflow Studio Ask for Approval action. Use Workflow Studio to create workflow-driven approval logic that is easier to maintain and provides better reporting information.

Procedure

1. Navigate to **All** > **System Properties** > **Approval Engines**.

   The following page appears with the **Approval Engine** option for each Task table in the system. If the **Approval Engine** option is greyed out and shows **Turn engines off**, read the **Notes** in the same row. The most common reason an approval engine is turned off is that a workflow is managing the approvals on the table. Having the approval engine turned off prevents conflicts with the workflow that could cause a range of issues. If you want to use an approval engine on the table, set the workflow to inactive.



2. Select the approval engine option for each Task table from the choice list.
3. Click **Save**.
   These preferences are saved as system properties that are named *glide.approval_engine.<table_name>*.

Approval rules

Many organizations rely on an approval process to ensure that requests are reasonable and fit an organization's budget.

Important:
Classic approval rules have been replaced by the Workflow Studio Ask for Approval action. Use Workflow Studio to create workflow-driven approval logic that is easier to maintain and provides better reporting information.

The service catalog can use these classes of approvals:

- Gating approvals: Must occur before a request can be initiated. For example, allow a manager to reject an employee's request for a company car. To learn more about gating approvals, see Gating approvals
- Process approvals: Take place within an execution plan process that has been initiated. For example, allow the security group to reject a request for access to SSN even though the employee's manager approved it. To learn more about process approvals, see Process approvals

Note:
To enable approval processes to operate smoothly, make sure that the appropriate users have the correct role, and that the role grants access to the necessary tables for users in all the relevant departments and domains.

Set automatic approval rules

Approval rules can automatically set the approval state to something other than *Not yet requested*. As a result, an approval rule can create a set of approvers. You can also start the approval process by setting the approval state to **Requested**.

Prerequisites

Important:
Classic approval rules have been replaced by the Workflow Studio Ask for Approval action. Use Workflow Studio to create workflow-driven approval logic that is easier to maintain and provides better reporting information.

Role required: admin

Approval rules have two new fields:

- **Run rule before**: If true, the approval rule runs before the record is inserted/updated.
- **Set State**: If this rule applies, then the task record's approval state is automatically set to this value.

Note:
The **Set State** field only behaves as expected if the **Run rule before** check box is enabled.

- In the example below, this rule automatically sets the state of the task to *Approved* thereby auto-approving the task.

Approval Rules fields

Gating approvals

A gating approval acts as a gate through which a request must pass before it can start.

Until all gating approvals are met, no notifications go out, no tasks get sent to technicians, and nobody starts working on the request in question.

Generate gating approvals with:

Gating approvals

Title	Description
Approval rules	Can apply to the service catalog as well as any other task table.
Item-based approvals	Flag specific catalog items as requiring specific approvals. Any requests for these items automatically require these approvals.

Set up a gating approval via an approval rule

You can set up a gating approval via an approval rule.

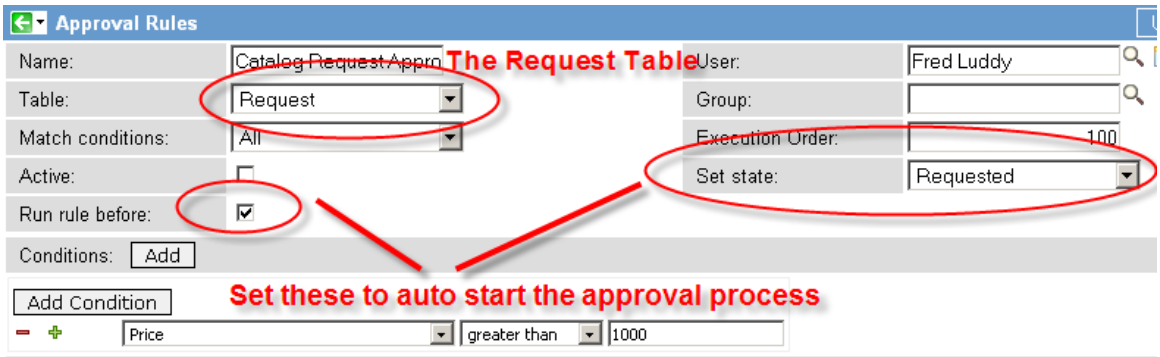
Before you begin

Role required: admin

> **Important:**
> Classic approval rules have been replaced by the Workflow Studio Ask for Approval action. Use Workflow Studio to create workflow-driven approval logic that is easier to maintain and provides better reporting information.

Procedure

1. From the left navigation pane, select **System Policy** > **Approval Rules**.
2. Click **New**.



Approval rules

Field	Description
Name	Name of this rule.
Table	Task table to which this rule applies. For most service catalog approvals, select **Request**. **Note:** The list shows only tables and database views that are Application scope as the approval rule.
Active	Indicator of whether the rule is active (defaults to true).
Run Rule Before	Indicator of whether the rule runs before or after the request record is saved. For most approvals, select this check box.
User	User who must approve this request (can be empty).
Group	Group that must approve this request (can be empty).
Set State	Value of the approval field on the task in after this rule runs. In most cases, select **Requested**.
Condition	Condition under which the rule applies.
Script	An optional server script to programmatically specify who the approver should be. For example, for the one-line script `current.requested_for.manager`, ServiceNow checks the requested_for reference field on the current record. It then locates the manager field on the referenced record and assigns that person as the approver. For other examples, see the Script field on approval rules provided by ServiceNow.

Notes and limitations:

a. You can have as many rules as you want on a given table. If more than one rule applies, you will get more than one approver.

b. You can't get duplicate approvers, for example, if two rules both want Fred Luddy to approve a particular request, the system will only create one approval entry for him.

c. By default all requests start out in a **Not yet requested** approval state. Approval notifications will not go out until the request's approval state is set to **Requested**. You can do that manually, or you can do it in script, but the easiest way to do it is to use the **Set State** field to automatically set the request to **Requested**.

Set up a gating approval based on the item being ordered

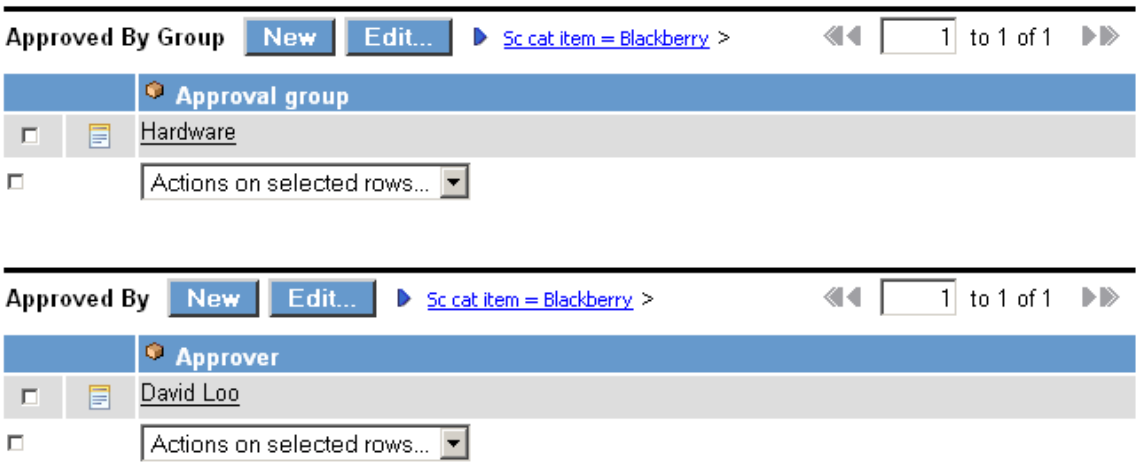
In addition to adding approvals via approval rules, you can also add approvals based on what kind of item is being ordered.

In addition to adding approvals via approval rules, you can also add approvals based on what kind of item is being ordered. We can, for example, specify that all Blackberrys need to be approved by David Loo.

To do so, navigate to the item in question and scroll to the related list of required approvers. There are two lists:

- **Approved By Group**: A list of groups that have to approve requests for this item
- **Approved By**: A list of users who have to approve requests for this item

Approve list



In the example above, this request must be approved by all members of the Hardware group and by David Loo.

Notes and limitations:

1. As with approval rules, you are protected against duplicate entries. Thus if a person is a member of the hardware group, as well as being a standalone approver, the person will only get one approval request.
2. Item-based approved work in addition to rather than instead of approval rules so you can (and probably will) use both.

Process approvals

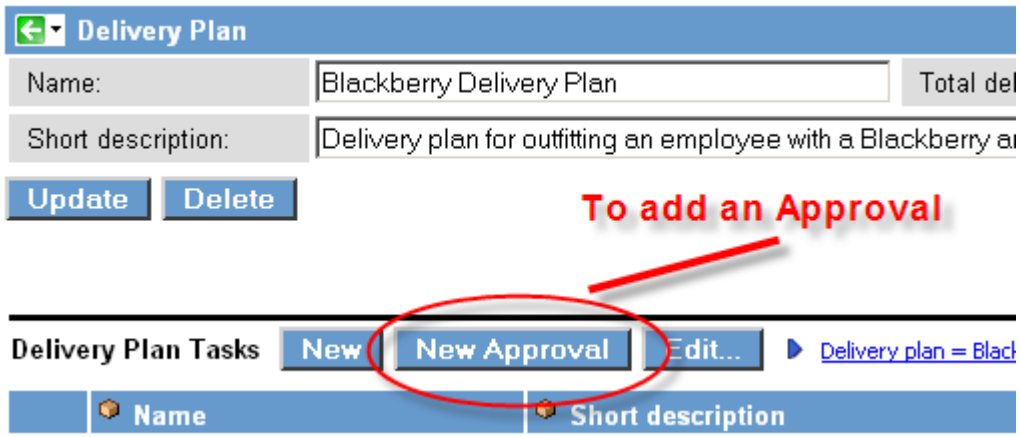
Once a request has passed its gating approvals, any relevant execution plans are initiated.

Important:
Classic approval rules have been replaced by the Workflow Studio Ask for Approval action. Use Workflow Studio to create workflow-driven approval logic that is easier to maintain and provides better reporting information.

Those plans, in turn, create a sequence of required tasks. You can add an approval step to an execution plan, which is configured to occur at the appropriate point.

From the left navigation pane, select **Service Catalog** > **Execution Plans**, and then select the plan to which to add an approval step. Then click the **New Approval** button.

Add approval

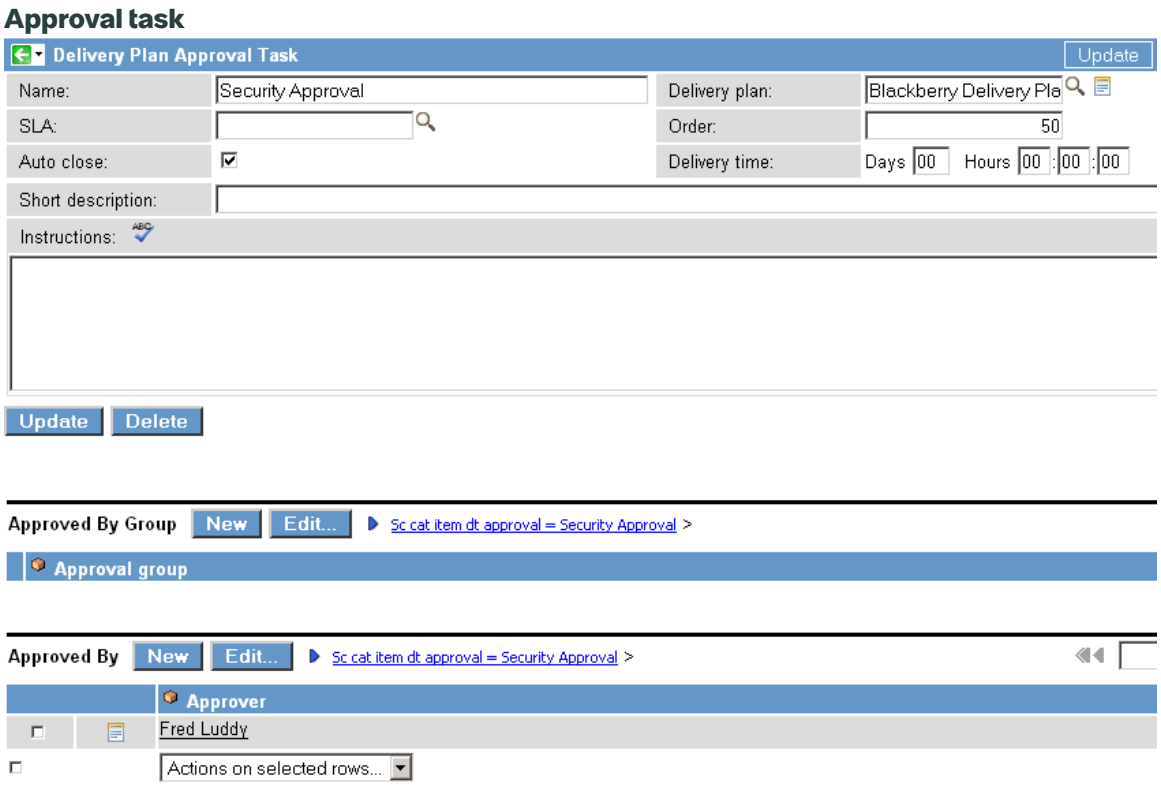


The Approval Task screen appears. Just like a regular Service Catalog execution task, an approval execution task has:

- **Name**: The name of this task
- **Order**: Sequence of this task within the plan
- **SLA**: SLA to which this task applies
- **Delivery Time**: Time allowed for the completion of this task

After you create the task, right click the title bar and select **Save**. Two related lists appear at the bottom of the screen:

- **Approved By Group**: A list of groups that must approve the request before this task is complete
- **Approved By**: A list of users who must approve the request before this task is complete



In the example above, this security approval task must be approved by Fred Luddy.

Note:
When an in-process approval is rejected, that particular line item is canceled as well, but the request itself isn't necessarily canceled. Thus if one ordered a blackberry and a laptop, and the blackberry was rejected, the laptop request would continue being processed.

Approve with a process guide

Process guides work similarly to approval rules in that their execution is controlled via a condition.

Before you begin
Role required: none

About this task
The default version of approval tasks allows you to specify that the approval in question be approved by:

1. One or more specific people
2. One or more groups of people

You can optionally use Process Guides instead of approval tasks. Process guides are more flexible in that they allow for:

1. "Any of" or "All of" approvals
2. Sequenced approvals

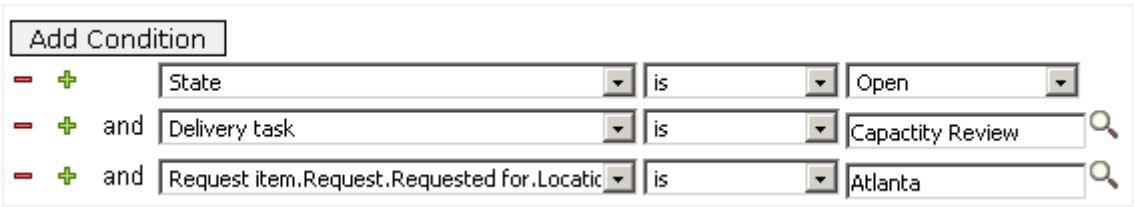
You can link a process guide to a execution task.

Procedure

1. From the left navigation pane, select **System Policy** > **Process Guides**.
2. Create a new guide.
3. Set the table to **Catalog task**.
4. Fill in a condition under which this guide should attach.

   Example #1: Apply to all "Capacity Review" tasks.

   Example #2: Apply to all "Capacity Review" tasks where there requester is in Atlanta.



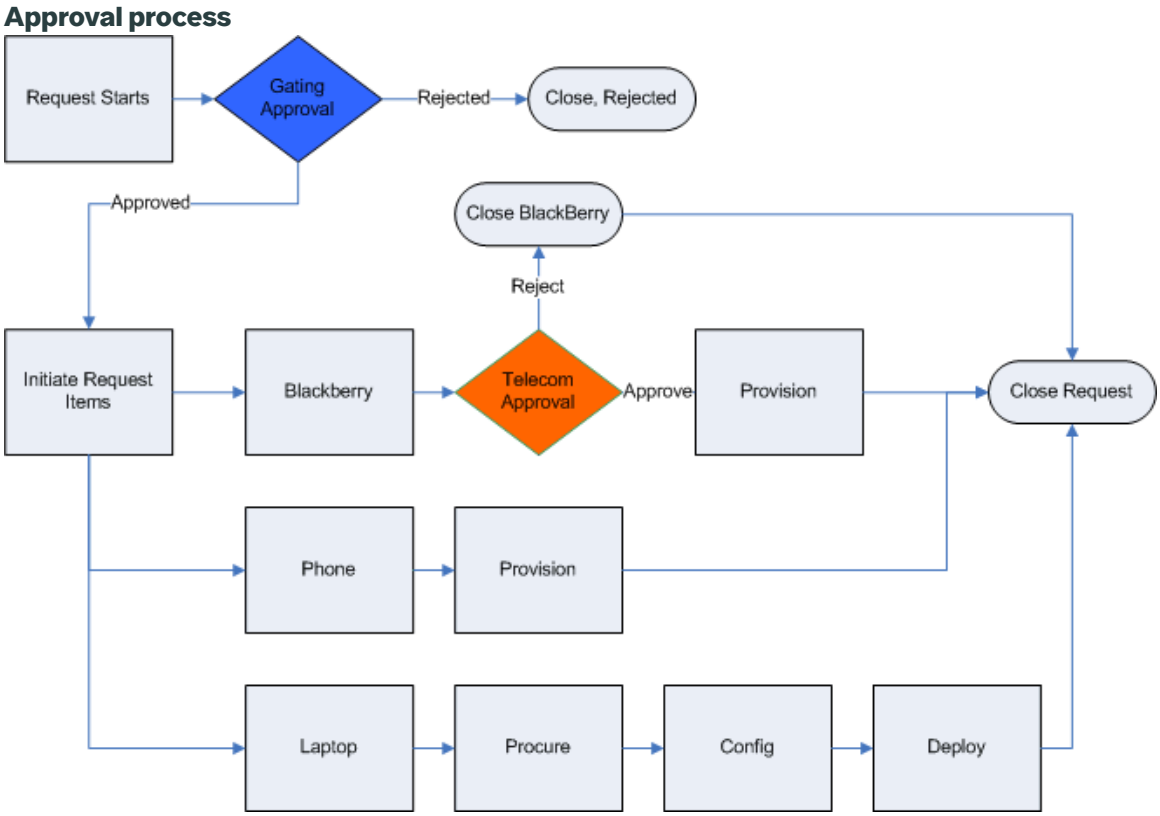
   Process guide tips and tricks:

   a. All catalog tasks are generated when a request is first submitted, but tasks which aren't active yet have a state of "pending". So if you do not want to send out approval requests until a task has actually started, add "state=open" as part of your condition.

   b. There is a "Default" process guide in the system for catalog tasks with a sequence number of 10,000. It behaves exactly the same way the old, pre-process guide code did in regards to approvals. Approvals are based on the execution of task-related lists.

Schematic of a hypothetical approval process

In the diagram below of a hypothetical approval process, the gating approval is color coded blue and an in-process approval is orange.

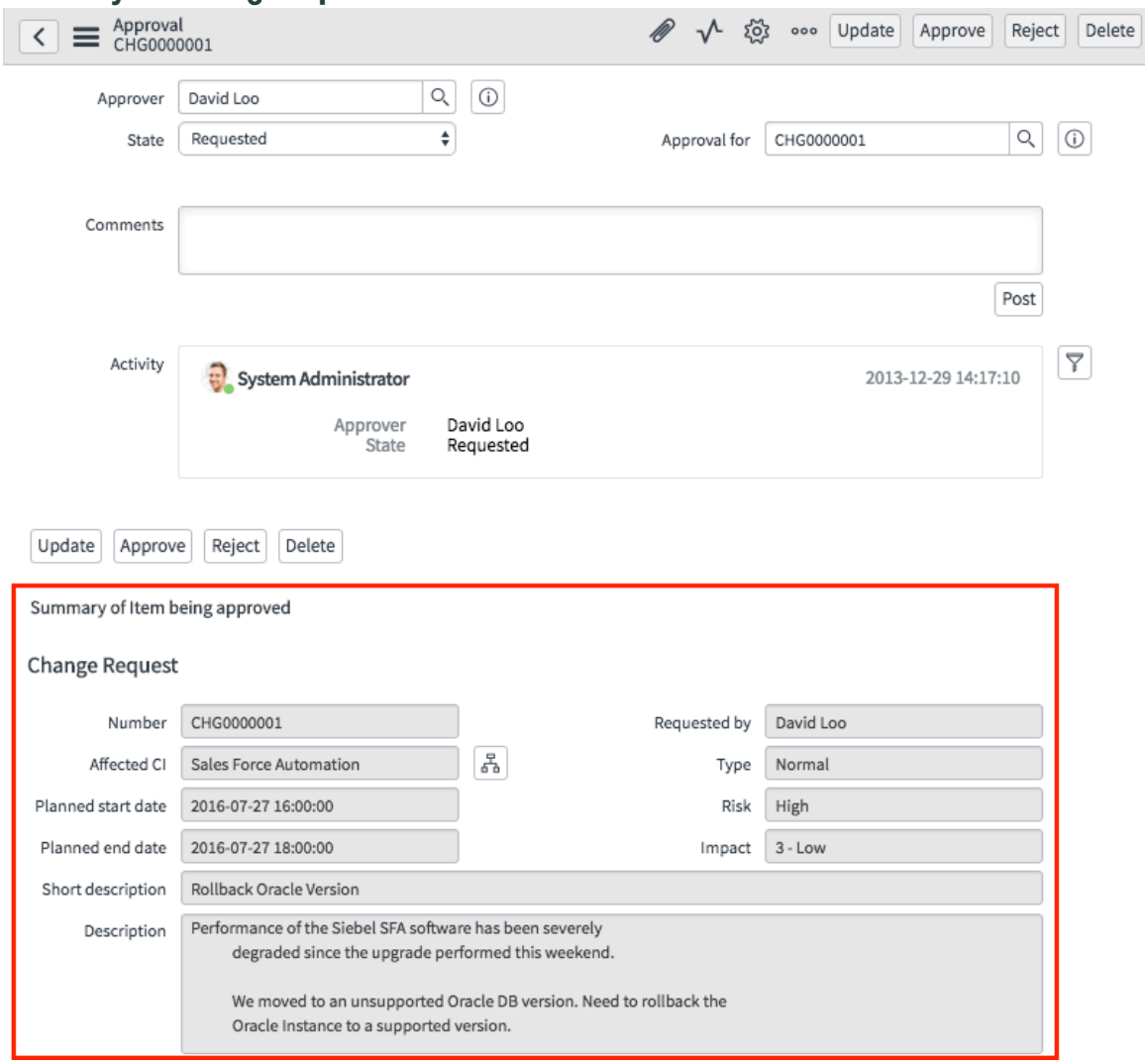


Approval summarizer formatter

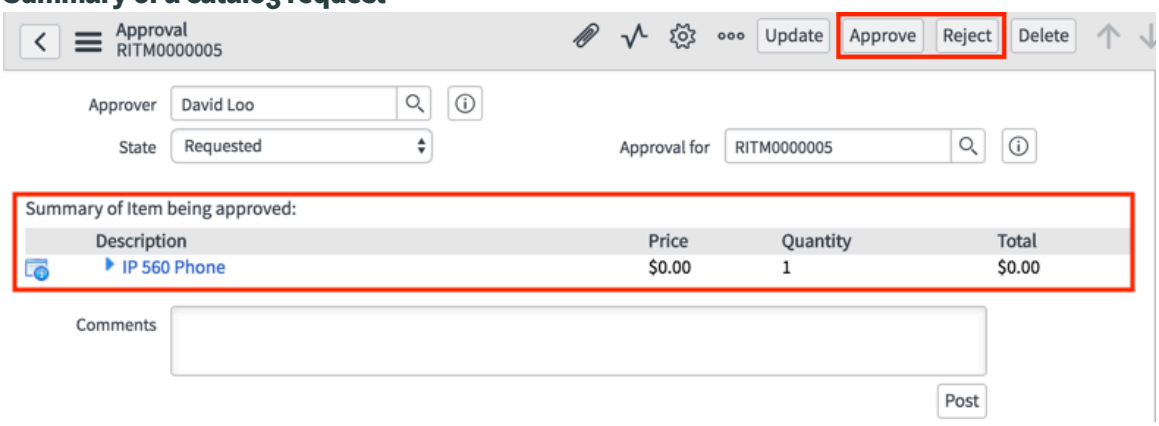
The approval summarizer formatter creates the summary at the bottom of an approval form.

The approval summarizer displays different information depending on what is being approved, such as a change request or a service catalog request. Following are two examples.

Summary of a change request



Summary of a catalog request



The **Reject** button allows the approver to deny one or more requested items in a multi-item request, before approving the overall request. If a requested item is denied, the workflow for that item never starts. The approver can then choose to **Accept** the item.

Note:
When the overall request is approved, you must ensure this **Reject** button is hidden. If this button is used after request approval, the requested item workflow is canceled, leaving the stage in an inconsistent state. Similarly, the **Accept** button on requested items should only appear before the overall request is approved or rejected.

Summarizers

Approval summarizers are stored in the Macro [sys_ui_macro] table.

From the left navigation pane, select **System UI** > **UI Macros**. Summarizers use a naming convention of approval_summarizer_ + '<table_name> (for example, approval_summarizer_change_request is the summarizer for change requests, while approval_summarizer_sc_request is the summarizer for service catalog requests).

Each summarizer is written in Jelly script, which is used to define internal forms. The script is stored in the large XML field at the bottom of the UI Macro form.

Change an approval summarizer

You can modify existing approval summaries to include additional information.

Before you begin
Role required: none

About this task
These are advanced customizations that might not be appropriate for all implementations, and require creating a custom form.

Procedure

1. Navigate to **All** > **System UI** > **UI Macros**.
2. Open the summarizer you want to change.
3. Copy the script to another location before editing, in case you need to revert it.
4. Modify the script.
5. Click **Update**.

Create a new custom approval summarizer

After you add a new table that has approvals to an instance, you can add a custom activity formatter by creating a new UI macro and then add it to the appropriate form.

Before you begin
Role required: admin

About this task
approval_summarizer can only be used on approval forms in the global scope.

Procedure

1. Navigate to **All** > **System UI** > **UI Macros**
2. Click **New**.
3. Give the macro a name that follows the summarizer naming convention:
   `approval_summarizer_<tablename>`
4. Complete the rest of the form and click **Submit**.

5. Create a formatter and add it to a form and add it to the appropriate form.
   To learn more about activity formatters, see Activity formatter.

Approval with e-signature

Approve or reject an approval record by re-entering your login credentials. See the history of an approval from its activity stream or its audit history.

Support for Title 21 Code of Federal Regulations (CFR) Part 11

The Approval with e-signature plug-in is validated for Title 21 CFR Part 11 electronic signature requirements. This plug-in is required for compliance to 21 CFR Part 11.

Each time someone approves a record, the system stores this information.

- The printed name of the signer
- The date and time when the signature was made
- The meaning of the signature (approval or rejection)

This information is stored as part of the activity stream and audit history of the record.

Approval tables

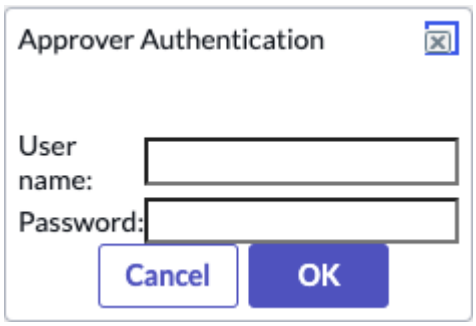
By default, Approval with e-signature supports these tables.

- Change Request [change_request]
- Standard Change Proposal [std_change_proposal]

When you add an approval table, any approval for the table needs an e-signature authorization. See Select an approval table to add or remove a table from supporting Approval with e-signature.

Approver Authentication dialog

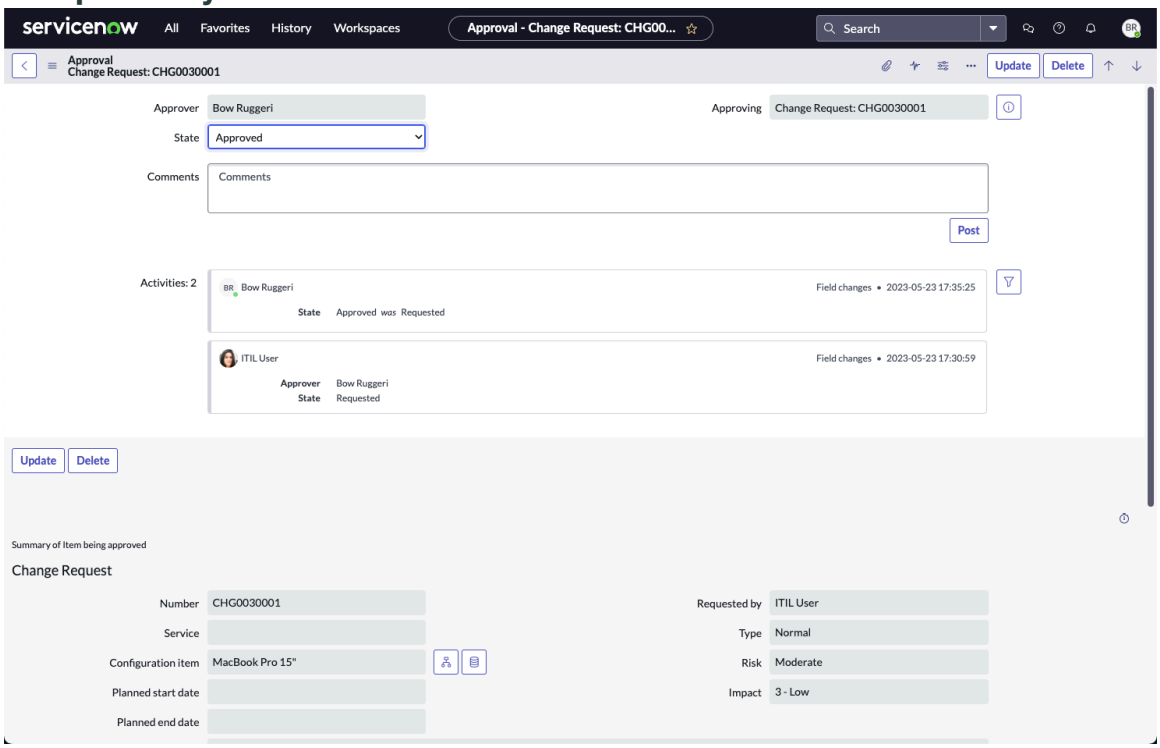
Approvers must enter their user credentials to change the approval state. If the credentials fail, then the approval remains unchanged. The system checks the user credentials against local user records or an active single sign-on integration.



Approval History

Each Approval [sysapprover_approver] record has an activity stream that tracks changes to it. Approvals made by e-signature are stored as field changes.

Example activity stream



Available approval options

The Change Request [change_request] table provides these approval options.

List options

The Change Request table provides context menu options to approve or reject records. When someone selects either the Approve or Reject context menu option, the system displays the Approver Authentication confirmation dialog.

UI actions

The Change Request form provides UI actions to approve or reject a record. When someone selects either the Approve or Reject UI action, the system displays the Approver Authentication confirmation dialog.

Change record state

The Change Request table provides a State field to approve or reject a record. When someone selects either the Approved or Rejected state, the system displays the Approver Authentication confirmation dialog.

User credential sources

Approval with e-signature supports these types of user credential sources.

- A set of local User [sys_user] records.
- A remote identity provider service accessed by the Multi-Provider Single Sign-On (SSO) integration.

User credentials must be set up before enabling Approvals with e-signature. See Creating users to create local User records.

See Multi-Provider Single sign-on (SSO) to create an integration to a Security Assertion Markup Language (SAML) 2.0 identity provider.

Activate Approval with e-Signature plugin

The Approval with e-Signature plugin (com.glide.e_signature_approvals) allows users to approve requests by re-entering their login credentials.

Before you begin

Role required: admin

Note:
You must install the Code Signing Signatures (`com.glide.code_signing.signatures`) to install the E-signature plugin.

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.

De-activate e-signatures

Use this procedure to de-activate e-signatures.

Although plugins cannot be removed, e-signatures can be disabled.

1. Navigate to **System Definition** > **E-Signature Registry**.
2. Set **Enabled** to **False** on any tables where e-signatures are no longer required.

Select an approval table

By default, activating the Approval with E-signature plugin enables e-signature for all tables for which an approval exists.

Before you begin

Role required: admin

About this task

E-signature approvals can also be enabled on a table-by-table basis.

Procedure

1. Navigate to **All** > **System Definition** > **e-Signature Registry**.
2. Select **New**.
3. In **Table name**, use the drop-down list to select a specific table.

Table Descriptions

Field	Input Value
Table	Table that requires e-signatures for approvals.
Enabled	Option to require e-signature for approvals. Clear this option to remove the e-signature requirement.

4. Select **Submit**.

Set up an approval from a local database

Enable users to authorize e-signature approvals using local database credentials.

Before you begin
Role required: none

Procedure

1. Activate the **Approval with e-signatures** plugin.
2. Create user records for approval users.

Use Multi-Provider SSO to set up an SSO approval for a SAML 2.0 authentication

An SSO approval with e-signature requires configuration on the SAML IdP and the ServiceNow instance.

Before you begin
Role required: admin

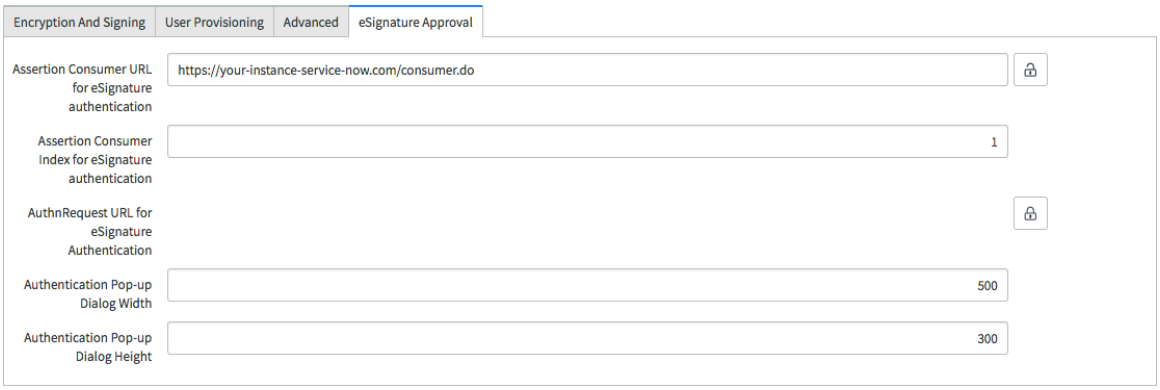
About this task
The SAML IdP must support and honor the forceAuthn attribute in SAML assertion requests. E-signature doesn't function without this IdP setting. Set up an approval with e-signature using credentials from a SAML 2.0 authentication.

Procedure

1. Activate or upgrade to SAML 2.0 with the Activate Multi-Provider SSO plugin.
2. Activate the Approval with E-Signature plugin.
3. Navigate to **Multi-Provider SSO** > **Identity Providers** and verify your 2.0 SAML IdP configuration Advanced tab shows the **Force AuthnRequest** attribute checked.
   Your SAML 2.0 IdP must support the **Force AuthnRequest** attribute, or e-signature isn't supported.
4. On the eSignature Approval tab, enter the following e-signature SAML properties.

Option	Description
Assertion Consumer URL for eSignature authentication	This property defaults to the appropriate URL. To configure this property, select the lock icon to make this field editable. After edits, select the icon to lock the field.

Option	Description
Assertion Consumer Index for eSignature authentication	If your Service Provider has more than one URL set for the AssertionConsumerURL, you can set the index to use for eSignature, start ing with index 1 or more.
AuthnRequest URL for eSignature Authenti cation	You can enter the URL that points to the SAML 2.0 IdP AuthnRequest URL for eSigna ture authentication. If the URL is the same as the Assertion Consumer URL, you can leave this setting empty.
Authentication pop-up Dialog Width	When a user approves a request using eSig nature, a dialog opens and a user can enter credentials. This setting controls the width of that dialog box. The default is 500.
Authentication pop-up Dialog Height	When a user approves a request using eSig nature, a dialog opens and a user can enter credentials. This setting controls the height of that dialog box. The default is 300.



5. Select the **Generate Metadata** button underneath the tabs to regenerate the service provider metadata.
6. Copy the service provider metadata, and update it on the SAML IdP.

Installed with approval with e-signature

Installing approval with e-signature installs certain properties.

- Module - E-Signature Registry
- UI Action - Approve (on table sysapproval_approver, with no action name)
- UI Action - Approve (on table sysapproval_approver, with no action name)
- UI Action - Approve (on table sysapproval_approver, with the action name authenticated_list_approval)
- UI Page - form_login_validate_dialog
- UI Page - login_validate_dialog
- UI page: saml2_esignature_login, the re-authentication page that appears when an approver tries to approve a request.

- Properties: see E-signature SAML properties
- Client Script - Authenticate Approver
- Script Include - User
- Script Include - UserAuthentication
- Processor: eSigSaml2AssertionConsumer

Installing the plugin also disables the two out-of-the-box Approve UI Actions on the `sysapproval_approver` table.

Approval status

The approval status of a change request is determined by looking at the current status of all the approvers.

If any approver has rejected the change, the approval status will be Rejected. If all approvers have approved the change, the approval status will be Approved. If all approvers are in the Not Requested status or if there are no approvers, the change status will be Not Requested, otherwise the status will be Requested.

For added flexibility when creating approvals, including the ability to set up an "one of" approval where only one person of a group of approvers needs to approve, consider using Workflows .

Related topics

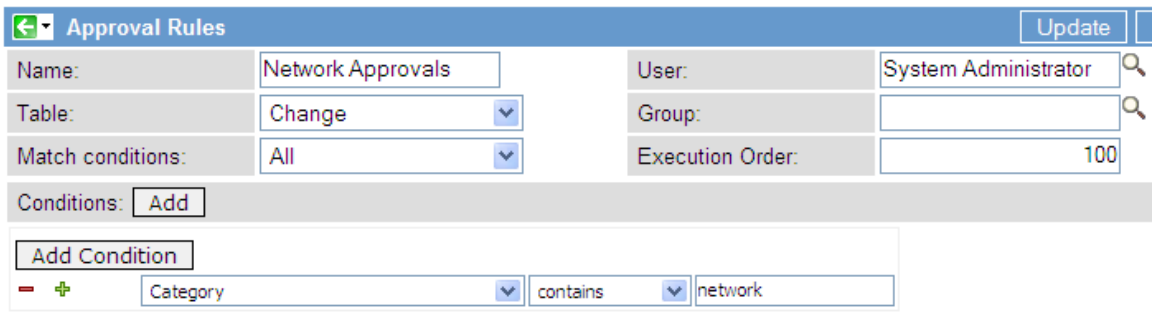
Classic approvals

Generate an approval using approval rules

The system can automatically generate an approval request to individuals or groups when specific criteria are met. The automatic generation of approval requests is driven using the System Policy feature.

In the sample below, a change opened in the category **network** is assigned to the System Administrator:

Approval Rules



When an approver is automatically added based on approval rules, the status of the approval automatically defaults to "Requested".

Related topics

Classic approvals

Generate approvals using the approvers related list

It is possible to manually add approvers to a request.

Additional approvers can be added by clicking the **Edit** button in the Approvers section near the bottom of a request. When an approver is added manually, the status for that approver defaults to **Not Yet Requested**. When the status of the approver changes to **Requested**, the approver is sent an email requesting approval action.

Related topics

Classic approvals

Generate approvals using Workflow flows

Workflow Studio flows are a powerful and flexible method of generating approvals. Use workflow flows to create group approvals and user approvals.

A variety of variables are available to fine-tune the approval process, including the actions that occur when approval or rejection take place. When a Workflow Studio flow generates an approval record, the system populates the **Flow** field on the approval record with a reference to the flow and the Ask for Approval action. Do not use this field when creating business logic. For more information, see Ask for Approval action.

Note:
Conflicts can arise when the approval process for a table is managed by both the Workflow Studio engine and the approval engine. In general, if there is a Workflow Studio flow that manages the approval process for a table, the approval engine should be turned off for that table.

Related topics

Classic approvals

Multiple approvers

With multiple approvers, all approvers must authorize the request before the status will change to "Approved".

Should any approver reject the request, the status will immediately be set to "Rejected".

Related topics

Classic approvals

Receive notifications

Individuals who are designated approvers automatically receive approval notifications, including approval status updates.

Approval notifications will be sent at the following times:

- When an individual is assigned as an approver either automatically or manually. If a group is chosen, then all members of the group will be sent an email. By default, the email an approver receives will contain a "mailto" link that will allow the approver to either approve or reject the request directly from their email system.
- When the request reaches approved status, the person assigned to the request will receive an email indicating it has been approved.

The details contained in the emails and the points at which they are sent can be tailored using **System Definition** > **Business Rules** and **System Policy**.

Note for Blackberry users: In order to see the "mailto" links mentioned above to approve or reject a request (i.e. 'Click here to approve CHG55555' or 'Click here to reject CHG55555'), your Blackberry device must be using version 4.5 of their software which supports HTML emails. If your Blackberry device is using an earlier version, you will not be able to view or use the "mailto" links. However, as a workaround, users can reply to the email and add the statements `state:approved` or `state:rejected` within the body of the email before sending it to force the automatic approval/rejection functionality.

If you create an appropriate Inbound Email Action, you can let approvers respond to approval email notifications with a simple "yes" or "no" answer.

Embed an approval request within the Outlook email client

Embed an interactive approval request for service catalog requests in the email notification sent to a user. The user can accept or reject the approval request from the email client.

Before you begin
Role required: admin
The Outlook Actionable Messages plugin (com.sn_ms_oam) should be activated.

Note:

- The Outlook Actionable Messages (OAM) feature is not supported in all Microsoft mail products and versions. To verify whether your version of Outlook supports OAM, refer to the Microsoft documentation.
- Actionable messages are supported only for emails sent from the @service-now.com email address. If you are sending an email from a customized email address, you should register as a new service in the Microsoft website setting the scope as **Organization**. Specify the provider ID value in the *`sn_ms_oam.outlookactionable.originator`* property.
- Actionable messages are based on the Sender Policy Framework (SPF)/DomainKeys Identified Mail (DKIM) validation for the email sender verification. If an email recipient receives email via an external provider, emails may not be rendered as adaptive cards.
- You cannot customize the default actionable message templates.

Procedure

1. Navigate to **All** > **System Notification** > **Email** > **Notifications**.
2. For the approval notification that requires user approval, in the **What it will contain** tab, add the following script in the **Message** field in addition to the existing information, or to the **Message** field in the configured email template if applicable.

   ```
   ${mail_script:include_approval_actionable}
   ```

   For example, for the Catalog Approval Request and Approval Request notifications, you can include the script in `request.itil.approve.role` and `change.itil.approve.role` email templates.

   This script includes the Outlook actionable message in the email notification sent to the user for approvals.
3. Click **Update**.

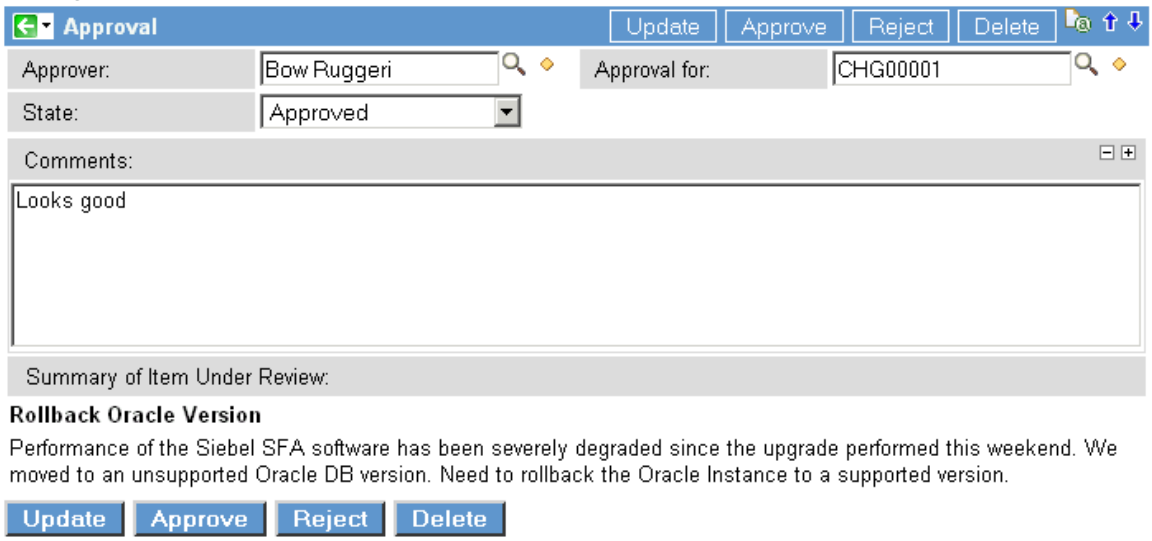
Dynamic approval forms

When you are looking at an approval request, the form has a context-appropriate summary of the item to be approved.

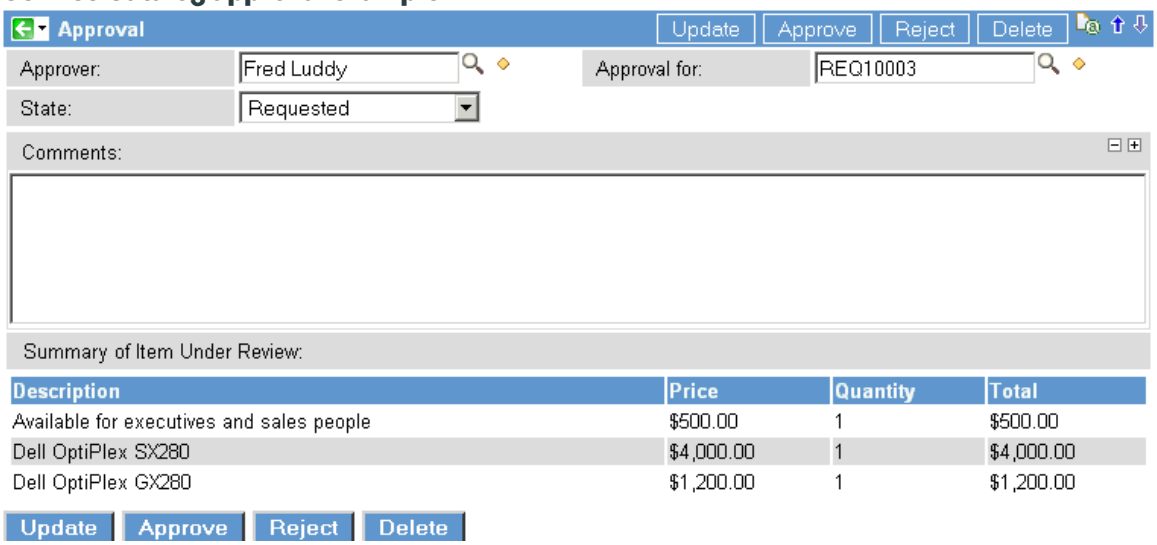
Important:
Classic approval rules have been replaced by the Workflow Studio Ask for Approval action. Use Workflow Studio to create workflow-driven approval logic that is easier to maintain and provides better reporting information.

For example, if you're looking at a Change Management approval request, you will see details from the relevant change request. For a Service Catalog approval request, you will get details of the request.

Change request approval example



Service Catalog approval example



Scripts and engines execution order

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.

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.

When the business rule should run (continued)

Option	When the rule runs
	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.

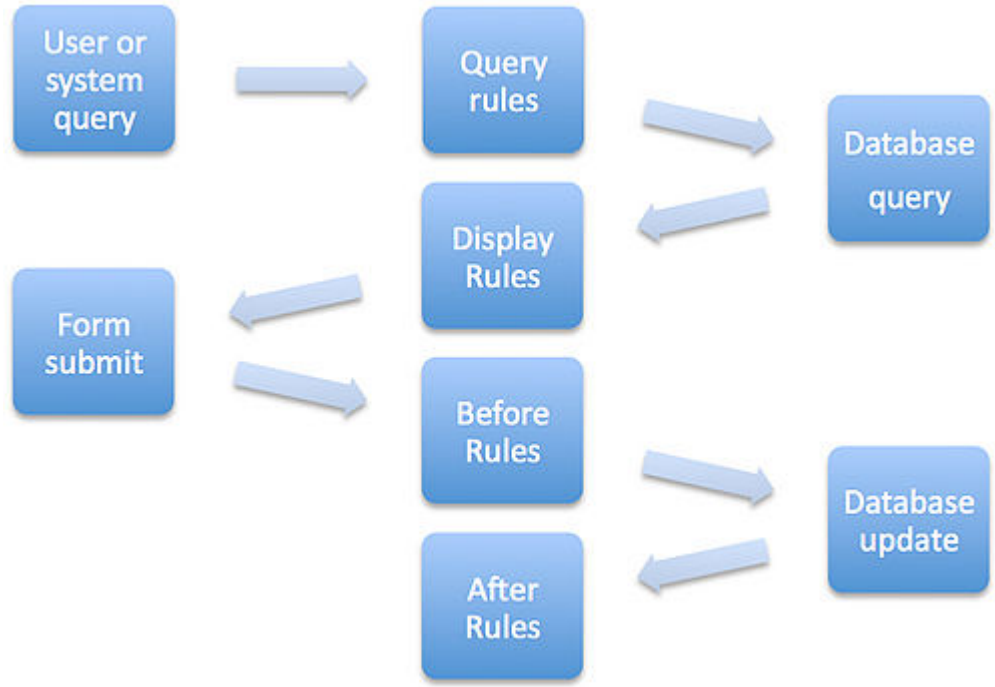
What record operation the business rule applies to

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.
Accessible from	Scope protection for a global business rule.

Field	Description
	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.

System Events

System Events in ServiceNow are crucial for automating processes and maintaining platform efficiency. They act as triggers for business rules, notifications, work-flows, and other actions, ensuring timely and consistent execution of tasks. By monitoring and responding to these events, organizations can streamline operations, improve response times, and enhance user experiences.

Get started

Explore

Learn about how system events function as triggers for actions and enable process automation.

Configure events

Set up and manage system events to trigger actions and automate processes.

Manage events

Connect system events with external systems and applications to enable seamless data exchange and automated work-flows.

Reference

More information about the predefined system events available and their usage.

Exploring system events

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.

System events overview

Learn about how system events function as triggers for actions and enable process automation.

Realize the benefits of leveraging system events by doing the following tasks in your instance:

- Create events
- Log events
- Implement automated job scheduling
- Move an event
- Register events
- Reprocess events
- Pass event parameters from a work-flow to a notification

System events users

Users

User	Description
Administrator	Administrators leverage system events to trigger automated actions and work-flows based on specific occurrences in the platform, enabling asynchronous processing and responsive automation. They can generate events via scripts or business rules, register them in the Event Registry, and handle them using notifications, script actions, or flows.
Business analysts	Business analysts analyze trends and processes by leveraging events to capture specific system activities, generate logs, or send data to analytic tools for better decision-making and process improvement.
Compliance officers	Compliance officers help to ensure adherence to regulatory requirements by using events to log critical activities, generate compliance reports, or initiate corrective actions when deviations from standards are detected.
Customer support	Customer support employees use system events to enhance customer service by triggering automated responses to customer requests, creating follow-up tasks, or escalating unresolved cases to appropriate teams.
Developers	Developers create and manage system events to integrate modules, trigger work-flows, and implement custom logic in response to user actions or record changes, helping to ensure extensibility and dynamic system behavior.
HR administrators	HR professionals use system events to automate HR processes, such as notifying employees of updates, generating on-boarding work-flows when a new hire is recorded, or triggering document reviews during off-boarding.
ITOM/ITSM	IT service managers use system events to monitor and automate ITSM processes, such as sending notifications for SLA breaches, escalating incidents, or triggering task work-flows, helping to ensure efficient IT service delivery and compliance with SLAs.
Marketing teams	Marketing teams automate customer engagement by using events to trigger campaign actions, such as sending promotional emails or updating CRM systems when specific conditions (e.g., customer interaction) are met.
Operations teams	Operations teams monitor and respond to operational alerts or system changes by triggering events that initiate corrective actions, such as restarting services, updating CMDB records, or notifying relevant teams of maintenance schedules.
Project managers	Project managers use system events to trigger notifications for project updates, create task dependencies automatically, or update stakeholders when milestones are achieved, enabling better project tracking and communication.
Security analysts	Security analysts leverage system events to trigger immediate responses to security incidents, such as creating alerts,

Users (continued)

User	Description
	generating logs, or initiating containment work-flows in response to detected threats or vulnerabilities.

System events benefits

System events benefits

Benefit	Feature	Users
Allows users to define new events that can be triggered by specific conditions or actions, enabling customized work-flows and automated responses.	Create an event	Developers, Administrators, IT Service Managers
Captures event activity in logs for auditing, debugging, and monitoring system behavior, providing insights into event performance and bottlenecks.	Event logs	Administrators, Compliance Officers, Security Analysts
Schedules jobs to be executed automatically at specific times or intervals, ensuring routine tasks are performed without manual intervention.	Implement automated job scheduling	Operations Teams, IT Service Managers, Administrators
Transfers events between queues or categories, helping optimize processing priority and load balancing in event-driven systems.	Move an event	Administrators, Developers
Registers custom or system-defined events in the platform to make them available for triggering specific actions or work-flows.	Register an event	Developers, Administrators
Provides the ability to reprocess failed or incomplete events, ensuring system reliability and minimizing the impact of errors.	Reprocess an event	IT Operations Teams, Developers, Administrators
Enables seamless transfer of contextual data from work-flows to notifications, ensuring messages contain relevant and actionable information for recipients.	Pass event parameters from a work-flow to a notification	Developers, IT Service Managers, HR Professionals

What to explore next

To learn more about configuring and using System events, see:

- Configuring system events
- Managing system events
- System events reference

System Events

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.

The system uses business rules to monitor for system conditions and to generate event records in the Event [sysevent] table, which is also known as the event log or event queue.

Event-generating business rules typically use this script logic:

If [some condition is true for the current record], then [add a specific event to the queue].

For example, here are some of the conditions in the **incident event** business rule:

- If a user adds a comment to an incident record, add an *`incident.commented`* event.
- If a user adds an incident record, add an *`incident.inserted`* event.
- If a user updates an incident record, add an *`incident.updated`* event.

Event-generating business rules use the GlideSystem `eventQueue` method to insert event records, which typically contain this information:

Event fields

Field	Description
Name	Unique name of event. Baseline event names include the record affected and the triggering action, such as *`incident.commented`*.
Parm1	Event-specific parameter the system uses to pass record information to other parts of the system, such as a record Sys ID or a field value.
Parm2	Event-specific parameter the system uses to pass record information to other parts of the system, such as a record Sys ID or a field value.
Table	Table to which the event applies. This is the same table on which the business rule ran.
Instance	Sys ID of the record to which this event applies.

Scheduled jobs periodically read the event queue and forward them to the appropriate handler for processing. The handler uses information from event records to take some kind of action such as:

- Run a script action
- Send a notification
- Trigger a workflow activity
- Trigger an inactivity monitor

By default, the system provides events covering a broad view of application activity. If existing events do not meet your needs, you can create your own events to watch for specific changes to records.

For developer training, see Scheduled Script Executions and Events Objectives on the ServiceNow® Developer Site.

Use the System Events and Jobs Dashboard to monitor the system event processing system and the scheduled jobs processing system.

Event processing during platform upgrade

Determine which events are to be processed during a platform upgrade by configuring the following properties:

- **glide.event_processor.all_events_upgrade_safe**: Set to **true** to process all events during platform upgrade. Default = **false**.
- **glide.event_processor.upgrade_safe_events**: Configure a comma-separated list of event names to indicate that only specified events are processed. Default = <empty>.

  This property is relevant only when *glide.event_processor.all_events_upgrade_safe = false*.

Event registry

The events registry lists the events the system recognizes. Use registered events to automate other activities, such as script actions or notifications.

After you create a new event and a business rule that uses the event, you must register it.

Registration lets other parts of the system, such as Email and SMS notifications and Script Actions, see the event in their list of available events and react to the event when it occurs.

Related topics

Register an event

Configuring System events

Plan, configure, and implement system events without having to complete any extra configurations first. Follow the task listed in the configuration overview to implement automated job scheduling.

Configuration overview

Implement automated job scheduling

Implement the message processing framework (automated jobs scheduling) by using the Queue Registration link on the Event Registration form.

Implement automated job scheduling

Implement the message processing framework (automated jobs scheduling) by using the Queue Registration link on the Event Registration form.

Before you begin
Role required: admin

Procedure

1. Navigate to **All** > **Performance Analytics** > **System** > **Event Registry**.
2. Select **New** to register a new event.

   Note:
   You can also select an existing event to update the configurations.

3. Click the Queue Registration link in the banner message on the Event Registration form.

   Note:
   You can also disable automated job scheduling by selecting **Disable Automatic Job Scheduling** button on the Queue Registry list page.

If you have started a new event, the Queue Registry list shows up. If you have selected an existing event, the Queue Registration new record shows up.

4. Select **New** to create a new queue.
   The Queue Registration form shows up.
5. On the form, fill in the fields.

Field	Description
Queue	Name of the queue
Application	Scope of the application
Automatic Job Scheduling	Option to implement automated job scheduling
Event Processing Order	Option to select the way of processing events ○ Parallel: Process multiple events simultaneously ○ Sequential: Process only one event at a time. The events are inter-dependent and can process only after the previous event is completed **Note:** Once you have configured the processing order and submitted the queue, it can't be edited. You can delete the queue and create a new queue.
Job configuration type	Type of job configuration **Note:** This field is disabled when you select Sequential as the Event Processing Order. If you select Parallel as the Event Processing Order, you have 2 options. ○ Constant: Creates the required number of jobs ○ Scale with node: Multiplies jobs by the number of nodes. You can use this option if you are unaware of the number of available nodes
Scale factor	Total number of jobs based on the Job configuration type selected
Poll interval	Determines how frequently jobs should poll and process events of this queue
Description	Optional field to describe about this queue related information

6. Click **Submit**.
   The Queue Registry list shows up. It also shows the update made to the recent queue.

7. Select the recently added queue.
   The Queue Registration form shows up.
8. Scroll down to the Related lists section.
9. Select the **Status** entry in the Queue Details related list.

   Note:
   Since the new queue has been submitted, the Status entry is Active.

   The Queue Details form shows up.
10. **Optional:** Update the number of jobs for this queue as per the requirement.
    You can now manually scale up or down as per the requirements without having to configure everything and can eventually avoid misconfiguration.
11. **Optional:** Click Rollback related link on the Queue Registration form if you want to roll back a queue to its previous configurations without automated job scheduling.

    Note:
    The Rollback related link shows up only if you are in an automated jobs scheduling configuration.

12. **Optional:** Select **Retrieve Configurations** if you want to retrieve any existing job configurations for this queue.

    Note:
    This process is possible only if you are in an automated jobs scheduling configuration.

    If any queue with your suggested queue name already exists in the manual jobs scheduling configuration, the configurations of the existing queue are retrieved. You can then validate and update them as required.

Managing system events

Managing system events involves creating, monitoring, and handling events to drive automation and streamline processes. Administrators and developers can define custom events, register them in the system, and set up actions or work-flows that respond to these events. This functionality helps ensure efficient task execution, helps improve system monitoring, and enables more robust error handling for enhanced operational reliability.

Overview of managing system events

- Create an event

  Create custom system events.

- Move an event

  Accelerate the execution process by moving high priority events from the default queue to an isolated queue.

- Pass event parameters from a workflow to a notification

  Pass two event parameters that send information about a record or related records from a work-flow to a notification.

- Register an event

  Register an event for a specific table and a business rule that fires the event.

- Reprocess an event

  Re-fire an event for testing or diagnostic purposes.

System events reference

For more information, see System events reference.

Create an event

If you do not find a suitable existing event, you can create your own.

Before you begin

Role required: admin

About this task

The gs.EventQueue function works directly with the backend and therefore business rules that are called by gs.EventQueue() are not invoked.

Procedure

1. Navigate to **All** > **System Policy** > **Events** > **Registry**, and then select **New**.
2. On the form fill in the fields.

Event Registration Form Completion

Field	Description
Event name	Name of your new event.
Table	Database table for this event. **Note:** Only tables and database views that are in the same scope as the event appear on the listing.
Queue	Name of the queue that the event is placed into when triggered. Use only lowercase letters, no spaces, and no special characters except underscore (_). For example, `my_queue`. See Using custom queues to process events.
Priority	Order in which messages will be processed. Lower values have higher priority. **Note:** The com.glide.sysevent.priority.enabled system property is enabled by default. If you disable it, events processing is not done by priority.
Caller Access	Restricted caller access settings. ○ Caller Restriction: calls to the resource must be manually approved. ○ Caller Tracking: Calls to the resource are automatically approved.
Fired by	Name of the business rule that runs the event. This field is for reference only and is not used by any process. Make sure that there is enough information to locate your event again.

Field	Description
Description	Short description of the purpose of the event.

3. Click the **Business Rules** related link.
4. If you are creating an event for a base system table, select the existing event business rule for the table.

   Example
   For example, select the *sc request events* business rule to create a custom Request event.
5. If you are updating an existing event business rule, add a new condition to the **Script**.

   Example
   The following sample script adds a *request.commented* event with the user's Sys ID as parm1 and the user's user name for parm2.

```
if (current.operation() != 'insert' &&
 current.comments.changes()) {
gs.eventQueue('request.commented', current, gs.getUserID(),
 gs.getUserName());
}
```

6. If you are creating an event for a custom table, create a new business rule that runs after database operations.

   Example
   For example, this business rule defines several events for a custom application called Marketing Events.

Sample event business rule

Field	Value
Name	Attendee Events
Table	Attendee [x_snc_marketing_ev_attendee]
Application	Marketing Events
Advanced	Selected
When	after
Insert	Selected
Update	Selected
Delete	Selected
Script	Add custom script that: ○ Checks for one or more conditions on the current record. ○ Calls the gs.eventQueue() method and specifies an event name. See code sample.

Note:
If you add **Filter Conditions**, **Role conditions**, or a **Condition** value, verify it runs the business rule when expected.

```
(function executeRule(current, previous /*null when async*/) {
   //This function will be automatically called when this rule
 is processed.
        //Add event when attendee inserted
        if(current.operation() == 'insert' &&
 current.marketing_event.changes()) {

 gs.eventQueue('x_snc_marketing_ev.attendee.added', current,
current.marketing_event, current.email);
        }
        //Add event when marketing event changes
        if(current.operation() == 'update' &&
 current.marketing_event.changes()) {

 gs.eventQueue('x_snc_marketing_ev.attendee.deleted', previous,
previous.marketing_event, previous.email);

 gs.eventQueue('x_snc_marketing_ev.attendee.added', current,
current.marketing_event, current.email);
        }
        //Add event when attendee deleted
        if(current.operation() == 'delete') {

 gs.eventQueue('x_snc_marketing_ev.attendee.deleted', current,
              current.marketing_event, current.email);
}
})(current, previous);
```

7. Register the event.

What to do next
Create a script action or notification to process the event.

Related topics
- Register an event
- Script actions

Move an event

Accelerate the execution process by moving high priority events from the default queue to an isolated queue with just one click.

Before you begin
Role required: admin

Procedure
1. Navigate to **All** > **Event registry**.
   A list of registered events shows up.
2. Select the event you want to move.
   The Event Registration form of the selected event shows up.

3. Select **Move to Default Queue** to move the event from the current queue and accelerate its execution.

   Note:
   If you are currently in the default queue, you will have the **Move to Adaptive Event Queue** option to select.

   The processor can execute only one event at a time. So, if you move an event to a queue, it gets executed only when the processor is available.

4. **Optional:** Navigate to **System Diagnostics** > **Stats** > **Adaptive Events** to track the average execution time of the events.
   You can select any slow moving event and change the queue as required. You can also register an event to show up on the event registry list.

Pass event parameters from a workflow to a notification

Pass two event parameters that send information about a record or related records from a workflow to a notification.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **System Policy** > **Events** > **Registry** and define a new event to call.
2. Create the activity step that calls the event from your workflow and assign the two possible parameters.
   These parameters can be references or fields on the record that triggered the workflow, such as *current.number* for the request item number. You can also dot-walk to records in other tables. The event then sends the parameters information to the notification that it triggers.
3. Retrieve the parameters in the notification with email scripts by using *event.parm1* and *event.parm2*.
   For example:

```
var string = event.parm1.toString();
template.print(string);
```

Example:

Another example using Sys IDs gives you access to anything in the record that triggered the workflow. Use a script like this to send information about a change or request to support personnel, for example.

```
var id = event.parm1.toString();
var gr = new GlideRecord('sc_req_item');
gr.get('sys_id', id);
if (gr.next()){
   // Do something.
}
```

Related topics

Scripting for email notifications

Register an event

You can register an event for a specific table and a business rule that fires the event.

Before you begin

Role required: admin

Procedure

1. To register an event, browse to **System Policy** > **Events** > **Registry**, and then click **New**.
2. On the form fill in the fields.

Event Registration Form Completion

Field	Description
Event name	Name of your new event.
Table	Database table for this event. **Note:** Only tables and database views that are in the same scope as the event appear on the listing.
Queue	Name of the queue that the event is placed into when triggered. Use only lowercase letters, no spaces, and no special characters except underscore (_). For example, `my_queue`. See Using custom queues to process events.
Priority	Order in which messages will be processed. Lower values have higher priority. **Note:** The com.glide.sysevent.priority.enabled system property is enabled by default. If you disable it, events processing is not done by priority.
Caller Access	Restricted caller access settings. ○ Caller Restriction: calls to the resource must be manually approved. ○ Caller Tracking: Calls to the resource are automatically approved.
Fired by	Name of the business rule that runs the event. This field is for reference only and is not used by any process. Make sure that there is enough information to locate your event again.
Description	Short description of the purpose of the event.

Reprocess an event

You can fire the event again for testing or diagnostic purposes.

Before you begin

Role required: admin

Procedure

1. Navigate to **All** > **System Logs** > **Events**.
2. Open an event.
3. Under **Related Links**, click **Reprocess Event**.
   The event returns to the event queue.

System events reference

Reference topics provide information about system events.

Event logs

The event log records all system events that occur within the ServiceNow AI Platform.

This log provides the following information for all events that occur:

Event log

Field	Description
Created	Date and time of the event for the locale of the machine running the instance.
Name	Name of the event as listed in the Event Registry.
URI	HTTP query that generated the event.
Parm1	Event-specific value that depends on the event and the recipient.
Parm2	Event-specific value that depends on the event and the recipient.
Table	Database table acted on for this event.
Processed	Date and time the event started processing. This time reflects the locale of the machine running the instance.
Processing time	Time taken to process this event, in milliseconds.
Queue	Processor queue name.

Event states

The event state describes where in the life cycle the event is.

Event states

State	Description
Ready	The system created the event and it is in the queue waiting to be processed.
Processed	The event successfully ran. An event does not necessarily trigger any further action when processed. Additional functionality must make use of the event.
Error	The event encountered an error during processing. This state is often caused by invalid event parameters. Reprocessing the event may resolve the error.

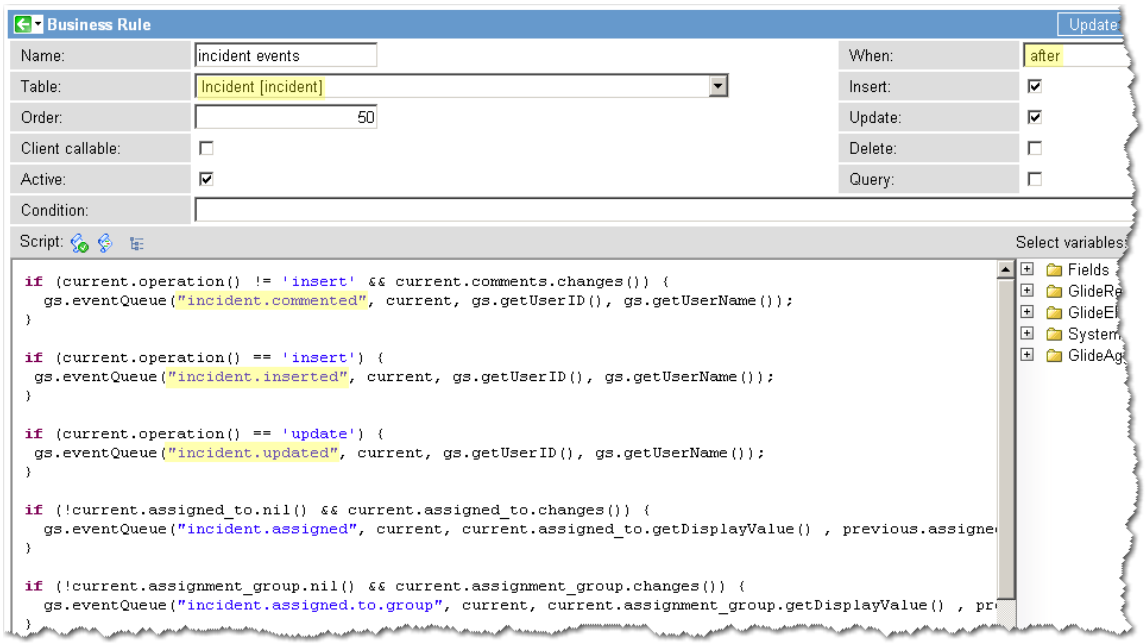
Event states (continued)

State	Description
Transferred	The event was rotated to a different shard of the Event [sysevent] table. When an event is rotated, a duplicate record is created in an active shard to be processed. A scheduled job processes the event when it is next in the queue, but it is not possible to predict when this will happen as because several events may need to be processed before it. Therefore, you can reprocess the event. See Reprocess an event.

The incident events business rule

The incident events business rule comes with the system and defines a number of events that can be triggered by different actions in the Incident table.

Incident events business rule



This business rule defines several events, three of which are triggered after a record in the Incident table is inserted or updated. The first script is:

```
if (current.operation() != 'insert' &&
 current.comments.changes()) {
gs.eventQueue("incident.commented", current, gs.getUserID(),
 gs.getUserName());
}
```

The condition in this script requires that a change be made to the **Comments** field in an existing (not inserted) incident record. If this condition is true, then the platform adds the `incident.commented` event to the event queue.

The second condition requires that a record be inserted before the event is added to the queue.

```
if (current.operation() =='insert') {
```

The third condition is true whenever the incident record is updated (including updates to the **Comments** field, as specified by the first script).

```
if (current.operation() == 'update')
```

The then part of each script, the *gs.eventQueue* function, adds the event to the event queue. This statement uses the following syntax, set off with braces:

```
gs.eventQueue("incident.updated", current, gs.getUserID(),
 gs.getUserName());
```

The *gs.eventQueue* function takes the following parameters:

Field	Input Value
Name	The name of the event triggered, set in quotation marks
Record	The record referenced when the condition in the script evaluates to *true*. Usually this is expressed as *current*, meaning the current record the business rule is working on. If the business rule is being triggered as part of a scheduled job, use a GlideRecord argument in its place.
Parm 1	An optional parameter you can use to pass system or record information with the event. For example, the GlideSystem API call *gs.getUserID()* passes the Sys ID of the user who acted on the current record as a string value. Other scripts can reference this string value as parm1 using the format `${event.parm1}`.
Parm 2	An optional parameter you can use to pass system or record information with the event. For example, the GlideSystem API call *gs.getUserName()* passes the user name of the user who acted on the current record. Other scripts can reference this string values as parm2 using the format `${event.parm2}`.

Note:
The gs.EventQueue function works directly with the backend and therefore business rules that are called by gs.EventQueue() are not invoked.

Global events

Your instance has a global function called *global_events()* that triggers from a business rule when certain conditions occur.

This function triggers when your instance is:

- Inserting new records
- Updating existing records
- Adding comments to an existing record
- Assigning a record to a user
- Exceeding a record's inactive timer

For example, if you add the script global.events(current) to a business rule on the `change_request` table, the instance automatically configures the following events:

- change_request.inserted
- change_request.updated
- change_request.commented
- change_ request.assigned
- change_ request.inactive

Script actions

You can use script actions to create server-side scripts that perform a variety of tasks, such as modifying a configuration item (CI), or managing failed login attempts. Script actions are triggered by events only.

Configuration

To create a new script action, navigate to **System Policy** > **Events** > **Script Actions** and click **New**.

Script actions

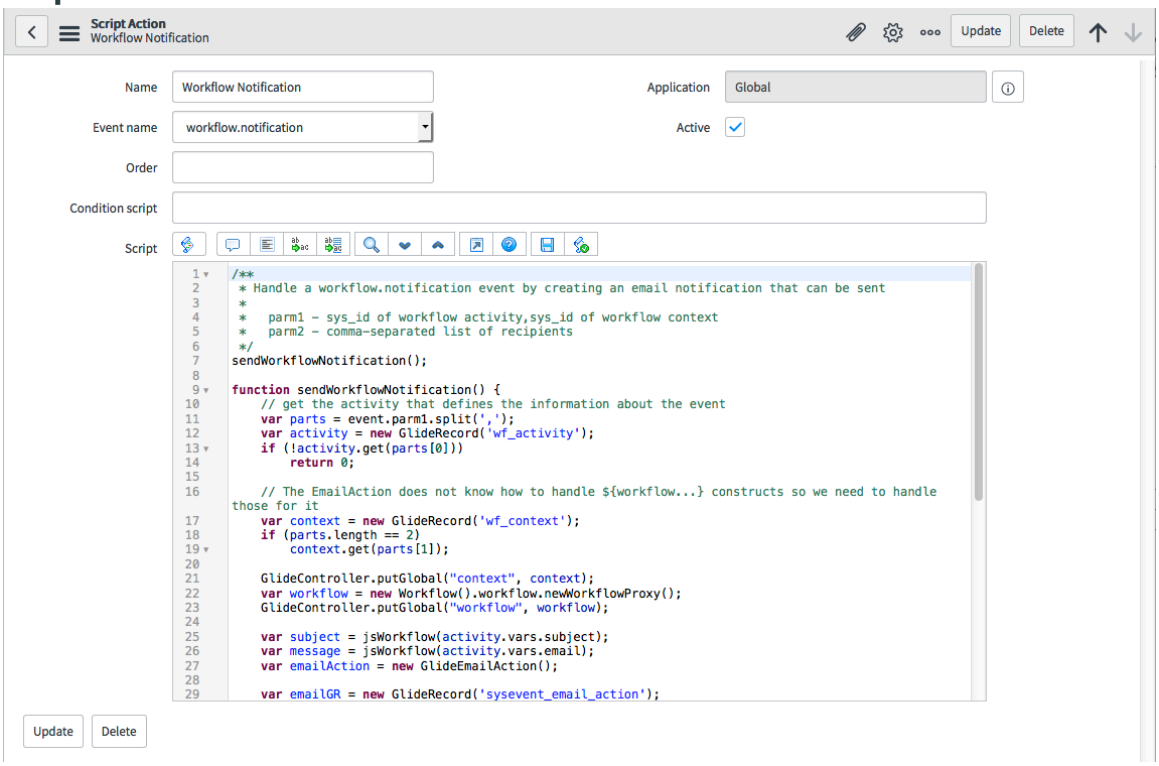
Field	Description
Name	Unique name for your script action.
Application	Application that contains the script.
Event name	Event to use for this script. If you do not find an event for your script action that suits your purpose, you can create a new one.
Active	Check box that enables or disables the script action. Select **true** to enable the script action.
Execution order	Order in which the script will be executed.
Condition script	Statement for a condition under which this script should execute. The system only parses the **script** field if the condition evaluates to true. If you decide to include the condition statement in the script, leave this field blank.
Script	Script that runs when the condition you define evaluates to true. Two additional objects are available in this script: • *event*: a GlideRecord - the sysevent that caused this to be invoked. If you want this first parameter on the event, use *event.parm1* or *event.parm2* for the second parameter. For the date/time of the event, use

Script actions (continued)

Field	Description
	event.sys_created_on. To get the user ID that created the event (if there was a user associated), use *event.user_id*. • *current*: a *GlideRecord* - the event scheduled on behalf of (incident for example).

Sample of a script action that creates an email notification for Workflow activity:

Script action workflow



Sample scripts from the change events business rule

Several scripts are found in the baseline change events business rule.

This business rule defines events that fire after a change request is inserted or updated.

```
if (current.operation() == 'insert') {
 gs.eventQueue("change.inserted", current, gs.getUserID(),
 gs.getUserName());
}

if (current.operation() == 'update') {
 gs.eventQueue("change.updated", current, gs.getUserID(),
 gs.getUserName());
}

if (!current.assigned_to.nil() && current.assigned_to.changes())
 {
  gs.eventQueue("change.assigned", current,
 current.assigned_to.getDisplayValue() ,
 previous.assigned_to.getDisplayValue());
```

```
}

if (current.priority.changes() && current.priority == 1) {
  gs.eventQueue("change.priority.1", current, current.priority,
 previous.priority);
}

if (current.risk.changes() && current.risk== 1) {
  gs.eventQueue("change.risk.1", current, current.risk,
 previous.risk);
}

if (current.start_date.changes() || current.end_date.changes()
 || current.assigned_to.changes()) {
  if (!current.start_date.nil() && !current.end_date.nil()
 && !current.assigned_to.nil()) {
    gs.eventQueue("change.calendar.notify", current,
 current.assigned_to, previous.assigned_to);
  }

  // Remove from previous assigned to, due to assigned_to
 changing
  if (!previous.assigned_to.nil()) {
      if (!current.assigned_to.nil() &&
 current.assigned_to.changes() &&
         (!previous.start_date.nil()
 && !previous.end_date.nil())) {
        gs.eventQueue("change.calendar.notify.remove", current,
 current.assigned_to, previous.assigned_to);
      }
   }
  // Remove old calendar from current assigned to, due to date
 changing
  else if (!current.assigned_to.nil()) {
     if ((current.start_date.changes()
 && !previous.start_date.nil()) ||
         (current.end_date.changes()
 && !previous.end_date.nil())) {
       gs.eventQueue("change.calendar.notify.remove", current,
 current.assigned_to, current.assigned_to);
     }
  }
}
```

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.
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.

Property	Description
	• 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.
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.

Business rule	Description
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.
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 Creator email notifications (continued)

Name	Description
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.

Email notifications created with new service categories (continued)

Name	Description
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.
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.

Roles (continued)

Role Title [Name]	Description
	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.
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.

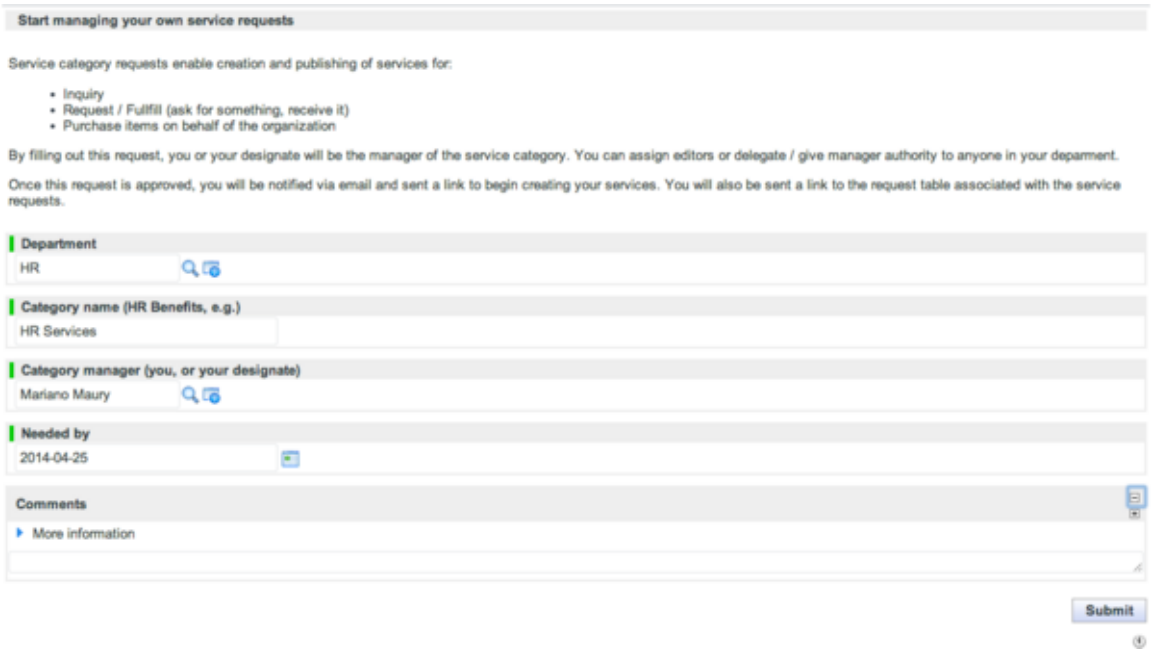
A service category request involves assigning a service category manager, which is typically the department manager who makes the request. After the request is submitted, a catalog administrator approves the request to publish the service category. When the category is published, the service category 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.

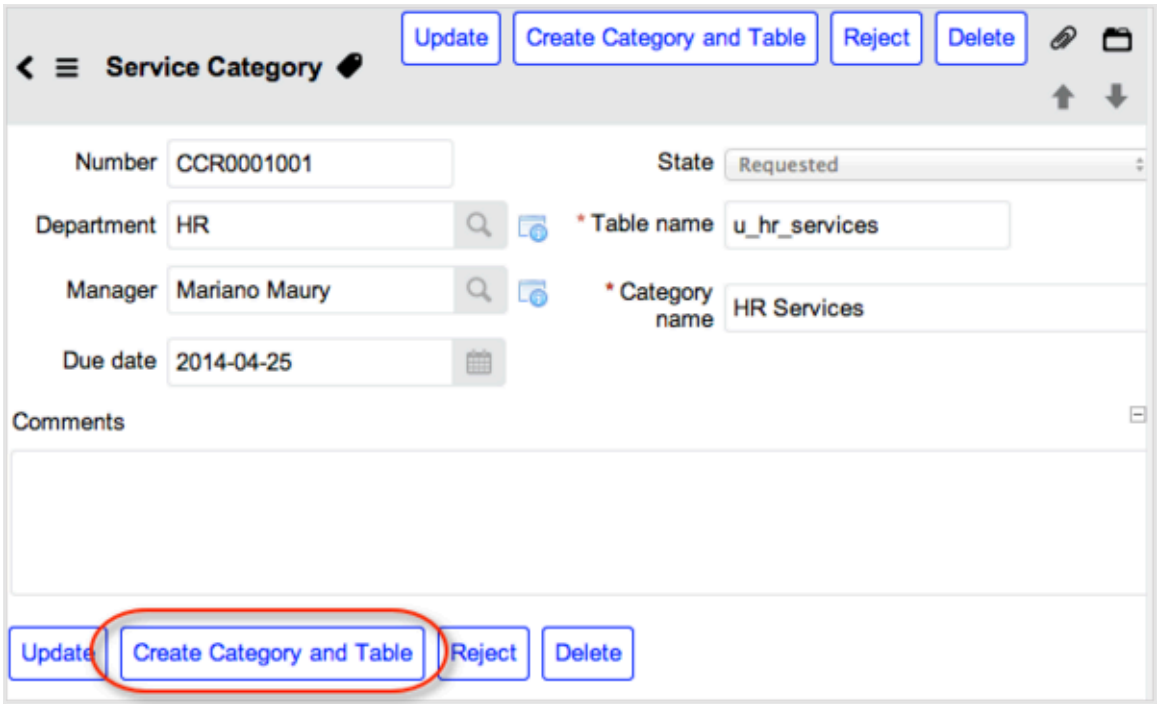
About this task

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.

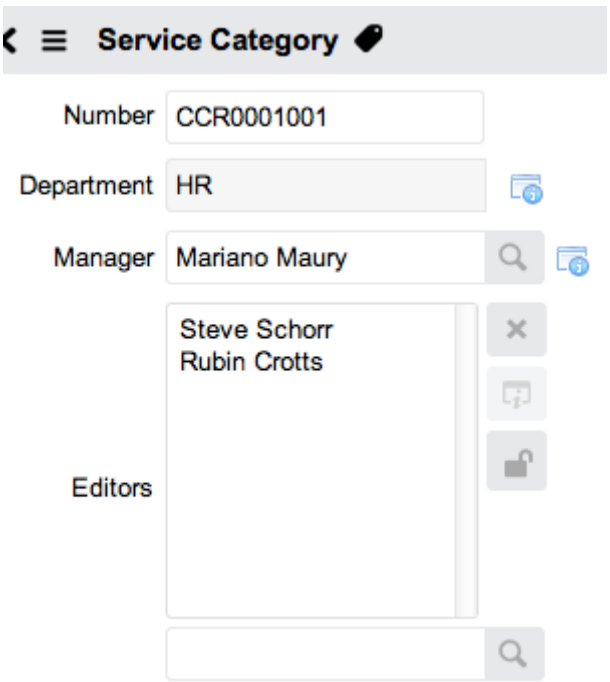
Procedure

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.

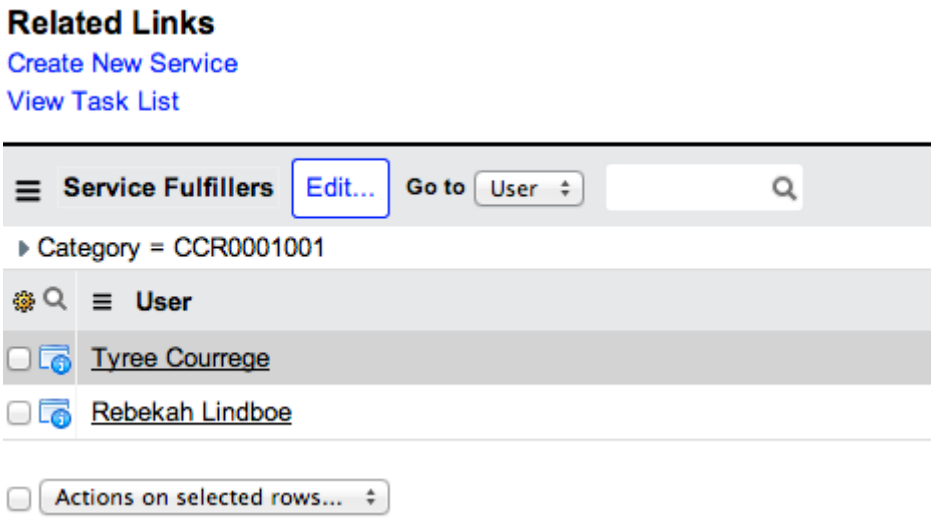
The *Service Fulfillers* related list on the Service Category form displays all users assigned as fulfillers for that service category. The service category manager can designate service fulfillers for a service category.

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**.

Classic Workflow

Workflow provides a drag-and-drop interface for automating multi-step processes across the platform. Each workflow consists of a sequence of activities, such as generating records, notifying users of pending approvals, or running scripts. The graphical Workflow Editor represents workflows visually as a type of flowchart. It shows activities as boxes labeled with information about that activity and transitions from one activity to the next as lines connecting the boxes.

Note:
As of the New York release, Workflow is a legacy product. Use Flow Designer for any new process automation. As many organizations have workflows in production, use this Workflow documentation to learn how to work with existing workflows.

Explore

- Xanadu
- Workflow movement with update sets
- Workflow activity pinning
- Domain separation and Workflow
- Workflow training on the ServiceNow® Developer Site

Set up

Getting started with workflows

Administer

- Workflow roles
- Administering workflow contexts

Use

- Workflow editor
- Create a workflow
- Workflow activities

Develop

- Developer training
- Developer documentation
- Workflow API reference
- Using variables in a workflow

Troubleshoot and get help

- Ask or answer questions in the Developer Community
- Troubleshoot workflows
- Search the Known Error Portal for known error articles
- Contact Customer Service and Support

Getting started with workflows

The graphical Workflow Editor provides a drag-and-drop interface for automating multi-step processes across the platform.

Parts of a workflow

Workflows consist of these parts.

Properties

Specify configuration settings such as the workflow name, the table whose records the workflow acts on, and the conditions under which to run it.

Activities

Specify the sequence of operations the workflow performs such as generating records, notifying users of pending approvals, or running scripts.

Transitions

Specify the conditions under which to run an activity.

Exit conditions

Specify the conditions under which to run a transition.

Contexts

Store historical runtime information about a specific workflow run in a Workflow Context record.

Versions

Store historical design information about a specific workflow in a Workflow version record.

Workflow life cycle

A workflow starts when a triggering event occurs. Common triggers include a record being inserted into a specific table, or a particular field in a table being set to a specified value. For example, you might create a workflow that runs whenever a user requests approval for an item they want to order from the catalog. You can also schedule workflows to run periodically or call them from scripts such as business rules.

When an activity completes, the workflow transitions to the next activity. An activity might have several different possible transitions to various activities, depending on the outcome of the activity. Continuing the example above, if the user's request is approved, the activity might transition to an activity that notifies someone to order the item. If the user's request is denied, the activity might transition to notifying the user that their request has been denied.

The graphical Workflow Editor represents workflows visually as a type of flowchart. It shows activities as boxes labelled with information about that activity and transitions from one activity to the next as lines connecting the boxes.

At each step in a workflow:

1. An activity is processed and an action defined by that activity occurs.
2. At the completion of an action by an activity, the workflow checks the activity's conditions.
3. For each matching condition, the workflow follows the transition to the next activity.

When the workflow runs out of activities, the workflow is complete. The Workflow Context stores the execution history of the activities and transitions run. The Workflow Version stores the design history of the activities, transitions, and exit conditions available to run.

Workflow properties

The workflow properties specify when to run a workflow and what records it acts on. For more information about workflow properties, see Workflow properties.

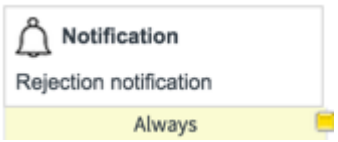
Workflow activities

A workflow activity contains instructions that are processed by the workflow.

Activities can include running scripts, manipulating records, waiting for a set period of time, or logging an event. Workflow conditions determine whether or not the activity is performed. Activities can be added, removed, or rearranged. Transitions can be drawn between activities.

This is an activity that triggers a notification:

Sample activity



Workflow runs activities as the user session that starts them. Workflows started from record operations will run activities as the user session that performed the record operation. Workflows started from schedules or restarted from timers run activities as the System user. Workflows started from script calls run activities as the user session that started the script.

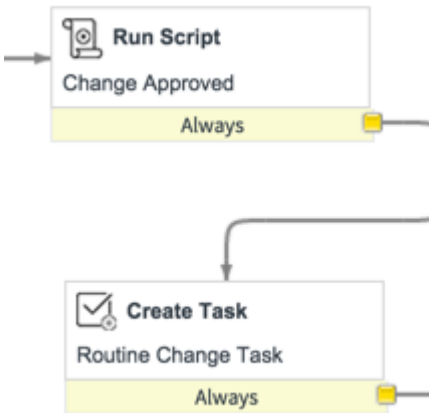
For more information on available activities and their behaviors, see Workflow activities.

Transitions

After the workflow condition is evaluated, the workflow transition determines which activity is performed when the workflow condition is met.

This is a transition that always leads from the *Change Approved* script to the **Change Task** activity:

Sample transition



Exit conditions

After a workflow activity is performed, the workflow condition is evaluated to determine which transition is activated.

The condition determines behavior based on a change being approved or rejected:

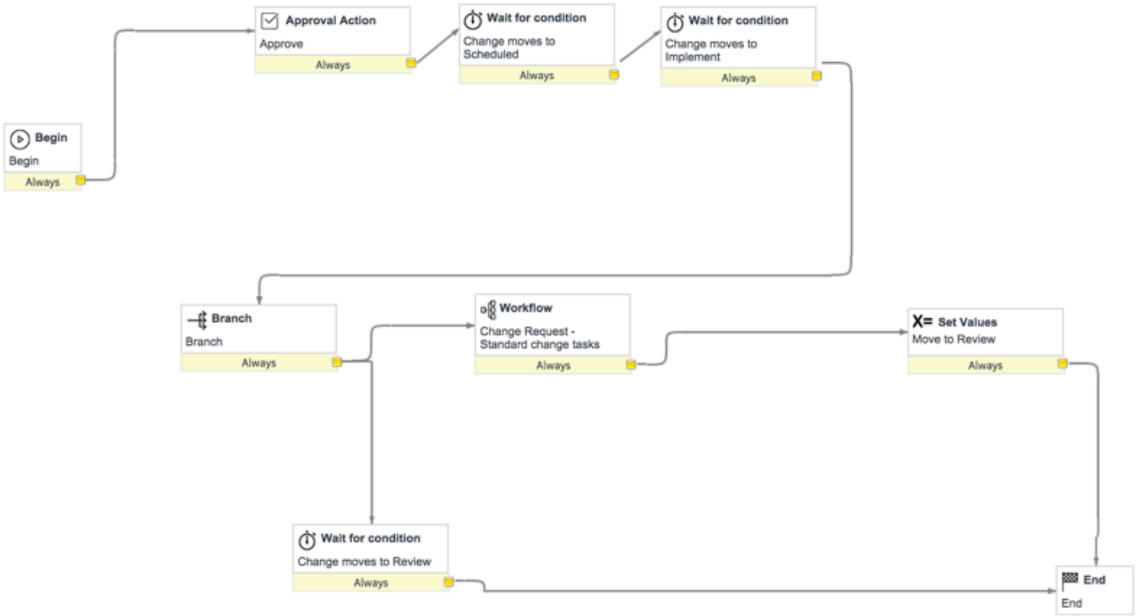
Sample exit conditions

Workflow example

During workflow editing or while an unpublished workflow is running, only the person who checked out the workflow can view the changes.

After a workflow is published, it is available to other users. The workflow moves through the process as defined in the Workflow Editor. The entire workflow is represented in one screen. For example, this is the Standard Change workflow:

Sample change workflow



Workflow editor

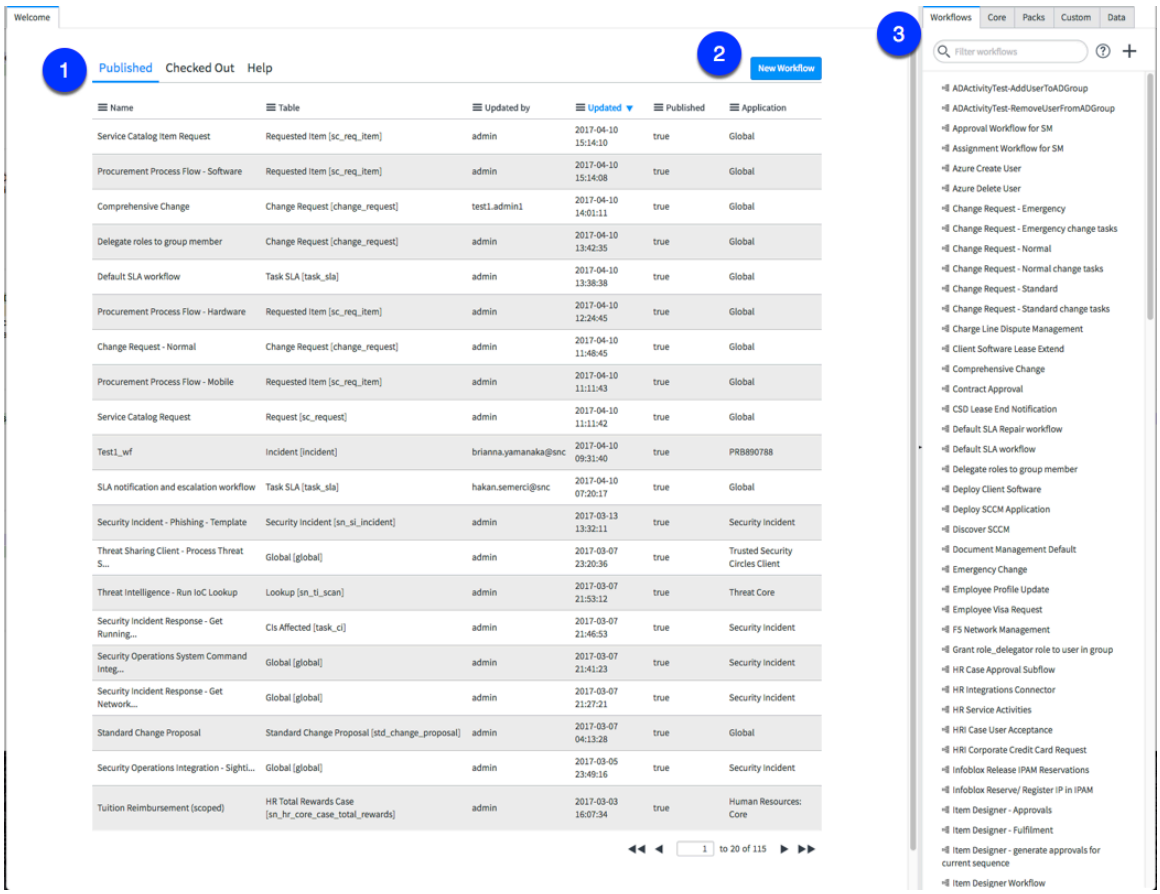
The Workflow Editor is an interface for creating and modifying workflows by arranging and connecting activities to drive processes.

You can manage multiple workflows in the same screen, create custom workflow activities, and use existing activities as data sources. Users with the workflow_creator role can create workflows. Users with the workflow_admin role can create, modify, delete, and publish workflows.

To open the Workflow Editor, navigate to **Workflow** > **Workflow Editor**. For information about using the editor, see Create a workflow.

Welcome screen

The editor opens with the **Welcome** page, which displays a list of active, published workflows. From this tab, you can open existing workflows, create new workflows, and open help resources related to workflow.



1 List display filters

Published: Click to view list of published workflows

Checked Out: Click to view list of workflows checked out to current user

Help: Click to view links to help resources for workflow

2 New Workflow button

Click to create a new workflow

3 Palette tabs

Workflows: Workflow activities and workflows you can use as subflows.

Core: Available workflow activities appropriate for the selected workflow. Contents can include activities provided by the base system and those purchased with orchestration.

Packs: Orchestration activity packs downloaded from the ServiceNow® App Store, organized by vendor and scope. Custom activities and workflows that you scope also appear as packs. Only visible if the orchestration plugin is installed.

Custom: Orchestration custom activities available. Only visible if the orchestration plugin is installed.

Data: Activities in the current workflow that output data. You can use these activities as data sources for other activities. Only visible if the orchestration plugin is installed.

Note:
If your workflow welcome page does not look like this example, you may have customized the workflow welcome page before upgrading to Xanadu. You can update the workflow welcome page to the latest version by editing the UI pages .

Navigate to **System UI** > **UI Pages** > **Workflow Editor welcome**. In the UI page record for workflow_editor_welcome, scroll to the **Versions** related list. Select the row for the version corresponding to the upgrade to Xanadu, right-click and select **Revert to this version**.

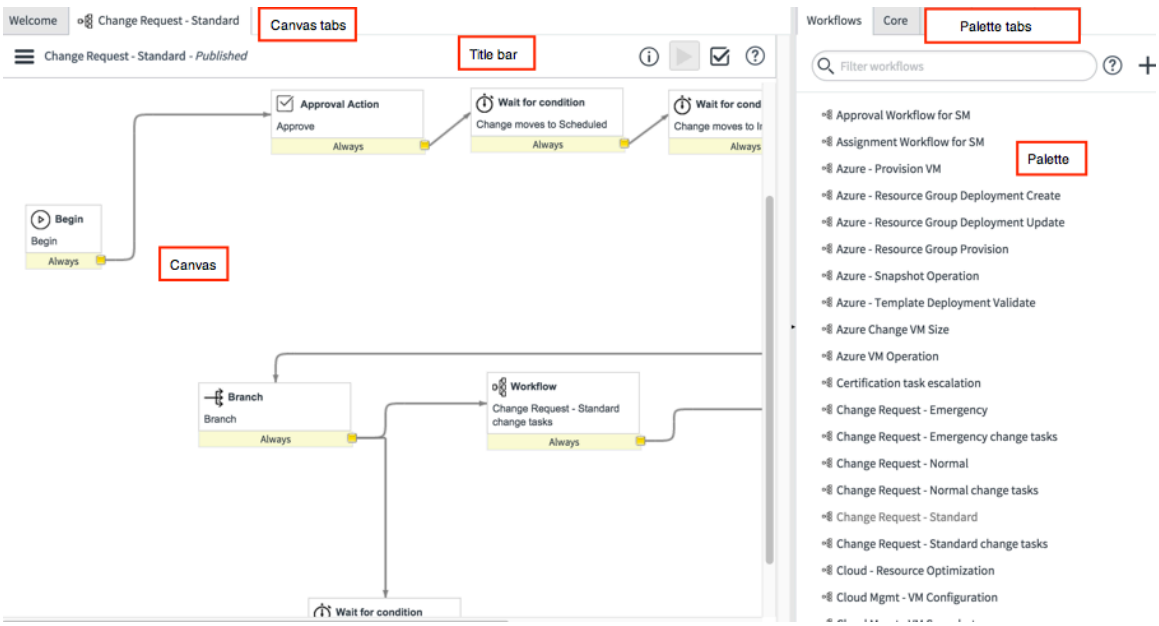
Workflow canvas

After you open or create a new workflow, the system displays the workflow canvas. On the canvas you interact with the Workflow Editor through several different elements: the canvas itself, the canvas tabs, the title bar, the palette, and the palette tabs.

The drawing canvas is where you add activities and configure transitions for checked out workflows. Add an activity by dragging it from the palette to the workflow in the canvas. For more information, see Create a workflow.

- **Canvas tabs**: Contains tabs for accessing workflows being edited or created.
- **Title bar:** Displays the workflow name and status. Provides a menu and controls for configuring, testing, and validating workflows.
- **Canvas**: Provides the working surface for creating new workflows or editing existing ones.
- **Palette tabs**: Contains tabs for accessing activities being edited or created.
- **Palette**: Contains all available workflow activities and existing workflows you can use as subflows. Drag activities and subflows to the canvas to create new workflows or edit existing ones.

Workflow user interface



Workflow palette

The default workflow palette contains workflow activities and existing workflows you can use as subflows.

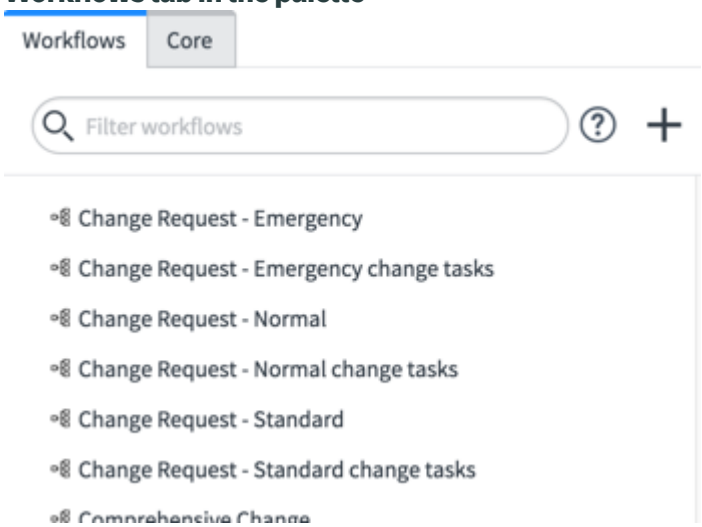
The basic workflow palette contains these tabs:

- **Workflows**: Displays existing workflows and provides controls for creating new ones.
- **Core**: Displays baseline workflow activities available to all systems and Orchestration activities (when Orchestration is activated).

Workflows tab

The **Workflows** tab lists existing workflows that you can edit or use as subflows in other workflows. Double-click a workflow to open it in the canvas. To add a workflow as a subflow, drag it to another workflow in the canvas. Click the **+** icon to create a new workflow.

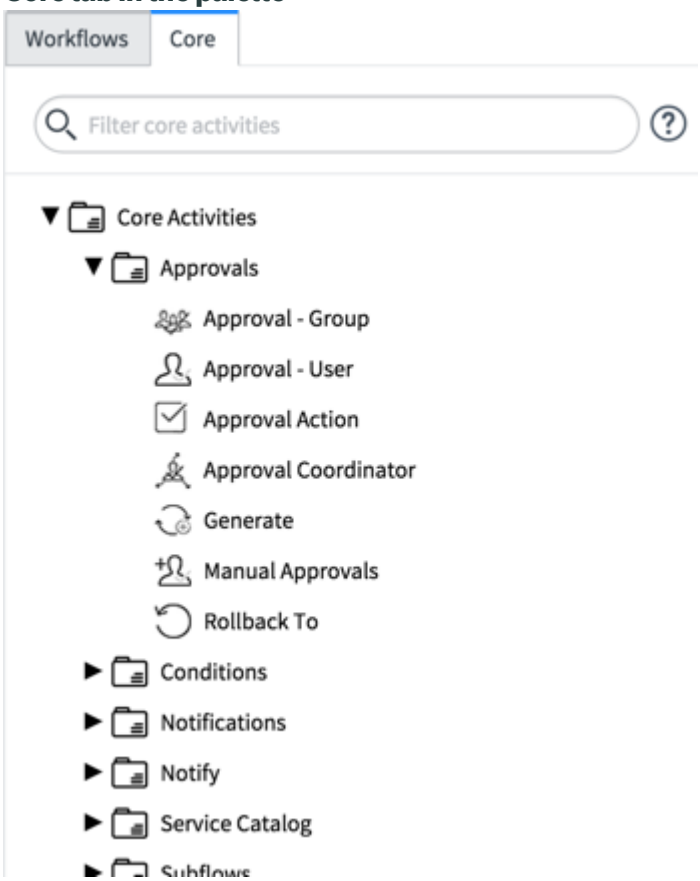
Workflows tab in the palette



Core tab

The **Core** tab contains the standard activities available by default to all workflows and any activities purchased with Orchestration, organized by category. Click the arrow icons to expand or collapse the activity lists under each category. To add an activity to a workflow, drag it to the canvas. For more information, see Add an activity to a workflow.

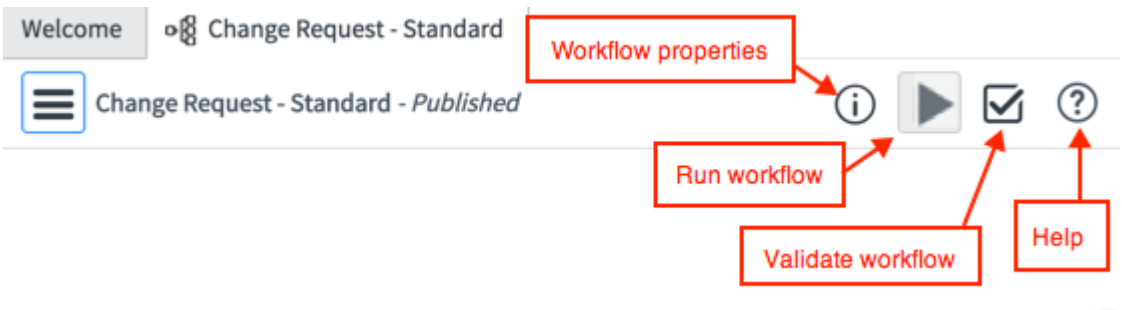
Core tab in the palette



Workflow editor title bar

When a workflow is opened in the canvas, the title bar displays the workflow title and the workflow status in italics. Possible states are **Checked out by <name>** and **Published**.

Workflow editor title bar



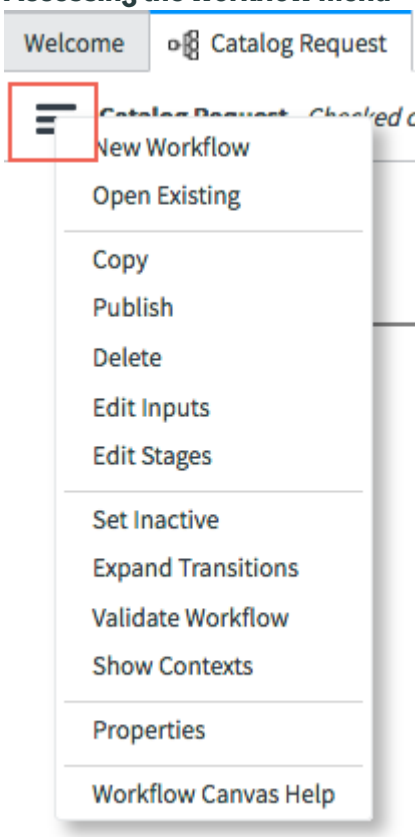
Controls on the right side of the title bar manage the workflow.

- **Workflow Properties** ⓘ: Opens the current workflow's properties form.
- **Start** ▶: Runs the workflow. This control is only available for workflows running on the Global table that are accessible from all application scopes. To test workflows that are on other tables, insert a record into that table that meets the condition of the workflow.
- **Validate** ☑: Tests the workflow prior to publication. Validation detects potential problems that can prevent the workflow from publishing or cause the workflow to fail. For more information, see Workflow Validation.
- **Help** ⓘ: Opens documentation to help you create the workflow.

Workflow menu

Click the menu icon in the title bar for additional options to configure the workflow.

Accessing the workflow menu



These menu options are available:

Workflow menu options

Option	Description
New Workflow	Creates a new workflow.
Open Existing	Opens another existing workflow.
Copy	Creates a duplicate of the workflow. Give the copy a different name.
Publish	Makes the personal workflow version public, overwriting the current published workflow version. This option is only available for checked out workflows.
Checkout	Creates a personal version of the workflow for you, which you can edit. This option is only available for published workflows.
Delete	Deletes the workflow. You cannot delete workflows that have contexts associated with them.
Set Inactive	Inactivates the workflow so that it cannot be used.
Expand Transitions	Redraws the transitions so that they do not overlap when they leave the activity condition.

Workflow menu options (continued)

Option	Description
Start Workflow	Starts a test run of the current workflow.
Validate Workflow	Runs validation tests on your workflow prior to publication. Use this validation to detect potential problems that can prevent the workflow from publishing or cause the workflow to fail. For more information, see Work on workflows.
Collapse Transitions	Redraws the transitions so they overlap when they leave the activity condition.
Show Contexts	Displays all the contexts for the current workflow. You can use this option to troubleshoot a workflow.
Properties	Opens the Workflow Properties form, which defines the workflow's attributes.
Edit Inputs	Opens the Workflow Inputs list of variables that the workflow can accept when used as a subflow. For more information, see Pass a variable from a workflow to a subflow.
Edit Stages	Opens the Workflow Stages list. For more information, see Workflow stages. For tables with a column of Type = Workflow.

Workflow editor keyboard navigation

The platform includes accessibility features that makes the interface accessible to users with disabilities.

These features improve the user experience when accessing platform functions with Using accessibility features. In general, you use the following set of standard keyboard navigation functions:

- Press **Tab** to navigate major groupings in a pre-defined sequence. This usually includes moving between standard interface controls (fields and lists) in a module, or between records within a tab.

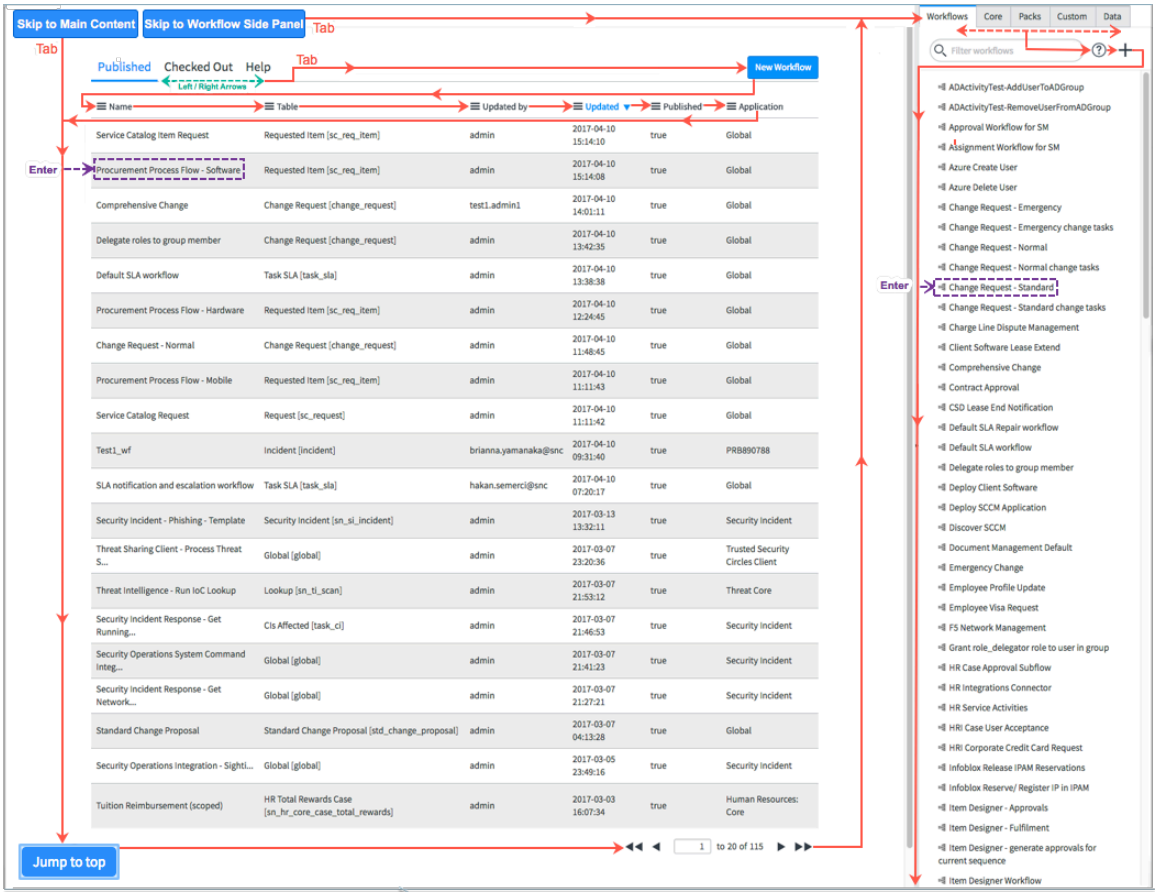
  Press **Shift Tab** to move backwards in a pre-defined sequence.

- Use arrow keys (left, right, up, down) to navigate between individual elements within a group. This usually includes moving between tabs, or between available selections within a specific control (for example, within a list).
- Press **Enter** to select a control or tab, or enter text within a control.

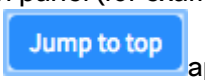
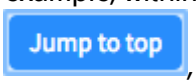
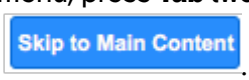
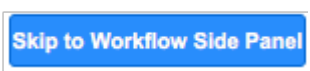
The Workflow Editor is constructed in a unique manner. It includes a series of main (left) panel tabs, a series of side (right) panel tabs, and a drawing canvas for workflow creation or editing. As such, it has its own unique set of keyboard accessibility functions and commands.

Welcome page keyboard commands

Use keyboard commands to navigate and operate the Workflow Editor Welcome page.



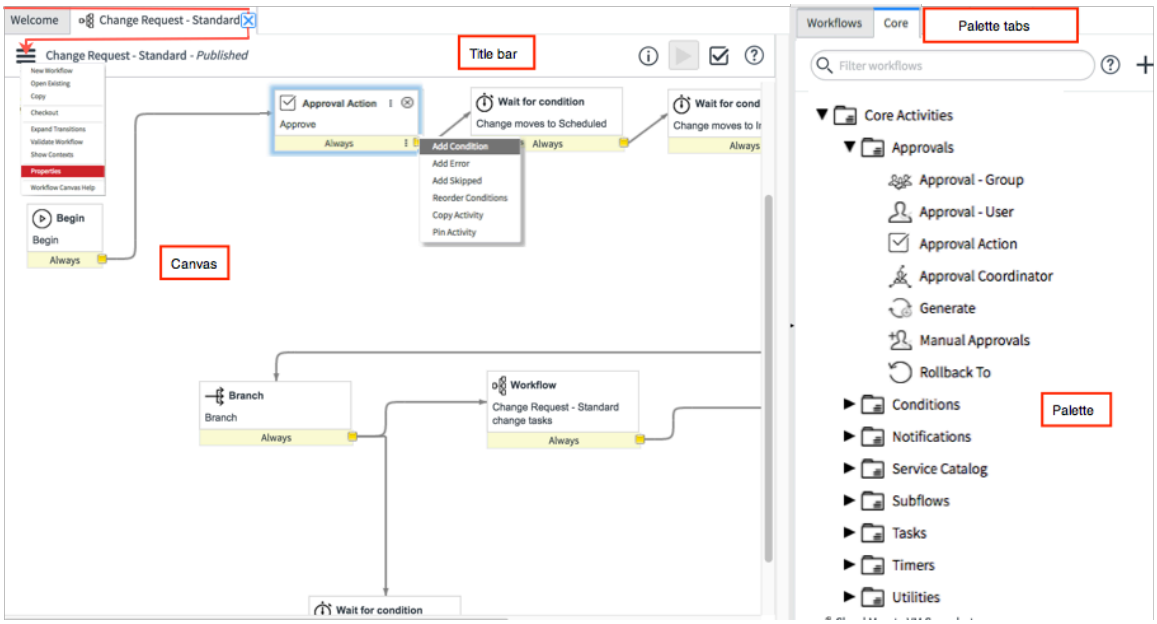
Task or Action	Keyboard Commands
Navigate to main content (left) panel, and select a workflow	**1.** After accessing the Workflow Editor from a menu, press **Tab**. Skip to Main Content appears in the upper left corner. **2.** Press **Enter** to position the cursor in the first record displayed in the **Published** tab. **3.** Press **Tab** to navigate down the listing of workflow records. Press **Enter** to select and open a workflow record in the Workflow drawing canvas.
Select a checked out workflow or a help function in main content panel	**1.** After accessing the Workflow Editor from a menu, press **Tab** until the **Published** tab is highlighted. **Note:** Do not select Skip to Main Content or Skip to Workflow Side Panel.

Task or Action	Keyboard Commands
	2. Press the right or left arrow keys to navigate between the **Published**, **Checked Out**, or **Help** tabs. **3.** Press **Tab** to navigate across the columns and down the listing of checked out workflow records or help selections. Press **Enter** to select and open a checked out workflow record in the Workflow drawing canvas, or open a help selection (work or orchestration videos, documentation, community discussions, knowledge base, or Live Feed).
Create a workflow	**1.** After accessing the Workflow Editor from a menu, press **Tab** until  is highlighted. **2.** Press **Enter** to open New Workflow
Jump to Top	After tabbing through the displayed records in a tab in the main panel (for example, within the **Published** tab),  appears at the bottom of the listing. Press **Enter** to return to the Welcome tab at the top of the Workflow Editor.
Navigate to page selection controls 	**1.** After tabbing through the displayed records in a tab in the main panel (for example, within the **Published** tab), including  , press **Tab** to access the page selection controls. Continue to press **Tab** until you access the desired one. **2.** Press **Enter** to operate the selected page control.
Navigate directly to side (right) panel to open help, create a new workflow, or open an existing one	**1.** After accessing the Workflow Editor from a menu, press **Tab** two times. Do not select   appears in the upper left corner. **2.** Press **Enter** to position the cursor in the **Workflows** tab in the side panel. **3.** To open help, press **Tab** to navigate to  . Press **Enter** to open help, or press **Tab** to skip.

Task or Action	Keyboard Commands
	4. To create a new workflow, press **Tab** to navigate to +. Press **Enter** to open New Workflow, or press **Tab** to skip. **5.** To open an existing workflow: ○ As needed, stop at the filter field to filter workflows. ○ Press **Tab** to navigate down the workflow list. ○ Press **Enter** to select and open a workflow record in the Workflow drawing canvas,
Add a core activity to a checked out workflow in the Workflow drawing canvas	If a checked out workflow is open in the Workflow drawing canvas, core activity records display in the **Core** tab. To select a core activity and place it in the checked out workflow: **Note:** Core activity records only display when a checked out workflow is open in the Workflow drawing canvas. Other tabs display in the side panel only if Orchestration functions are enabled. **1.** In the Workflow drawing canvas, press **Shift Tab** until Skip to Workflow Side Panel appears in the upper left corner. **2.** Press **Enter** to position the cursor in the **Workflows** tab in the side panel. **3.** Press the right arrow key to navigate to the **Core** tab. **4.** Press **Tab** or the down arrow key to navigate down the listing of core activities. ○ To open a core activities folder (for example, Approvals), press the right arrow key. ○ To navigate up the list, press the left or up arrow keys. **5.** Press **Tab** to select a core activity and open the New Activity dialog to specify properties for it. **6.** After creating the activity, remember to link it to another activity in the workflow. See Create a connection from a condition on one activity to the next activity that follows in Workflow drawing canvas keyboard commands.

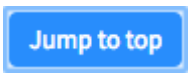
Workflow drawing canvas keyboard commands

Use keyboard commands to navigate and operate the Workflow Editor canvas.



Task or Action	Keyboard Commands
Select Workflow Actions menu command (for example, **Validate Workflow** or **Publish Workflow**)	**1.** Press **Tab** until the context menu (☰) is highlighted. **2.** Press **Enter** to open the context menu. **3.** Use the down arrow key to move to the command, then press **Enter** to select it.
Set general workflow properties	**1.** Press **Tab** until ⓘ is highlighted. **2.** Press **Enter** to open Workflow Properties.
Navigate from activity box to activity box.	Simply press **Tab** or use the arrow keys (right, left, up and down) to navigate from activity to activity within the workflow. **Note:** Depending on the complexity and number of activity boxes in the workflow, the sequence of navigation is not always predictable. Depending on how you are navigating within the workflow, the navigation sequence is generally (but not always) row-by-row, and focus is usually placed on the nearest activity box.
Modify a selected activity.	**1.** Navigate to the activity box, then press **Enter** to select it and place it in Edit mode. When you select an activity box, it appears as highlighted. **2.** Once in Edit mode, use **Tab** to move around within the activity box to change or access elements such as: ○ Activity Properties (☑) ○ Title ○ Node context menu ○ Delete Node (⊗)

Task or Action	Keyboard Commands
	○ Condition Properties (Always) ○ Node conditional context menu ( ) ○ Node conditional port options ( ) **Note:** All elements in a selected activity are only accessible when working with a checked out workflow. When working with a published workflow, you can only access its activity properties and title, **3.** Press **Enter** to select an element, or press **Esc** to escape an element without making changes.
Add a condition to an activity.	**1.** Within an activity box, select to access **Condition Properties**. **2.** In **Condition Properties**, specify the conditions for the activity.
Create a connection from a condition on one activity to the next activity that follows	**1.** Within an activity box, select **Node conditional context menu** ( ). **2.** Select **Link to...** to create a connection to the next activity box that follows, or select **Delete** to delete an existing connection.
Add a core or custom activity	**1.** Within an activity box, select **Node conditional port options** ( ). **2.** Select **Add Core Activity** to access Workflow Activity Definitions to add a new core activity, or select **Add Custom Activity** to add a custom activity to the workflow.
Move an activity box	**1.** Press **Tab** to navigate to the activity box, enter press **Enter** to select it. **2.** Use the arrow keys to move the activity box. To move the activity box one pixel at a time, press **Shift** while using the arrow keys.
Validate a workflow	**1.** Press **Tab** until is highlighted. **2.** Press **Enter** to validate the workflow. You can also select **Validate** from the Workflow Actions menu.
Run a workflow	**1.** Press **Tab** until is highlighted. **2.** Press **Enter** to run the workflow. **Note:** If the workflow is tied to a database table, this function is disabled. The workflow runs when the proper table conditions are activated (for example, insertion of a new record into the table).

Task or Action	Keyboard Commands
Close workflow drawing canvas	1. Press **Tab** until  (on the right side of the tab that contains the name of the workflow) appears. 2. Press **Enter** to close the canvas.
Publish a workflow	1. Press **Tab** until the context menu () is highlighted. 2. Press **Enter** to open the context menu. 3. Use the down arrow key to move to the **Publish** command, then press **Enter** to select it.
Jump to Top	After tabbing through the entire workflow,  appears at the bottom of the listing. Press **Enter** to jump to the top of the Workflow drawing canvas.

Workflow management

Create, edit, validate, and publish workflows to automate multi-step processes across the platform. Understand workflow activities and variables and how to use them effectively. Take a deeper look at how workflows are constructed, validated, and used.

Create a workflow

Automate a multi-step process by creating a workflow with the Workflow Editor.

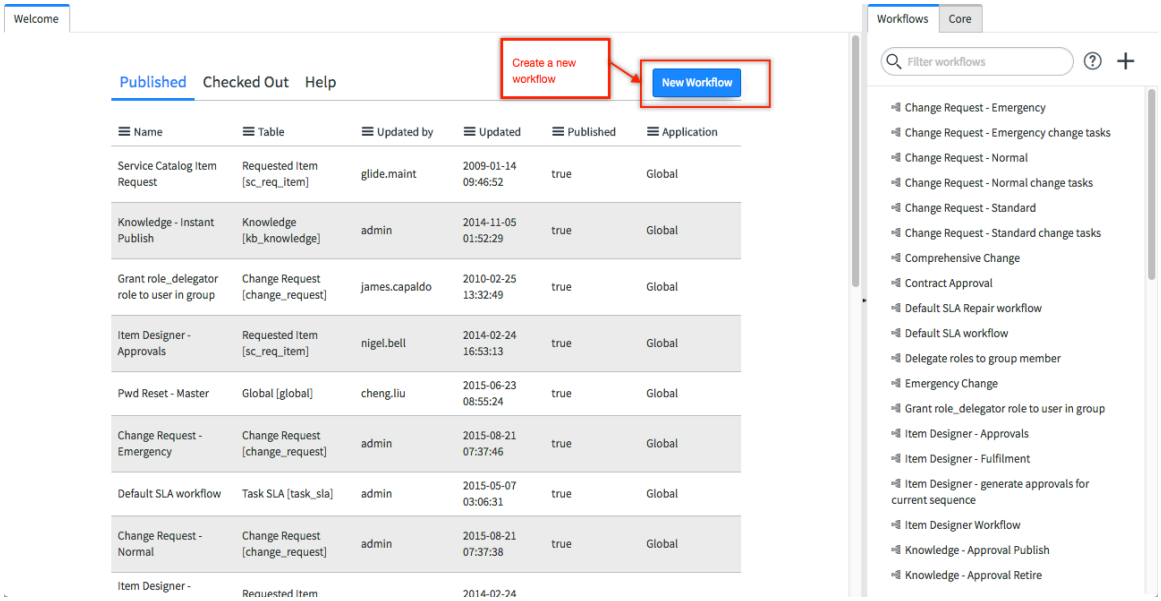
Before you begin

Role required:

- You must have the workflow_admin or workflow_creator role to use the Workflow Editor.
- If you are designing the workflow as part of an update set process, see Workflow movement with update sets before creating the workflow.

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
   The **Welcome** tab of the Workflow Editor opens.
2. On the **Workflows** tab in the palette, click **New Workflow**.



A simplified version of the New Workflow form opens.

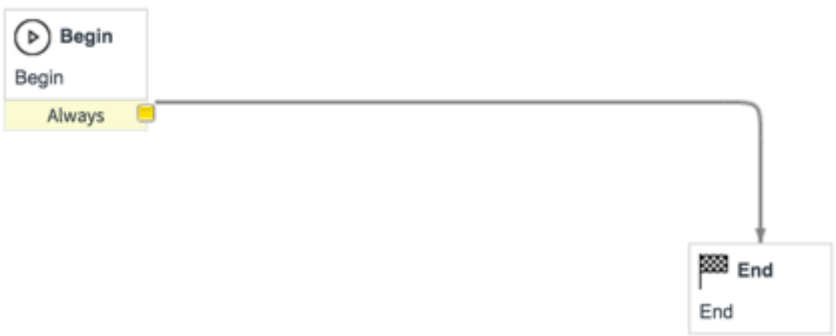
3. Fill in the **Name** and **Table** fields
4. **Optional:** Add a **Description**.
5. **Optional:** Do one of the following:

   a. If the **Conditions** UI section is displayed, specify a **Condition** if needed and edit the fields.
   The **Conditions** UI section shows only if the selected table supports conditions for launching workflows. For example, if you select the sc_req_item table, conditions are not applicable and the **Conditions** UI section is not displayed.

   b. If the **Stages** UI section is displayed, check that the **State rendering** and **Stage order** fields contain the correct information.
   The **Stages** UI section shows only if the selected table supports stages. For example, if you select the sc_req_item table, the **Stages** UI section is displayed.

6. Click **Submit**.

   The new workflow is created with the **Begin** and **End**activities connected by a single transition.



7. Finish creating the workflow by adding activities, validating, and publishing so the workflow is available to other users.
   For more information, see Work on workflows.

8. To change advanced settings for the workflow, click the **Properties** icon ⓘ.
9. If you make changes, click **Update**.

Workflow properties

In the properties of a workflow, you can configure settings such as its application scope, start conditions, schedule, inputs, stages, and run time metrics. You can also view information such as the workflow author, version, and history.

When you create a new workflow, the following fields are available in the dialog box:

- **Name**
- **Table**
- **Description**
- **If condition matches**
- **Condition**

If you click **Diagrammer view**, in **Related Lists**, the following UI sections are available in the dialog box:

- General
- Conditions
- Inputs
- Activities
- Application
- Schedule
- Stages
- Estimated Runtime

General

Field	Description
Name	A name to identify the workflow.
Table	The table for the workflow to run on. Workflows that run on specific tables can still interact with other tables. Select **Global [global]** to run the workflow on all tables. **Note:** The list shows only tables and database views that are in the same scope as the workflow. Also, all users who edit the workflow must have access to the necessary tables and domains.
Checked out	[Read-only] When the workflow was checked out. Automatically set by the **Checkout** action in the workflow menu.
Checked out by	[Read-only] The user who has this workflow checked out. This value is automatically set by the **Checkout** action in the workflow menu.
Published	[Read-only] Check box to indicate whether the workflow has been published. Automatically set by the **Publish** action in the workflow menu.

Field	Description
Description	The purpose of the workflow.

Conditions

Create conditions to trigger the workflow. The Conditions section does not appear if you select a table, such as sc_req_item, that does not require a condition.

Field	Description
If condition matches	When the condition evaluates to true, the workflow launches an active context: • **None**: The workflow is not automatically started by the workflow engine. To run this workflow, write a script to start the workflow. • **Run the Workflow**: The default value. The workflow engine starts the workflow if the information in the **Condition** field matches a record that is inserting into the table. • **Run if no other workflows match yet (deprecated)**: The workflow only runs if no other workflows are running on the execution thread that started this workflow. Avoid using. • **Run if no other workflows matched**: The workflow only runs if no other workflows are running on a specific record. For example, there are four workflows inserted into the Incident table, which have a condition such as **short_desc** contains **test**. A new workflow, which has **If condition matches** is set to **Run if no other workflows match yet**, only runs if none of the four workflows have started running on the Incident record.
Condition	A condition builder for specifying workflow conditions that trigger the behavior selected from the If condition matches list.
Order	Numeric value that determines the order of the workflow, relative to other workflows. Workflows are evaluated in order from the lowest order number to the highest. A workflow runs if it is the first to match conditions.

Inputs

The Inputs section lists all the activities in the current workflow that input data, the data type, and the default value. The Inputs section is only available after a workflow has been created. To create a variable, click **New**.

Field	Description
Label	Displayed column label. Localized depending on user locale.
Reference	Input field from another table.
Type	Data type. For example, integer or string.
Default value	Value used if you do not provide a value.

Activities

The Activities section enables you to set activity pinning and maximum activity count.

Field	Description
Activity pinning	List of options that control updates to custom activities at the workflow level. Pinning protects custom activities from being updated automatically when downloaded from the ServiceNow Store. For more information, see Workflow activity pinning. The possible options are: • **Set by activity**: Allows all activities in the workflow to use their own pinning settings. This is the default pinning option. • **Pin all activities**: Pins all activities in the workflow to their current version. • **Unpin all activities**: Allows all activities in the workflow to be updated.
Max activity count	The maximum number of activities performed by the workflow. This value is used to prevent infinite loops and is set to **100** by default. When the stated maximum count is reached, the workflow is canceled. If this field is blank, the maximum count is set to **-1**, and the workflow is canceled.

Application

The Application section enables you to see application scope and scope restrictions.

Field	Description
Application	[Read-only] Scope of this activity. For more information, see Application Scope.
Accessible from	Scope restrictions for this workflow. Possible settings are: • **All application scopes**: Workflow is accessible to all application scopes. • **This application scope only**: Workflow access is limited to the scope named in the **Application** field. For more information see Workflow scope.

Schedule

Use the Schedule section to create a schedule for this workflow using the schedule builder.

Field	Description
Delivery based on	The schedule type for this workflow. Possible types are: • **User-specified duration**: Duration based on a user-specified value. This is the default schedule type. • **Relative duration**: Duration calculated from a preconfigured schedule, such as **8-5 weekdays**.
Expected time	User-defined interval. This field is visible when the schedule type is **User-specified duration**.
Schedule	Preconfigured schedule that determines when this workflow runs.
Timezone	Time zone for this instance.

Stages

The Stages section appears if you select a table with **Type** = **Workflow**.

Field	Description
Stage	Displays the workflow stage progress on the selected table. Optionally, select **Stage rendering** and **Stage order** schemes to customize the appearance of the stage field. The default values cover typical scenarios.
Stage rendering	The renderer to use when displaying stage icons on a form or list view. For more information about renderers, see Workflow stage renderers.
Stage order	The order of workflow stages when you view a workflow field in a list. Select **Computed** to let the workflow engine compute the stage order from the order of execution in the workflow. Select **User Specified** to use the **Order** field from that Workflow stages.
Name	The name of the stage as it appears in workflow fields..
Duration	Time allocated for the specific stage.
Order	The order of workflow stages when you view a workflow field in a list. Select **Computed** to let the workflow engine compute the stage order from the order of execution in the workflow. Select **User Specified** to use the **Order** field from that Workflow stages.
Value	The value of the stage when it is referenced from elsewhere in the system, such as in a script.

Estimated Runtime

The **Estimated Runtime** section opens the controls for configuring the **ERT** for the workflow. Core workflows included in the base system are not configured for estimated run time by default. All new workflows are configured with default ERT values automatically. You can edit existing run time estimates or configure new ones for any existing workflow. For details about how estimated run times are configured and calculated, see Workflow run time metrics.

Field	Description
Requires ERT	Check box to indicate that this workflow requires an estimated runtime configuration. You can use the ERT calculations to determine if workflows are running longer or shorter than expected and to identify errors in workflow processing. By default, new workflows require an ERT.
Estimated Run Time	The initial estimate for the workflow's run time.
Number of data points	[Read only] The number of times the system has compared the estimated run time to an actual run time result.
Outlier Percentage Threshold for ERT	[Required] The percentage deviation from the estimated run time that identifies an outlier workflow run time. The system uses a default value of 20. For more information, see Outlying workflow run times.

Create a workflow from a table

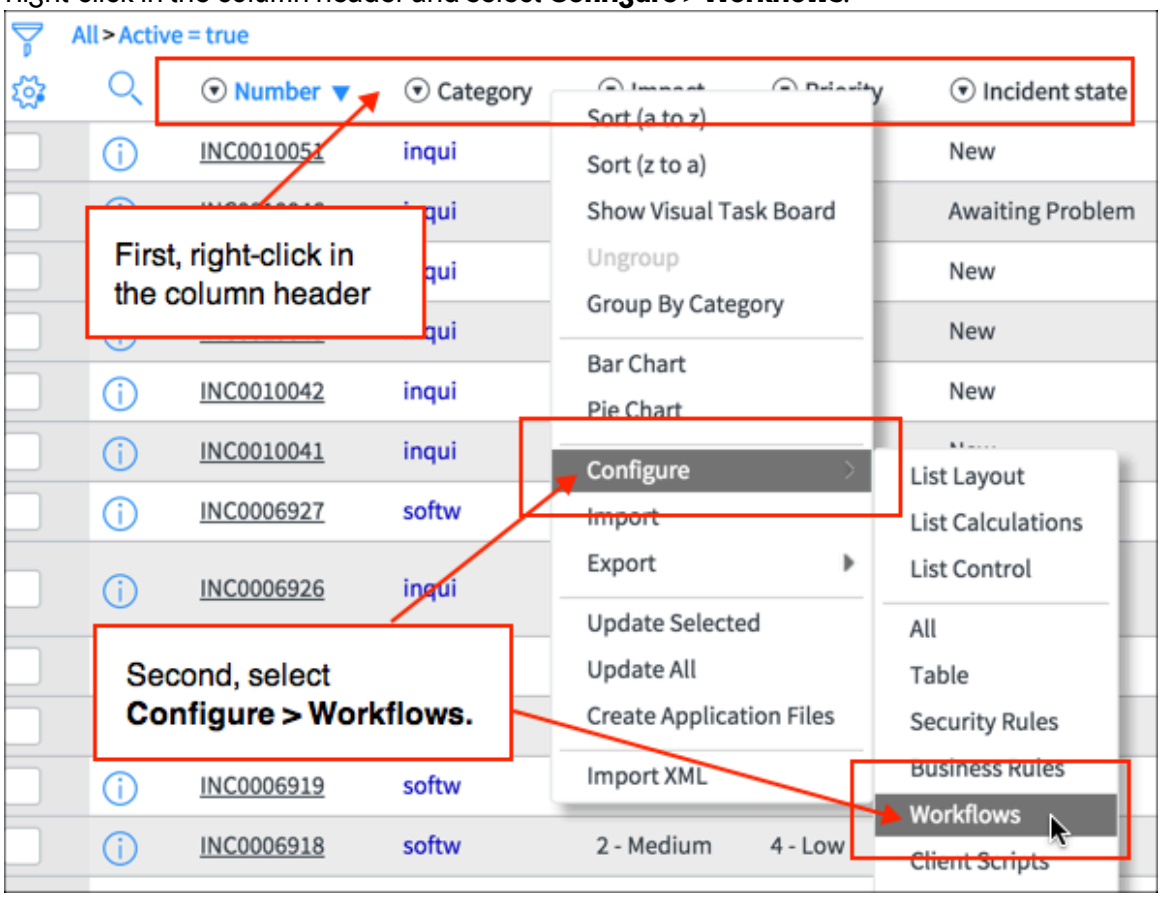
Automate a multi-step process by creating a workflow from the list view of any table that supports workflows.

Before you begin

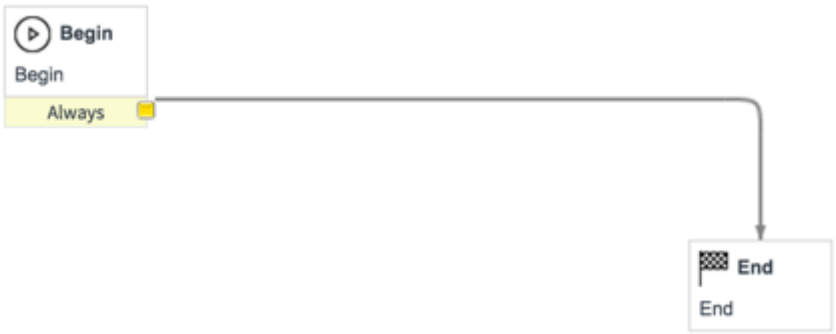
If you are designing the workflow as part of an update set process, see Workflow movement with update sets before creating the workflow.

Procedure

1. Open a table, such as Incident or Problem, in list view.
   For example, navigate to **Incident > Open**.
2. Right-click in the column header and select **Configure > Workflows**.



   The **Workflow Versions** on that table appear in a list.
3. Click**New**.
   The **Workflow Version** opens in **New Workflow** view. The **Table** field is filled in with the table you selected in step 1 and is read-only.
4. Enter **Name**.
5. **Optional:** Enter **Description**.
6. **Optional:** Edit conditions fields as necessary.
7. Click **Submit**.
   The new workflow is added to the **Workflow Versions** list.
8. Click the workflow **Name**.
   The new workflow is created with the **Begin** and **End**activities connected by a single transition.



9. Finish creating the workflow by adding activities, validating, and publishing so the workflow is available to other users.
   For more information, see Work on workflows.
10. **Optional:** To change advanced settings for the workflow, click the **Properties** icon .
11. If you made changes, click **Update**.

Create a workflow for a new service catalog item

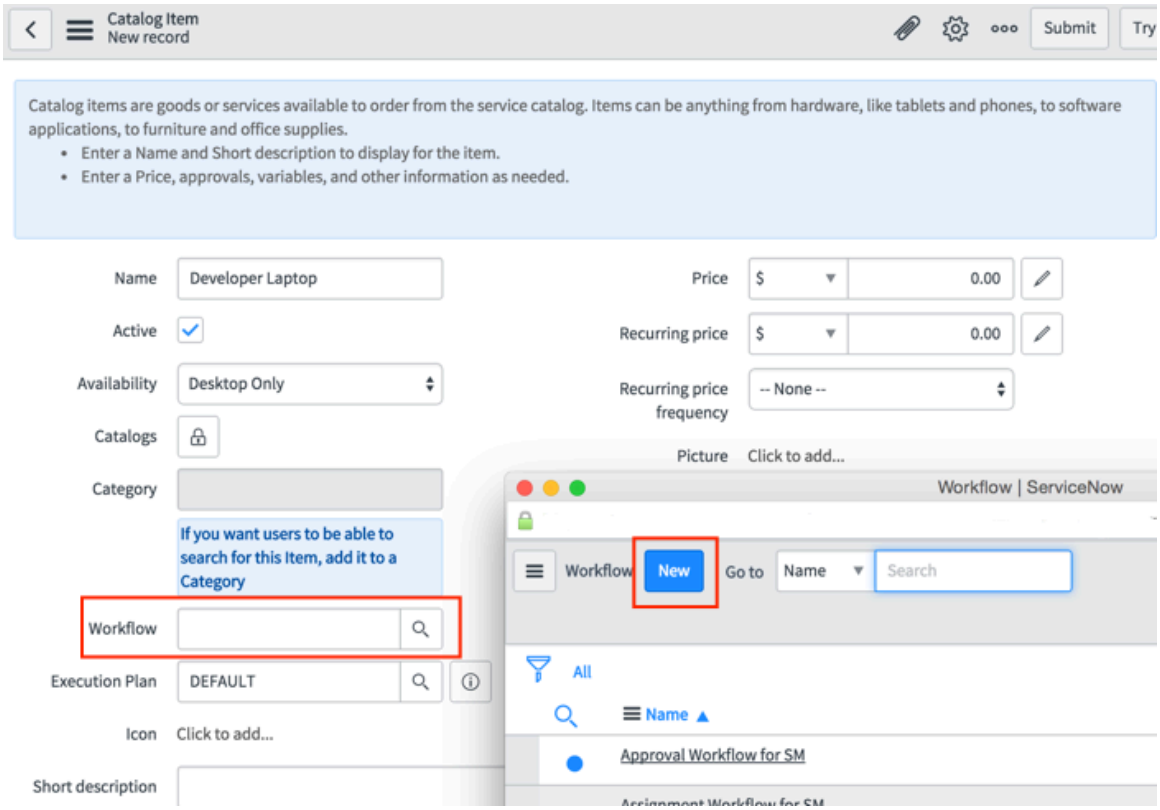
When you create a new service catalog item, you can create a new corresponding workflow at the same time.

Before you begin

- If you are designing the workflow as part of an update set process, see Workflow movement with update sets before creating the workflow.

Procedure

1. Navigate to **All** > **Service Catalog** > **Catalog Definitions** > **Maintain Items**.
2. At the top of the form, next to **Catalog Items**, click **New** .
   The Catalog Item form opens.
3. Add a **Name**.
4. Select the **Process Engine** tab.
5. Next to the **Workflow** field, click the lookup icon .
6. Next to Workflow at the top, click **New**.



The Workflow version dialog opens in the New Workflow View. The **Table** field is set to **Requested Item (sc_req_item)** and is read-only.

7. Add a **Name**.
8. [Optional]Add a **Description**.
9. [Optional] Change the stage information as necessary.
10. Click **Submit**.
    The new workflow is created with the **Begin** and **End**activities connected by a single transition.

11. Finish creating the workflow by adding activities, validating, and publishing so the workflow is available to other users.
    For more information, see Work on workflows.
12. To change advanced settings for the workflow, click the **Properties** icon.
13. Click **Update**.
    If you close the Workflow Editor, you can see the Catalog Item record. Note that the workflow is added to the Workflow field. The Show Workflow and Information icons appear next to the **Workflow** field. Hover over the information icon to view a read-only summary of the workflow.

Create a workflow for an SLA Definition

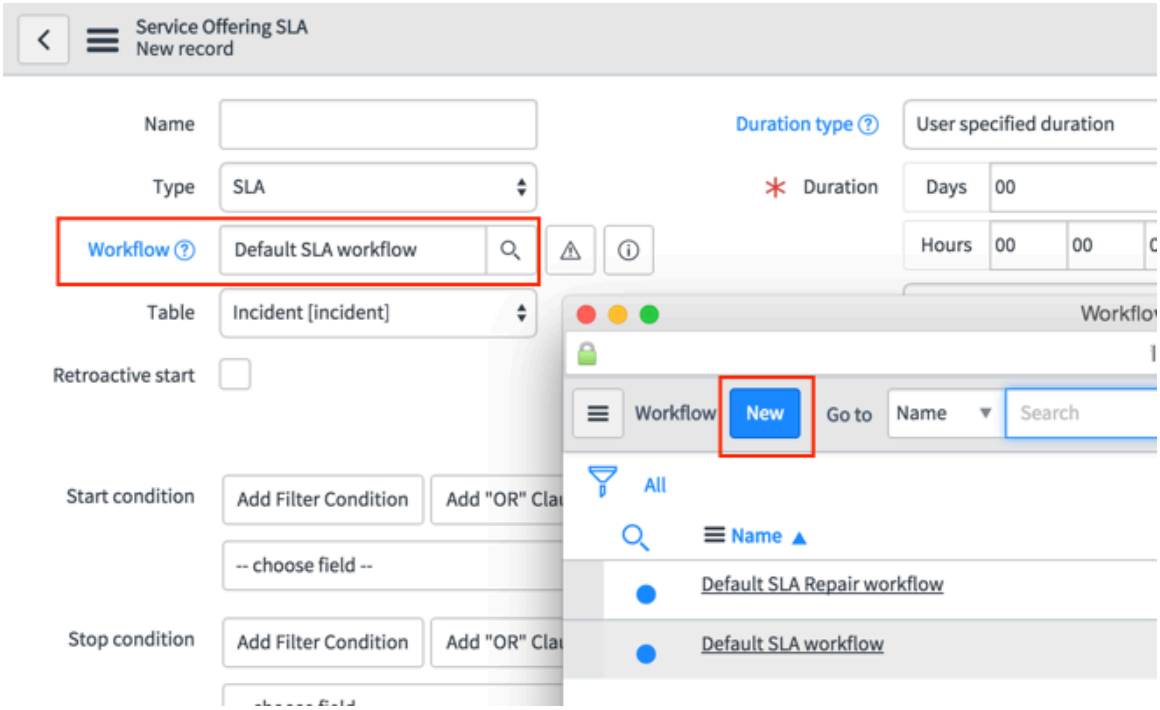
Automate a multi-step process by creating a workflow from an SLA definition.

Before you begin

- If you're designing the workflow as part of an update set process, see Workflow movement with update sets before creating the workflow.

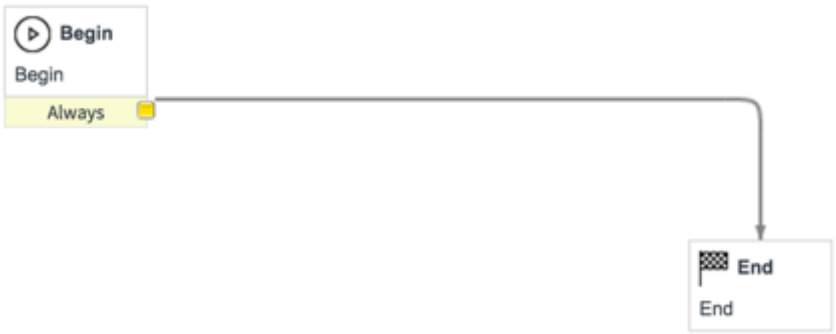
Procedure

1. Open a list of SLA definitions.
   For example **Facilities > SLA Definitions** or **Service Level Management > SLA Definitions**.
2. At the top of the form, next to **SLA Definitions**, click the new icon (New).
3. Select **Contract SLA** or **Service Offering SLA**.
4. Next to the **Workflow** field, click the lookup icon.
5. Next to **Workflow** at the top, click **New**.



   The Workflow Version dialog shows in the New Workflow view. The **Table** field is set to **SLA Definition (contract_sla)** or **Service Offering SLA (service_offering_sla)** and is read-only.

6. Enter a **Name**.
7. [Optional] Enter a **Description**.
8. [Optional] Edit the conditions fields as necessary.
9. Click **Submit**.
   The new workflow is created with the **Begin** and **End** activities connected by a single transition.



10. Finish creating the workflow by adding activities, validating, and publishing so the workflow is available to other users.
    For more information, see Work on workflows.

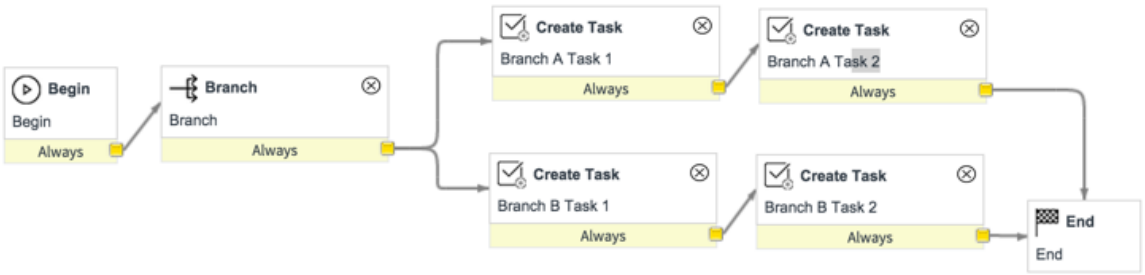
11. [Optional] Change the advanced settings for the workflow by clicking the properties icon ( ).
    If you make changes, click **Update**.

Ending workflows with multiple branches

A workflow is complete when it reaches the **End** activity, even if there are still active branches of the workflow in progress. To ensure that both branches are completed, add a **Join** activity to resolve the branches.

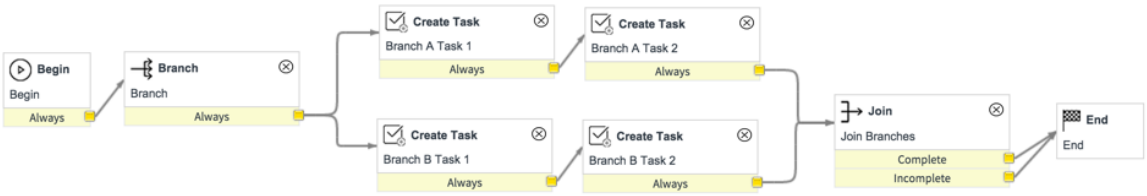
For example, the following figure shows a workflow with two branches that execute independently. When Task 1 and Task 2 of Branch B are completed, the workflow is marked complete even if the Branch A tasks are not completed.

Workflow with uncompleted branch



For both branches to complete, add a **Join** activity to resolve the branches. When one branch reaches the join, the workflow waits for the other branch. When both branches are complete, the workflow reaches the end. The **Incomplete** condition of the a **Join** activity is met only if one of the branches cannot be completed.

Workflow with completed branches



Work on workflows

To complete a workflow, you add workflow activities, validate the workflow, and publish it.

Before you begin
Role required: None.

Add a workflow activity

Activities determine the functionality of the workflow.

Before you begin
Role required: workflow_admin, workflow_creator, or admin

About this task
When they are created, all workflows contain **Start** and **End** activities.

For more information, see Workflow activities.

Procedure

1. Open a workflow.
2. Check out the workflow.
3. Drag a workflow activity from the Activities menu into the workflow body.
4. Populate the Workflow Activity form that appears.

Validate a workflow

You can manually validate a workflow from the Workflow Editor. You can generate a workflow validation report from the Workflow Version form.

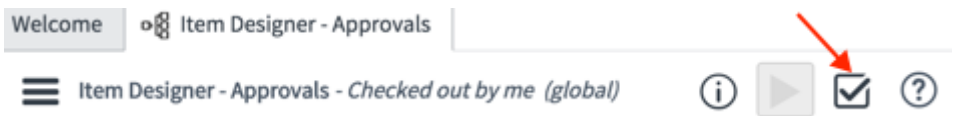
Before you begin
Role required: workflow_admin, workflow_creator, workflow_publisher, or admin

About this task
Running a workflow on a new node automatically attempts to validate the workflow. If validation is successful, the system updates the workflow version record to indicate the workflow has been validated and marks the record as updated by the user who ran the workflow.

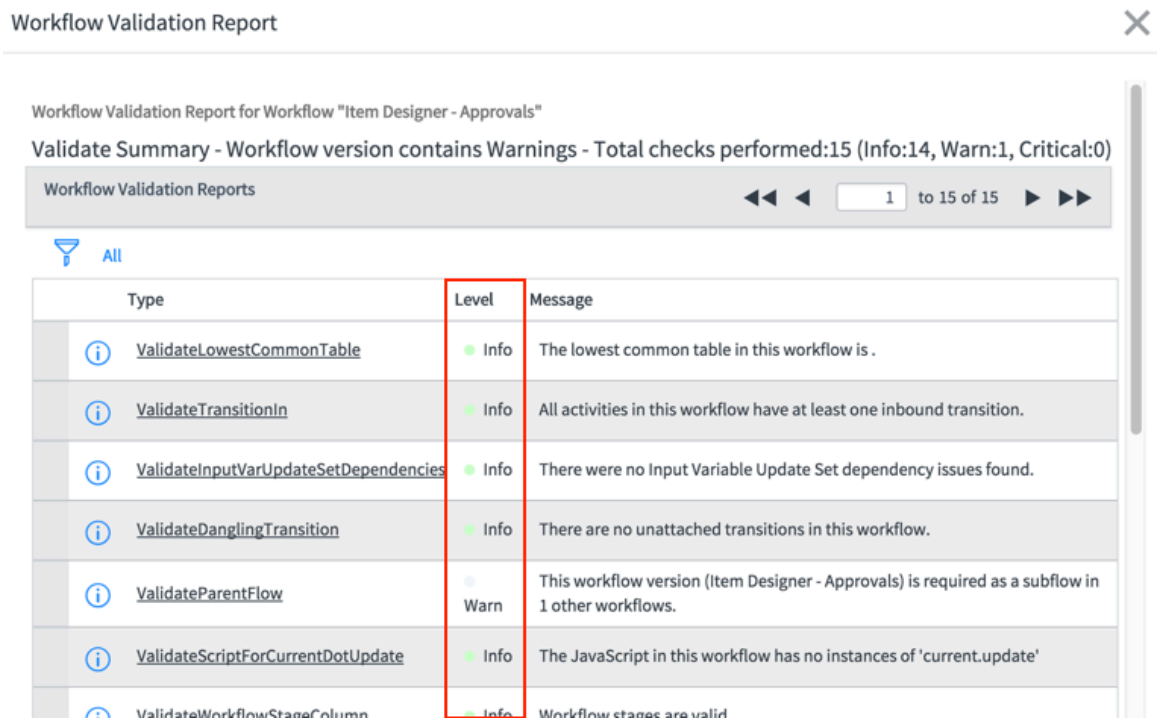
Procedure

1. Open the workflow to validate in the Workflow Editor.

   When the workflow is loaded, the workflow validator icon appears in the toolbar.



2. Click the validator icon to run a series of validation tests on the current workflow version and generate a report.



3. Complete the following steps to generate a workflow validation report from the Workflow Version form
   a. Navigate to **Workflow** > **Administration** > **Workflow Versions**, and select a workflow to validate.
   b. Under **Related Links**, click **Validate Workflow**.

Publish a workflow

When a workflow is complete, publish the workflow so that it is available to all users.

Before you begin

Before you publish a workflow, validate it to test it for issues that might cause it to fail, such as missing subflows or disconnected transitions. For more information, see Workflow validation.

Role required: workflow_admin, workflow_creator, workflow_publisher, or admin

About this task

To publish a workflow:

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open the workflow that you want to publish.
3. In the title bar, click the menu icon and select **Publish**.

Result

If you published a new version of workflow, the changes are not applied to running workflow contexts. Any currently running workflow context continues using the workflow version that was available when the workflow started. The next time the workflow runs, it uses the new version.

Determine whether a workflow can run

A workflow can run only if a checked out version is available to the user who has it checked out, and a valid, published version is available for all users with permission to run it.

Before you begin

Role required: workflow_admin, workflow_creator, or admin

Procedure

1. In the navigation filter, enter `wf_workflow.list`, and then open one of the workflows.
2. In the **Versions** related list, check for all of the following conditions:

   a. There is only one workflow version in a state of **Checked out** and **Checked out by**.

   b. There is only one version and it is not checked out. This version must be both **Active** and **Published**.
   You may need to personalize the list and add the **Active** column.

   c. If there are multiple versions, only one is **Published**.

   These checks determine the only two conditions under which a workflow can run:

   - A checked out version of a workflow is available for the user who has it checked out.
   - A valid, published version of a workflow is available for all users who have permission to run the workflow.

   Main flows containing subflows that do not meet one of these two conditions are not permitted to execute against a current record transaction. Instead, a critical log entry detailing the subflow state is added to the Workflow Context record. To enable the workflow to execute on the next appropriate transaction, remove the subflow from the main flow or modify the published and active states of the subflow.

Edit a published workflow

You can edit a published workflow after you check it out.

Before you begin

Role required: workflow_admin, workflow_creator, workflow_publisher, or admin

About this task

Note:
You cannot check out or delete workflows that are associated with a read-only application file.

To check out a workflow:

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open the workflow that you want to edit.
3. In the title bar, click the menu icon and select **Checkout**.

A new version of the workflow is created and assigned to you.

If you are in a different domain than the published workflow, the new workflow version is created in your domain.

What to do next

After you finish editing the workflow, validate and publish the workflow to make the new version available to other users.

Copy a workflow between two application scopes

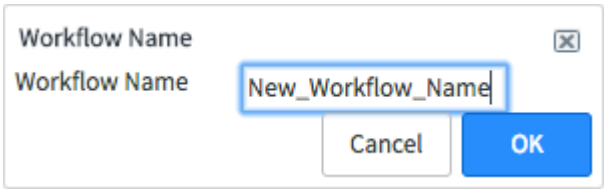
Application scoping protects applications by identifying and restricting access to application files and data. You can copy a workflow created in one application scope (for example, Test) to another (Production) as needed.

Before you begin

Role required: admin

Procedure

1. On the Home page, click , located next to the logged in user name.
2. In the **Developer** tab, in the **Application** field, select the application scope (for example, Test) in which you want to operate the ServiceNow platform.
3. Close the System Settings page.
4. Navigate to **Workflow** > **Workflow Editor**.
5. Create a workflow in the Workflow Editor. For more details, see Create a workflow.
6. In the Workflow Editor, click .
7. In the **Application** tab, **Application** is set to the current application scope selected in System Settings.
8. In **Accessible from**, select **All application scopes** if the workflow is available to all application scopes, or select **This application scope only** if it is only available to, and accessible in the current application scope only.
   Only those workflows that are accessible from all application scopes can be copied to another application scope.
9. Go back to the Homepage, click .
10. In the **Developer** tab, in the **Application** field, select the application scope (for example, Production) to which you want to copy the workflow.
11. Navigate to **Workflow** > **Workflow Editor**.
12. Refresh the page, then open the same workflow you created.
    An **Out of scope workflow, workflow belongs to <scope name> scope** message appears, where <scope name> is the application scope in which the workflow was originally created.
13. In the Workflow Editor, click .
14. Select **Copy**.
    The **Workflow Name** dialog appears:



   a. In **Workflow Name**, type the new name for the copied workflow.
   b. Click **OK**. The system creates a workflow in the current application scope.

15. In the Workflow Editor, click ⓘ.
16. In the **Application** tab, **Application** is set to the current application scope.
17. In **Accessible from**, select **This application scope only** if to make the newly copied workflow a private one that cannot be accessed from outside current scope.
18. Click **Update**.

Result

A new workflow record is created in the selected application scope and marked as private if designated as one in the Workflow Editor.

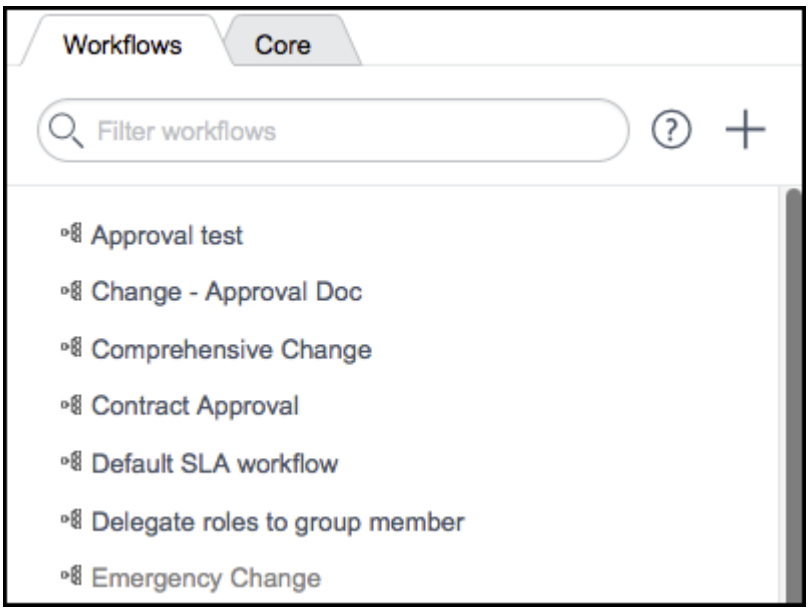
Workflows used as subflows

A workflow can launch another workflow as an activity.

The parent workflow triggers the subflow and then waits for the subflow to complete before continuing. Run the workflow validation tool prior to publishing to detect missing subflows and other dependency problems, such as those involving update sets.

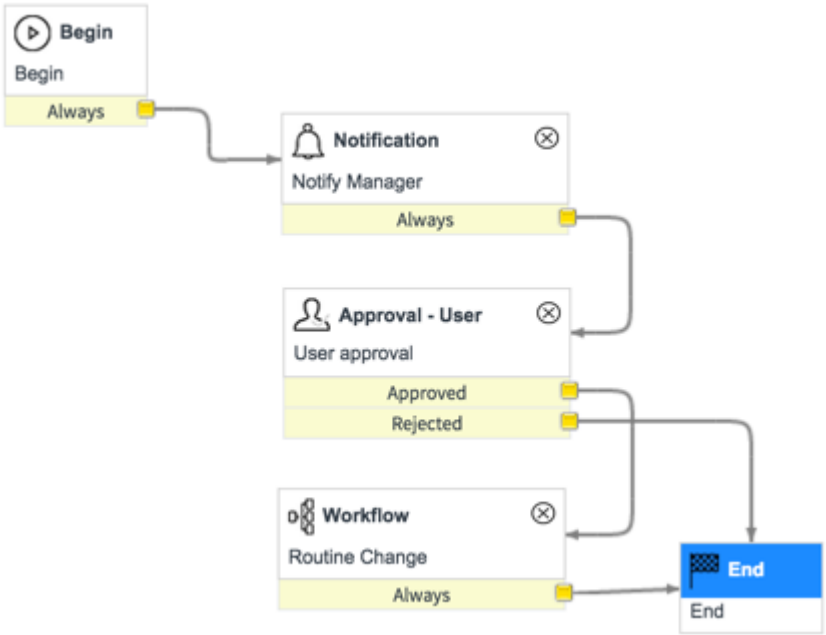
The **Workflows** tab in the Workflow Editor contains a list of the workflows available for use as subflows.

Workflows available to use as subflows



Make sure that the selected subflow is active. If the subflow is inactive, the main workflow will hang with a **Loading** message. If you place an inactive subflow into a workflow, the subflow appears with a red banner, indicating that it cannot run. An active subflow is highlighted in blue when selected.

Workflow with active subflows



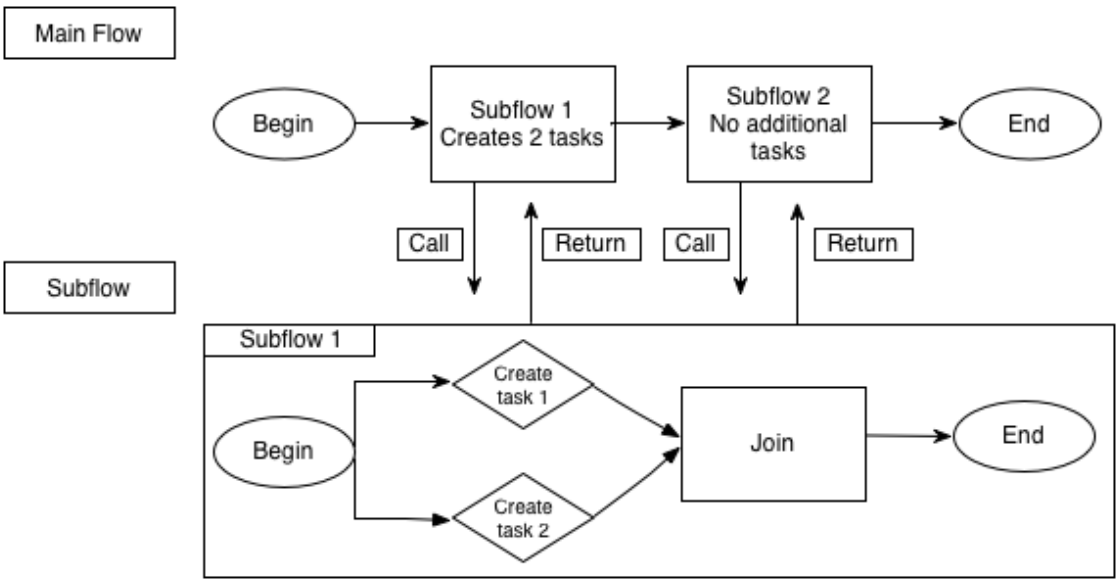
Subflows and the Create Task activity

If a workflow contains a **Create Task** activity that has executed on the current record, additional task activities in the workflow might not execute as expected.

This can happen when the same subflow containing a Create Task activity runs more than once in a parent flow. When the subflow reruns and attempts to execute the **Create Task** activity again, the system reopens the first task activity instead and does not create an additional task.

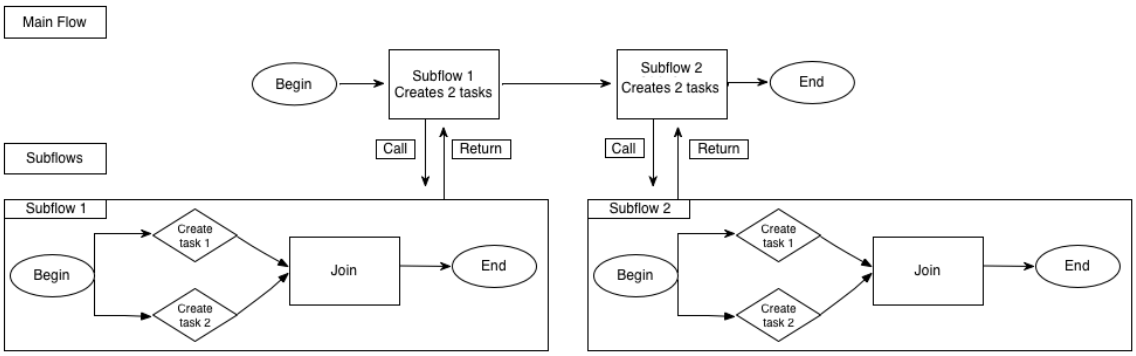
Note:
An alternative to creating duplicate subflows that use the **Create Task** activity is to add a **Run Script** activity to the workflow that creates a task with a script.

The same create task activity runs twice in a workflow



In this configuration, the workflow does not run the same subflow containing a **Create Task** activity more than once. This allows the workflow to create additional tasks.

Running different subflows containing the Create Task activity



Pass a variable from a workflow to a subflow

Use this process to pass variables from a parent workflow to a subflow.

Before you begin

Role required: workflow_admin, workflow_creator, or admin

About this task

Note:
You can also use the **Return Value** activity in the subflow to return values to the parent workflow. Make sure to have a **Return Value** on every ending transition path.

Procedure

1. Prepare the subflow to accept variables from the parent workflow by defining the inputs.
2. Include the subflow in the parent workflow and connect the inputs to the parent workflow variables.

Note:
You cannot pass variables to a subflow that runs on the Requested Item [sc_req_item] table.

Define inputs for a subflow

Define the input variables for a workflow to request from parent workflows when it is launched as a subflow.

About this task

All inputs are stored in the Variables [var_dictionary] table.

Procedure

1. In the editor, open and check out the workflow.
2. In the title bar, click the menu icon and select **Edit Inputs**.
3. In the Workflow Inputs window, click **New**.
4. Populate the record with the definition of the variable, including the column name, the label that is displayed to the user, and the type of field.
5. Click **Submit**.

Invoke a subflow in a workflow

Use this procedure to add a subflow to a workflow.

Procedure

1. In the Workflow Editor, open and check out the parent workflow.
2. Drag the subflow from the **Workflows** tab to the parent workflow.
3. In the New Activity dialog box, define the variables defined by the subflow's **Inputs**.

   These fields can accept both static values or variables in the following format:

   `${variable_name}`
4. Click **Submit**.

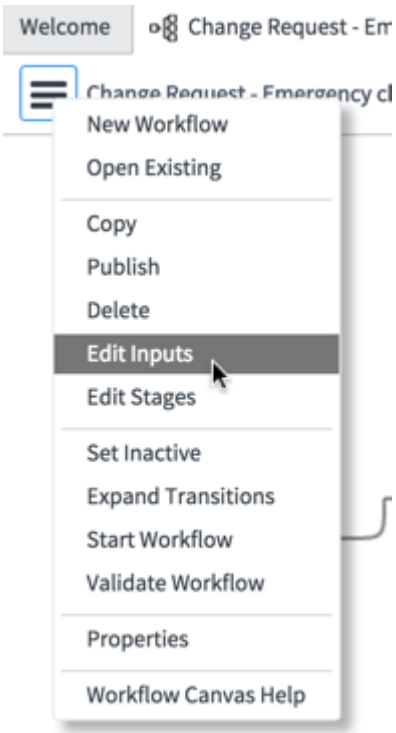
   The workflow triggers the subflow at the appropriate time and passes the variables as indicated by the **Inputs** definition.

Prepare a subflow

Review the process of preparing a subflow for use in a parent workflow, and for preparing the parent workflow to use a subflow.

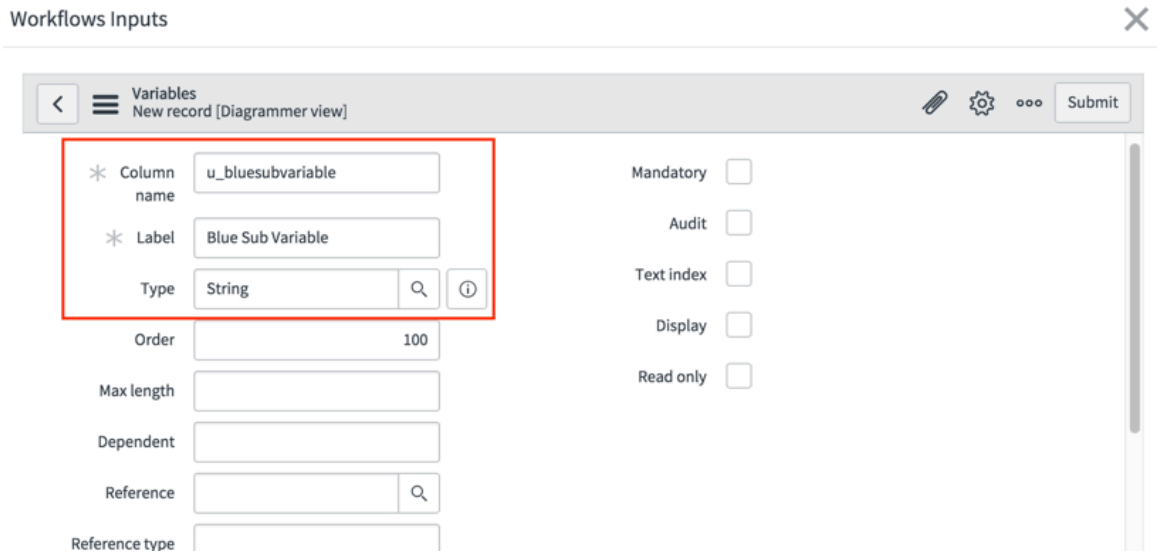
Procedure

1. In the editor, open and check out the workflow that you want to use as a subflow.
2. In the title bar, click the menu icon and select **Edit Inputs**.



3. In the Workflow Inputs window, click **New** in the **Variables** list.
4. Add a new variable depending on the type of values that it is going to store.

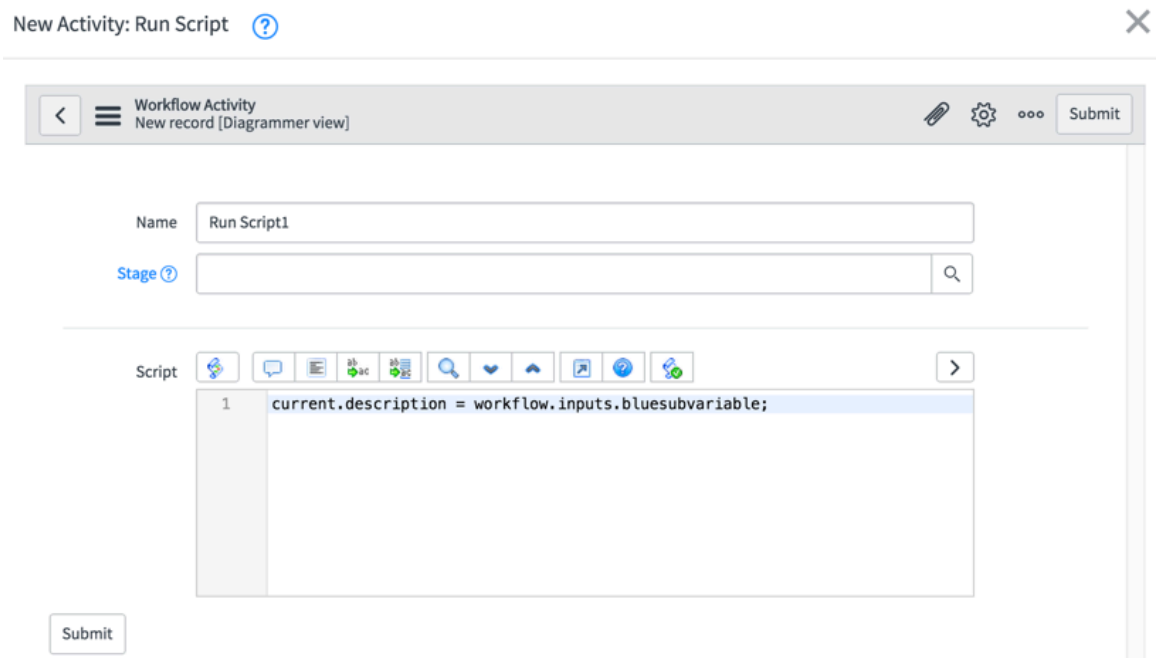
   The following example sets up a string value.



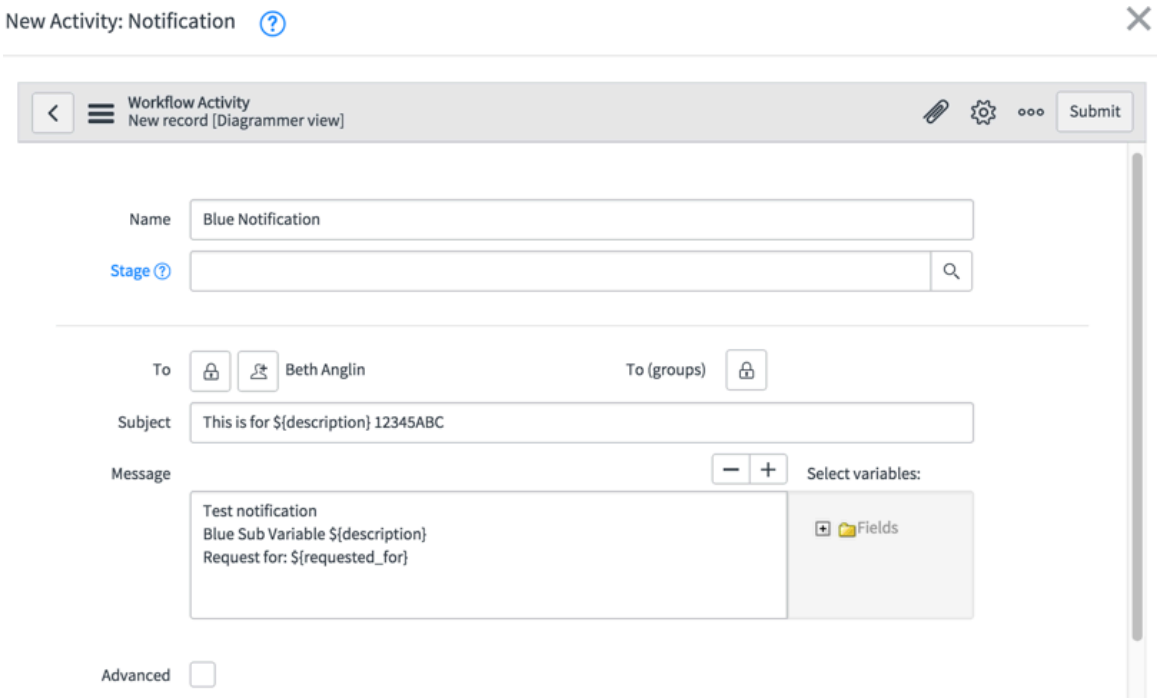
5. Click **Submit**.
6. Close the **Workflows Inputs** dialog.
7. Create a **Run Script** activity on the subflow.
   - Set the value from the parameter to a field on the current form. This is important because the **Notification** activity can only pull values from the current variable and not from the newly added variable. The following example sets the value in the **Description** field.

     ```
     current.description = workflow.inputs.bluesubvariable;
     ```

   - Create a new field on the request form but do not display the field. This serves as temporary storage.



8. Create a **Notification** activity on the subflow and use `${description}` in the subject to return the value from the field.



This is what the subflow would look like:

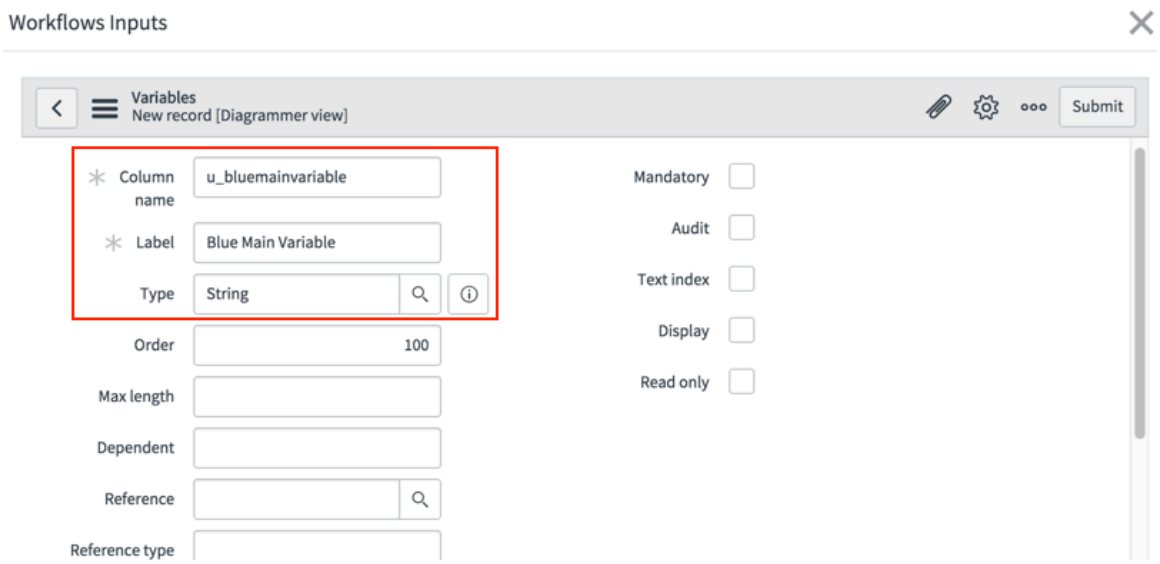
Prepare a workflow to use a subflow

After you create a subflow, use this procedure to prepare the parent workflow.

Procedure

1. On the parent workflow, create a variable similar to what you did on the subflow, but name it something different.

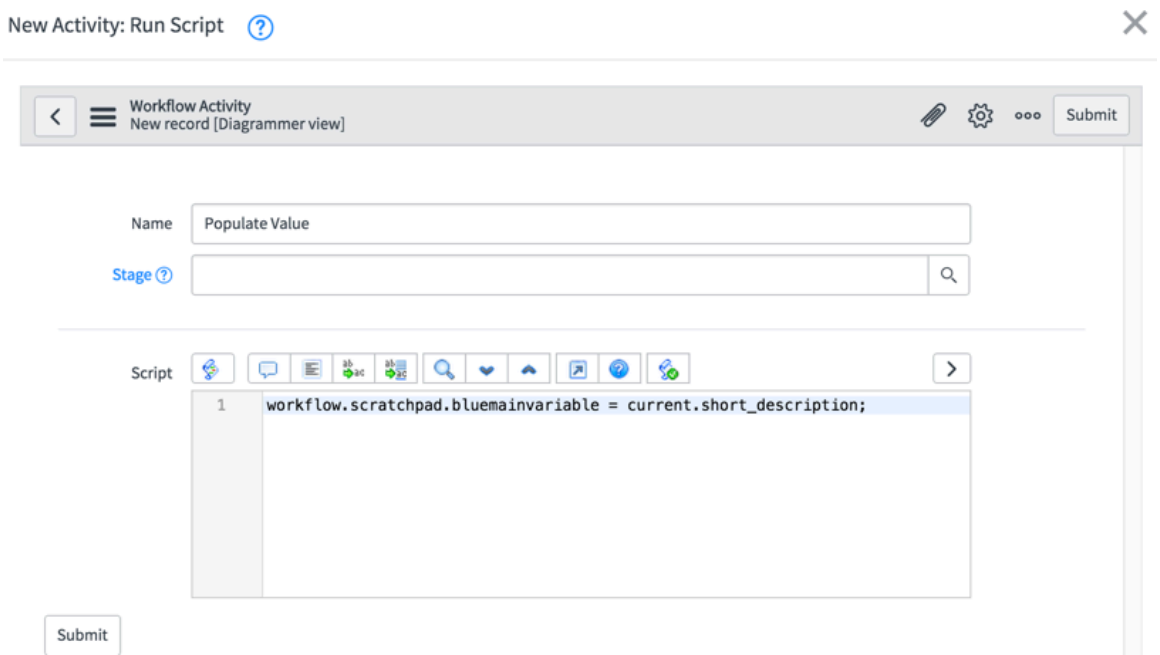
   In the following example, the variable is named **Blue Main Variable**.



2. Click **Submit**.
3. Insert a **Run Script** activity to return the value from a field to the newly created variable.

   In this example, the value of the **Short Description** field is returned and given to the newly created variable.

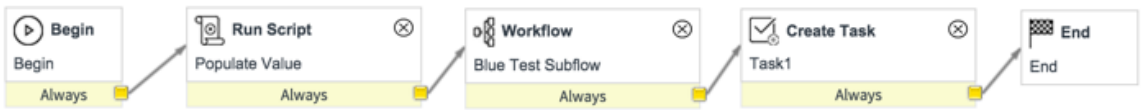
   ```
   workflow.scratchpad.bluemainvariable =
   current.short_description;
   ```



4. Click **Submit**.
5. In the subflow activity, set the **Blue Sub Variable** to pass the `bluemainvariable` to the `bluesubvariable`.

   ```
   ${workflow.scratchpad.bluemainvariable}
   ```

   This is what the main workflow looks like:



Using variables in a workflow

Within workflow there are several different types of variables that are available.

Some variables define, describe, or compose the pieces of the workflow (such as workflow activities). Some variables are available specifically to facilitate the execution of a workflow (such as scratchpad variables). Some variables define, describe, or compose the data records being manipulated by the workflow process (for example, the elements of a Glide Record that are defined in a dictionary.xml file describing a record, such as an incident).

Activity variables

Activity variables are properties specifically associated with a workflow activity.

Before you begin
Role required: admin

About this task
These are the inputs that appear on the activity definition when a specific activity is dragged onto the workflow canvas. These variables are properties that define, compose or describe the workflow activity, or the work it is designed to perform.

Activity variables are declared in a related list within the Workflow Activity Definition.

To add, view, or modify a variable:

Procedure
1. Navigate to **All** > **Workflow** > **Activity Definitions**.
2. Select a workflow activity, such as **Approval - User**.
3. In the **Activity Variables** section or tab, add, view, or modify the variables. Activity variables are defined using the same fields as System dictionary.

Note:
If defining a new activity variable that is dependent on another activity variable, put the prefix "vars." before the name of the parent activity variable. For example, if the child field is dependent on a field named **parent**, put the value `vars.parent` in the **Dependent** field.

To control the visibility of an activity variable on the workflow canvas after the activity has been dragged to the canvas, use a UI policy on the wf_activity table.

To access the variables or assign values to these variables within the workflow activity, use the *vars* variable of the workflow activity.

```
activity.vars.variable_name = new_variable_value
```

Workflow input variables

Workflow variables are external values that are passed into and referenced by a workflow during its execution.

Before you begin
Role required: admin

About this task
Workflow variables can be declared externally in a script and passed into a main workflow or they can be declared within a main workflow and passed as inputs to a subflow.

Note:
There is a specific kind of input variable for any workflow that is declared against the table Requested Item [sc_request_item]. To understand the use of these variables, see Workflow catalog variables.

Workflow variables are declared in a form available from the Workflow Editor gear menu.

Procedure
1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Edit and check out a workflow.
3. Select the Workflow Activities menu and select **Edit Inputs**.

4. Select an existing variable or select **New**.

   Note the **Column name** field. Use this value when accessing the variable from a script.

   Workflow variables do not appear on the workflow canvas, they are only available for view from the Workflow Activity menu.

   The input variables are available to use in scripts. To access an input variable from a script:

```
var input  = workflow. variables. variableName //get the workflow
variable identified by column name .
```

Assigning values to variables from outside of workflow

After being declared in a workflow, values can be assigned to the variables from where the workflow is kicked off.

The following example demonstrates how a workflow variable's value can be set in a script include, business rule, or UI action:

```
//Declare an instance  of workflow.js
var wf  = new Workflow ( ) //Get the workflow id
var  wfId  = wf. getWorkflowFromName ( "Workflow Name" ) ;
//Start workflow, passing along name : value pair(s) for mapping
 to variable
//where input_var_name is the name of the variable declared in
 gear menu
//and input_var_value is whatever that value should be for this
 execution of
//workflow
wf. startFlow (wfId , null , "Workflow Name" , {input_var_name :
 input_var_value } ) ;
```

Another example that demonstrates a more readable way of passing multiple variables involves declaring an object, adding the names/values to the object, and passing it:

```
//Declare an instance  of workflow.js
var wf  = new Workflow ( )
//Get the workflow id
var  wfId  = wf. getWorkflowFromName ( "Workflow Name" ) ;
//Start workflow, passing along object containing name/value
 pairs mapping to inputs expected by the workflow
var vars  = { } ;
vars. input_var_name1 = input_var_value1 ;
vars. input_var_name2 = input_var_value2 ;
// add as many variables as your workflow is expecting, then
 pass the object
wf. startFlow (wfId , null , "Workflow Name" , vars ) ;
```

Assigning Values for Subflow Inputs from inside of a Main Workflow

If a workflow that accepts inputs is called from within another workflow, those input values can be set from the workflow canvas within the UI of the workflow activity. Each input variable declared for a subflow will have a text box input area displayed on the activity. Within these text boxes, the variables can be set with any value either from the scratchpad or from within the current record.

For more information, see Workflows used as subflows.

Reading the Value of a Workflow Input Variable

The value of a Workflow input variable is accessible anywhere within the workflow that accepts javascript by dot walking the current workflow object:

```
var readValue = workflow.inputs.variable_name;
```

Workflow scratchpad variables

Store and share string-based variables as name-value pairs between workflow activities.

The scratchpad is global to the instance of the running workflow and as such, is available equally to all activities.

Declaration

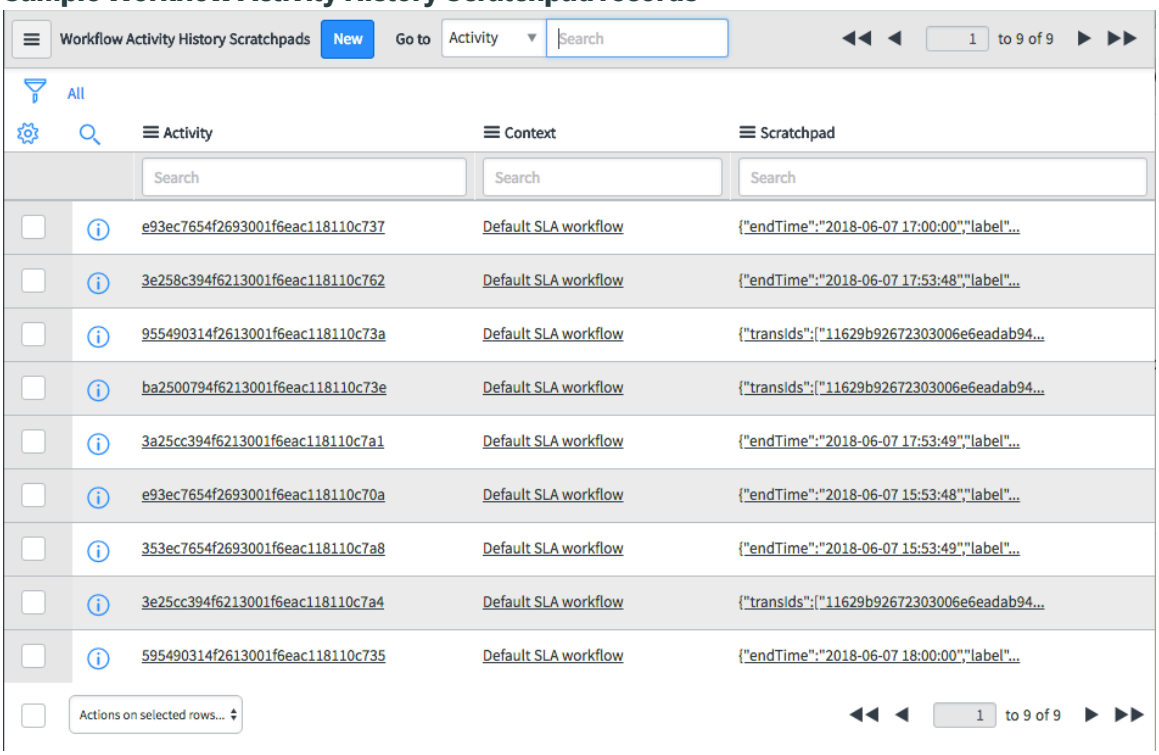
The scratchpad itself is automatically available to an executing workflow and requires no specific declaration. Variables are declared and stored in the scratchpad simultaneously by referencing it. For example:

```
workflow.scratchpad.variableName = variableValue;
```

Display

View activity scratchpad values from the Workflow Activity History Scratchpad [wf_history_scratchpad] table. Workflow Activity History Scratchpad is also an embedded list in Worfklow Activity History records.

Sample Workflow Activity History Scratchpad records



Access and Assignment

Use standard JavaScript object notation to access or assign scratchpad values. For example, use this format to assign a particular scratchpad variable value.

```
workflow.scratchpad.variableName = variableValue;
```

Use this format to assign a JavaScript variable to a scratchpad variable.

```
var myValue = workflow.scratchpad.variableName;
```

Current variables

Current is the database GlideRecord that kicked off the workflow, either by association to the table in the Workflow properties table or by being associated with a catalog item.

Declaration

Variables in current are the columns that are defined in the `dictionary.xml` file that support the database record. There is no way at runtime to add variables to the current record.

Display

Variables of a current record are displayed on the Glide Forms and Lists throughout the applications that use them.

Access and Assignment

To access the variables or assign values to the variables within the workflow activity, get the value from the current record by referencing the Glide Element.

```
var myVarValue = current.getElement(nameOfTheField);
```

or

```
var myVarValue = current.fieldName;
```

or

```
current.variableName.setValue( "A Variable Value");
```

Workflow catalog variables

Workflows that are associated with the Requested Item [sc_req_item] table have a specific type of workflow input.

The inputs into this workflow are essentially question:answer pairings that, when associated with a specific catalog task, become options on the task form. These options are generated by that catalog task activity within a workflow.

Scope

For the purposes of Service Catalog and workflow, a variable's scope determines its availability to a catalog task activity within a workflow. You can define variables as global or catalog item-specific. When a variable is associated to a catalog item and the workflow generating the catalog task is associated to the catalog item, the variable scope determines which variables are available for mapping within the catalog task activity in a workflow. If a workflow is associated with Requested Item [sc_request_item], and is not specifically associated with a catalog item, any catalog variable with a blank Cat Item field value is available to all catalog task activities within the workflow.

Declaration

You can declare requested item catalog variables at the catalog item level or at a global level.

About this task

You can declare global catalog variables using the Workflow Editor context menu.

Also see, Global variable declaration option 2.

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open and check out a workflow that runs against the Requested Item [sc_req_item] table.
3. Click the menu icon and select **Edit Catalog Variables**.
4. Click **New** to create a new variable or select an existing variable.
5. Select the **Global** check box.
6. Click **Submit**.

Global variable declaration option 2

You can declare global catalog variables using the Service Catalog.

Procedure

1. Navigate to **All** > **Service Catalog** > **Item Variables**.
2. The list of variables that appears in the workspace has a default filter of **Cat Item != <blank>**.
   Change the filter to **Cat Item = <blank>** and run the query.

   The catalog variables that appear are available to any catalog task that is initiated from within a workflow.
3. Click **New** to create a new variable.
4. Leave the **Cat Item** field blank.
5. Select the **Global** check box.
6. Click **Submit**.
   Also see, Global variable declaration option 1.

Catalog item specific variable declaration option 1

You can declare a catalog item-specific variable.

About this task

These variables are only available to the catalog item referenced in the **Cat Item** field on the variable record.

Procedure

1. Navigate to **All** > **Service Catalog** > **Catalog Variables** > **Item Variables**.
   Note the list of variables that appears in the workspace has a default filter of **Cat Item != <blank>**.
2. Enter or select a catalog item in the **Cat Item** reference field.
   Selecting a **Cat Item** restricts the scope and availability of the catalog variable to that specific catalog item.
3. Click **Submit**.
   Also see, Catalog item specific variable declaration option 2.

Catalog item specific variable declaration option 2

You can declare a catalog item-specific variable directly from a catalog item record.

About this task

Creating a catalog variable in this way automatically sets the **Cat Item** reference to the catalog item selected.

Procedure

1. Navigate to **All** > **Service Catalog** > **Catalog Definitions** > **Maintain Items**.
2. Select or create an a catalog item.
   Note the **Variables** related list. All variables declared using this related list have a **Cat Item** reference value of the current catalog item.
3. From the **Variables** related list, click **New**.
4. Ensure that the **Cat Item** field correctly references the catalog item previously selected or created.
5. Go to or open the **Question** section or tab.
6. Add a **Question**.
7. Add a **Name**.
   Variable names should not include white space and cannot begin with a number.
8. Click **Submit**.
   Also see, Catalog item specific variable declaration option 1.

Display

Catalog specific item variables are visible in several places depending on where in the process the variable is viewed.

Declaration can happen and variables can be seen from within the menu, inside both the **Maintain Items** module and in the **Item Variable** modules of the Service Catalog.

Within a workflow, the Catalog Specific Item variables are available to the **Catalog Task** activity in the form of a slushbucket at the bottom of the **Catalog Task** activity. Item variables that are selected, are the question and answer pairs that will appear on the task that is generated by that instance of **Catalog Task** when executing that workflow.

If a workflow is associated with a specific catalog item, the association acts as a filter for item variables that appear in the slushbucket of the **Catalog Task** activity.

The last place the variables are seen is in the task form that is generated by the Catalog Task Item. The variables selected in the slushbucket are the question and answer pairs that appear to the user on the task form.

Access and assignment

The Catalog Item Variables are available and assigned to a specific **Catalog Task** activity (thereby to a specific task) using the slushbucket entry of a catalog task.

The user working the task enters the values of the variables.

To access the values of a Catalog Item Variable inside a script:

```
var now_GR  = current ; // or create and query a new GlideRecord
var itemVariable  = now_GR. variables [ variableName
  ] ; //access the service catalog variable identified by the
 variable name.
var itemVariableValue  = itemVariable. getValue ( ) ;
var itemVariableName  = itemVariable. getName ( ) ;
```

```
var itemQuestion  = itemVariable. getQuestion ( ); //All
 GlideappQuestion API are accessible on itemQuestion
```

Workflow events

The system employs two types of events: registered platform events and workflow events.

Registered platform events

Registered events are created in business rules and are used for such tasks as sending email notifications when records are inserted into the database. Workflow events are registered within workflows only and are not used anywhere else in the platform. Registered platform events can be triggered by a workflow for external use, but cannot be used within a workflow.

Workflow events

Workflow events follow different rules than platform events that are registered using the event registry. Platform events are entered into the Event Registration [sysevent_register] table and are available for platform processes to use. Workflow events are triggered exclusively for the workflow engine and are used only to direct the work of executing workflow contexts. When an event is registered in a workflow, it is attached to a currently executing activity in the **registered_events** column of the Workflow Executing Activity [wf_executing] table.

Workflow events also can be broadcast to a workflow from any scripting source that has access to the workflow context, such as a script include or a **Run Script** activity. In this case, the event, such as **cancel**, is passed to all records in the Workflow Executing [wf_executing] table for a specific context.

Whether by registry or by broadcast, an event is handled by the activity definition associated with the currently executing activity. Each activity definition comes with a set of handlers. For example, most activities come with `onExecute`, `onCancel`, and `onUpdate` event handlers. As an example of a more specific event, the **Approval - User** activity also comes with `onDetermineApprovalState`, which is specific to the work of the approval activity.

Multiple parallel events

A single workflow can have multiple event threads running concurrently, such as when a workflow has timers that overlap on separate workflow branches. If any additional thread completes before the first thread, the system stores event information from the additional thread on the Workflow Queued Commands [wf_command] table. After the first thread completes, the system retrieves the information stored by the additional thread and proceeds through the workflow with the event information from each thread.

Workflow events in the base system

Several workflow events are available in the base system.

Workflow events in the base system

Event	Description	Purpose	To use
activityComplete	String value used by activity definitions to respond to the `onActivityComplete` event handler.	Informs records in the Workflow Executing Activity [wf_executing] table about the completion of other activities in	If the activity is allowed to set the boolean value for `wf_executing.notify_ter` then set the value to `true` `(activity.notify_termin` `true)` during the `onExecute` e

Workflow events in the base system (continued)

Event	Description	Purpose	To use
		the same workflow context.	
otherEvent	String value used by the **Join** activity to respond to an otherEvent.	Informs records in the Workflow Executing Activity [wf_executing] table about an otherEvent that has completed.	The **Join** activity transitions from **n** preceding activities. These preced all create a wf_executing record, w a check to see if the record alread **Join** already exists, then the **Join** the executing transition sets the w record for deletion.
timer	String value used by workflow activities to respond to a **Timer** activity that has expired.	Allows wf_executing records to be informed about a timer activity that has completed and has fired the timer event.	The **Timer** activity schedules a job script. The script calls fireEvent (w timer).
execute	String value used by workflow activities to respond to a **Timer** activity that has expired.	Informs a record in the wf_executing table with the initial state of **Executing** to proceed with its primary work.	The workflow engine, for each tran executed, creates an executing re state of **Executing**. Once created, record is put in a queue for proces each item in the queue, the Rhino established, the activity definition the executing record is instantiate `run()` function is called. When th record is **Executing**, this function a `onExecute`.
execute (specific to **Lock**)	String value used by the **Lock** activity to respond to a waiting lock that is ready to make another attempt to obtain a specific lock. This execute is different than the previous execute because it is called on a separate thread, at specified intervals, and is treated as an outside event.	Informs a wf_executing record waiting to execute that the specified wait interval has passed and that it should attempt to get the lock again.	**Lock** activity schedules a job with that uses the workflow script inclu `fireEvent(wf_executing, 'execute')` method.
determineApprovalState	String value used by approval activities to respond to a change in the overall approval status of the current record.	Informs wf_executing records for approval activities about an approval that completed and triggered the timer event.	**Approval Coordinator** both regist the event and triggers the event. T approvals have listeners that dete approval state.
cancel (from within activity definitions)	String value used by workflow activities to	Informs all wf_executing records	The **End** activity uses the global `workflow.broadcastEvent`

Workflow events in the base system (continued)

Event	Description	Purpose	To use
	respond to a request for cancellation.	in a context that the workflow is being canceled.	to interrupt the currently running w records. This changes the state of to **Cancelled**.
cancel (outside current context)	String value used by workflow activities to respond to a request for cancellation.	This event is the same as the cancel event above and handled the same. However, the management of this event is subtly different. This event informs all wf_executing records in a context that the workflow is being canceled. The event is managed via the *onCancel* event handler of each executing activity definition, but the event is called differently. In particular, the call to cancel from outside an activity definition is blocked by the current mutex. This is a significant difference in that the event does not interrupt a currently executing activity that is still operating within the parameters of the current mutex.	Any script can call cancel on a kno context via the workflow script incl example, `var w = new Work w.cancel(context); //wh` is a GlideRecord of the context to
stop (see **End** activity)	The **End** activity checks for this event.	If the stop event is the current event, then the cancel operation of the **End** activity is bypassed.	Only in the **End** activity.
listener	String value that the workflow (subflow) activity triggers as an event.	When a main workflow calls a subflow, the workflow keeps the ID of the subflow's context	The listener event is passed to the context on completion of a subflow managed by the onListener action workflow activity.

Workflow events in the base system (continued)

Event	Description	Purpose	To use
		in the scratchpad. When the subflow is complete, it triggers the listener event via a business rule.	
probe_complete	String value triggered in the workflow by an Orchestration activity indicating that the MID Server has completed a task.	The probe_complete event is triggered from Orchestration sensor processors via the workflow helper function `handleEventById`.	The `onProbe_complete` event the `WebServiceActivityHa` used by most Orchestration activi
pause	String value sent to a workflow from an SLA to pause the **Timer** activity.	When an SLA is paused, the SLA workflows must be paused if there is a timer running.	Use is exclusive to the SLA timer
resume	String value used by the **Timer** activity to resume a paused timer (see pause).	When an SLA is resumed, the SLA workflows must be resumed as well.	Use is exclusive to the SLA timer.

Glide events relative to workflows

Workflow uses several Glide events.

Workflow Glide events

Event	Description	Purpose	To Use	Source	Thread	Listeners
Insert	Global event set upon the insert of a GlideRecord that causes the script engine, and through that the workflow engine, to wake up.	Starts workflows that are associated with the current GlideRecord either by reference, as in request items and SLA timers, or by conditions associated with the GlideRecord's table.	There is no explicit customer-facing use for this in a workflow. It is part of the Glide engine, and with this event, the only thing workflows can do is start. Workflows can also be started manually using a script.	Workflow Engine, RunEngine	Current thread, current mutex	User action of insert
Update	Global event set upon the	Looks to the Workflow Context	There is no explicit customer-facing use for this in	Workflow Engine, RunEngine	Current thread,	User action of

Workflow Glide events (continued)

Event	Description	Purpose	To Use	Source	Thread	Listeners
	update of a GlideRecord that causes the script engine, and through that the workflow engine, to wake up.	[wf_context] table to find running workflows that are associated with the current GlideRecord by document ID.	a workflow. It is part of the Glide engine, and with this event, the only thing workflows can do is advance through the next set of transitions.		current mutex	update of a GlideRecord
Delete	Global event set upon the delete of a GlideRecord that causes the script engine, and through that the workflow engine, to wake up.	Looks to the Workflow Context [wf_context] table to find running workflows that are associated with the current GlideRecord by document ID.	There is no explicit customer-facing use for this in a workflow. It is part of the Glide engine, and with this event, the only thing workflows can do is advance through the next set of transitions.	Workflow Engine, RunEngine	Current thread, current mutex	User action of delete of a GlideRecord
Query	Global event set upon the query of the Glide database that causes the script engine, and through that the workflow engine, to wake up.	Looks to the Workflow Context [wf_context] table to find running workflows that are associated with the current GlideRecord by document ID.	There is no explicit customer-facing use for this in a workflow. It is part of the Glide engine, and with this event, the only thing workflows can do is advance through the next set of transitions.	Workflow Engine, RunEngine	Current thread, current mutex	User action of query of a GlideRecord

Workflow event-specific functions

There are several functions that relate specifically to workflow events.

Workflow event-specific functions

Function	Description	Purpose
registerForEvent (eventName)	Function in the workflow environment that writes events represented as strings to the `wf_executing.registered_events` field.	The workflow events are just strings. When an activity that has registered for an event executes, a comma delimited set of events is stored with the Workflow Executing Activity [wf_executing] record. When the event is triggered in the workflow

Workflow event-specific functions (continued)

Function	Description	Purpose
		context, the wf_executing table looks for all executing records that contain the string that represents the event in the `wf_executing.registered_event` field
unRegisterForEvent (eventName)	Function in the workflow environment that removes a string value representing an event that has been written to the `wf_executing.registered_events` field.	The workflow events are just strings that are written to the `wf_executing.registered_event` field. When an activity unRegisters for an event, the comma delimited set of events stored with the Workflow Executing Activity [wf_executing] record is searched, and if that string is found, it is removed.
fireEvent (eventName)	Function in the workflow environment that examines the contents of the `wf_executing.registered_events` field, comparing its contents to the `eventName` passed in.	The workflow events are just strings that are written to the `wf_executing.registered_event` field. When *`fireEvent(eventName)`* is called by a workflow activity, the workflow engine queues up any executing records that contain the string in the registered field.
fireEvent (eventRecord, eventName)	Function in the workflow environment that sends an event to a specific Workflow Executing Activity [wf_executing] record. The `eventRecord` is a GlideRecord of the type wf_executing.	This event call expects an *`onMyEvent`* event handler in the activity represented in the event record (Workflow Executing Activity [wf_executing] table). When *`fireEvent(eventRecord, eventName)`* is called by a workflow activity, the workflow engine queues up the specific executing record with that event and passes the event into the activity definition for the on<eventName> handler to manage. This event is queued up in its own mutex, so the current queue complete before this event is processed.
fireEvent (eventRecordSysId, eventName)	Function in the workflow environment that sends an event to a specific Workflow Executing Activity [wf_executing] record. The `eventRecordSysId` is the sys_id of a GlideRecord of the type wf_executing.	This is the same as the *`fireEvent`* above except that it accepts an ID and returns the Workflow Executing Activity [wf_executing] record.
fireEvent (eventRecordSysId, eventName, optionalJSONObject)	Function in the workflow environment that sends an event to a specific Workflow Executing Activity [wf_executing] record. The `eventRecordSysId` is the sys_id of a GlideRecord of the type wf_executing.	This is the same as the *`fireEvent`* above except that it accepts a JSON object as a third parameter. This object can specify any data expressible as JSON. You can also specify additional functionality when creating a workflow activity.
broadcastEvent (contextId, eventName)	Function in the workflow environment that sends an event to all currently running Workflow Executing Activity [wf_executing]	This is the same as the *`fireEvent`* above except that it accepts an ID and returns the Workflow Executing Activity [wf_executing] record.

Workflow event-specific functions (continued)

Function	Description	Purpose
	records in a specified context, regardless of their state.	
broadcastEvent (eventName)	Function in the workflow environment that sends an event to all currently running Workflow Executing Activity [wf_executing] records in the current context, regardless of their state.	This should not be confused with `broadcastEvent` above. This event is only available to current Workflow Executin Activity [wf_executing] records.

Event-specific workflow activities

The following workflow activities trigger events.

Event-specific workflow activities

Activity Name	Description	Purpose	To Use	Source	Thread	Listeners
Create Event	Requires an event from the event registry rather than a workflow event. This activity is located in the **Notification** category of the workflow tree.	Fires the notification event specified in the Workflow Activity [wf_activity] table.	1. Navigate to **System Policy** > **Events** > **Registry** 2. Create an event. 3. Navigate to **System Policy** > **Templates** and create an Email templates. 4. Navigate to **System Policy** > **Email** > **Notifications**. 5. Create an email notification that is triggered by the event you created and sends the template you created. 6. On the workflow canvas, drag the **Create Event** activity onto the canvas and associate it with the newly registered event.	Event Registry	Triggered in the current thread and handled on the worker thread in notifications. Never processed by a workflow	On the notificati thread, outside c workflow

Event-specific workflow activities (continued)

Activity Name	Description	Purpose	To Use	Source	Thread	Listeners
			When the workflow executes, the event is created and the email associated with the event is sent.			
Wait for WF Event	Listens for workflow events, as described in the Workflow Events in the Base System table, and only within the current context. This activity is located in the **Conditions** category of the workflow tree.	Waits for another transition branch of the current context to trigger an event.	Takes an event name as input. When the activity is executed, the specified event name is added to a string array stored in the **registered_events** column. The values in this column are in a list of all events the activity waits for when it is executing.	The **Wait for WF Event** activity has a generic `onUnhandledEvent` that tests the current event against the value that was passed to the variable. If they match, the **Wait for WF Event** moves the workflow forward.	Triggered in the current thread or from a script include	The `onUnhandled` of the **Wait for W Event** activity

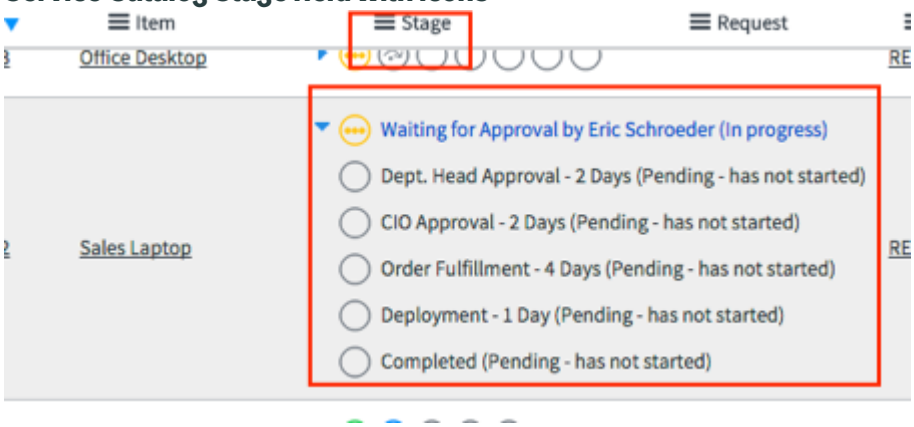
Workflow stages

Workflows can indicate workflow progress by updating any field designated as a stage field.

For example, the Incident [incident] table has an **Incident state** field that indicates progress, but the service catalog uses the **Stage** field.

To indicate the workflow's progress through the possible stage values, the interface updates the **Stage field** selected in the workflow properties. Available fields depend on the table used by the workflow. If the field provides a choice list, then the choices are available as stage values for the workflow. If the field is a workflow field, it displays an icon to indicate the workflow's progress, as with the Service Catalog's **Stage** field.

Service Catalog Stage field with icons



After stages are added to the workflow, they can be assigned to each workflow activity. If an activity with an assigned stage is encountered when the workflow runs, the workflow engine assigns the stage to the record associated with the workflow context.

For workflows that use the Requested Item [sc_req_item] table, the stage field is automatically set to the **Stage** field of the table and cannot be changed. The stage state displayed for a workflow running on the Requested Item table is based on the state of the workflow activities.

- If an activity is active, then the stage is shown with the state of **In progress**.
- If an activity is in the **Pending** or **Completed** state, the stage reflects this state.
- If an activity is canceled, **Request Cancelled** appears in the **Stage** field. The "Cancelled" label set in the wf_stage table is a reserved word, and does not display in the **Stage** field.

How stage values are derived

Stage values are derived from various sources in the interface.

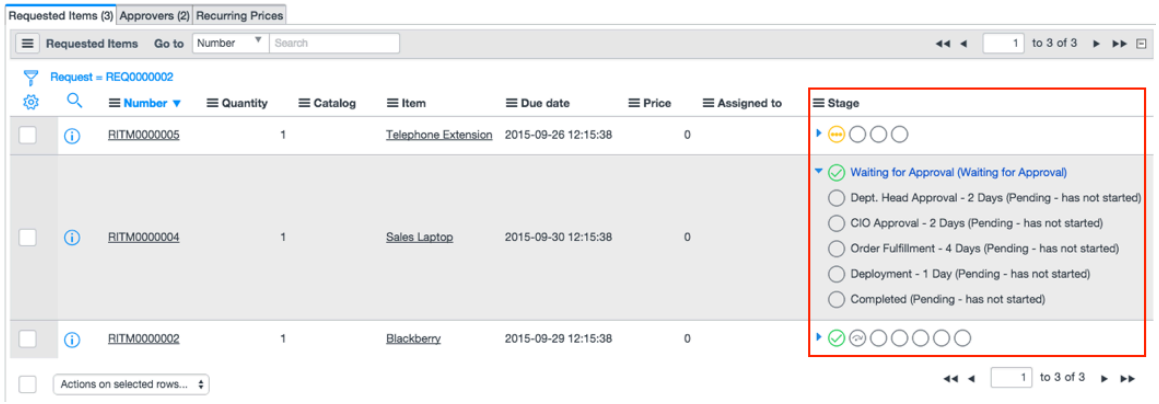
Note:
An updated method for managing workflow stages as a set is also available. For information about grouping and reusing stages, see Workflow stage sets.

Stage values are derived from the following sources:

- **Choices for Stage column**: Choices defined for the column selected as the **Stage** column for the workflow.
- **Default stages for table**: Stages defined in the Stage Default [wf_stage_default] table for the table selected.
- **Workflow-specific stages**: Applied only to the workflow for which they were defined in the Workflow Stage [wf_stage] table.
- **Stage values in existing records**: Values from the designated **Stage** column in the table assigned to the workflow are inherited from existing records.

If the stage field for a workflow is the table column named **Stage**, then the progress of the workflow appears in any list view containing the **Stage** column.

Stages in a list



Stage values shown in the list views are accompanied by the state, based on the workflow activities being executed. If an activity has a stage specified for it, and the activity is currently active in the workflow, then the stage is shown with a state of **In progress**. Similarly, if the activity is in the **Pending** or **Completed** state, the stage reflects this state.

Example

If the workflow table is Request Item [sc_req_item], then the stage field is automatically set to the **Stage** column of that table and cannot be changed. The following stage values for the request item are displayed in a choice list from the Dictionary Entry [sys_dictionary] table:

- Waiting for Approval
- Fulfillment
- Delivery

In addition, the Request Item table has the following default stages:

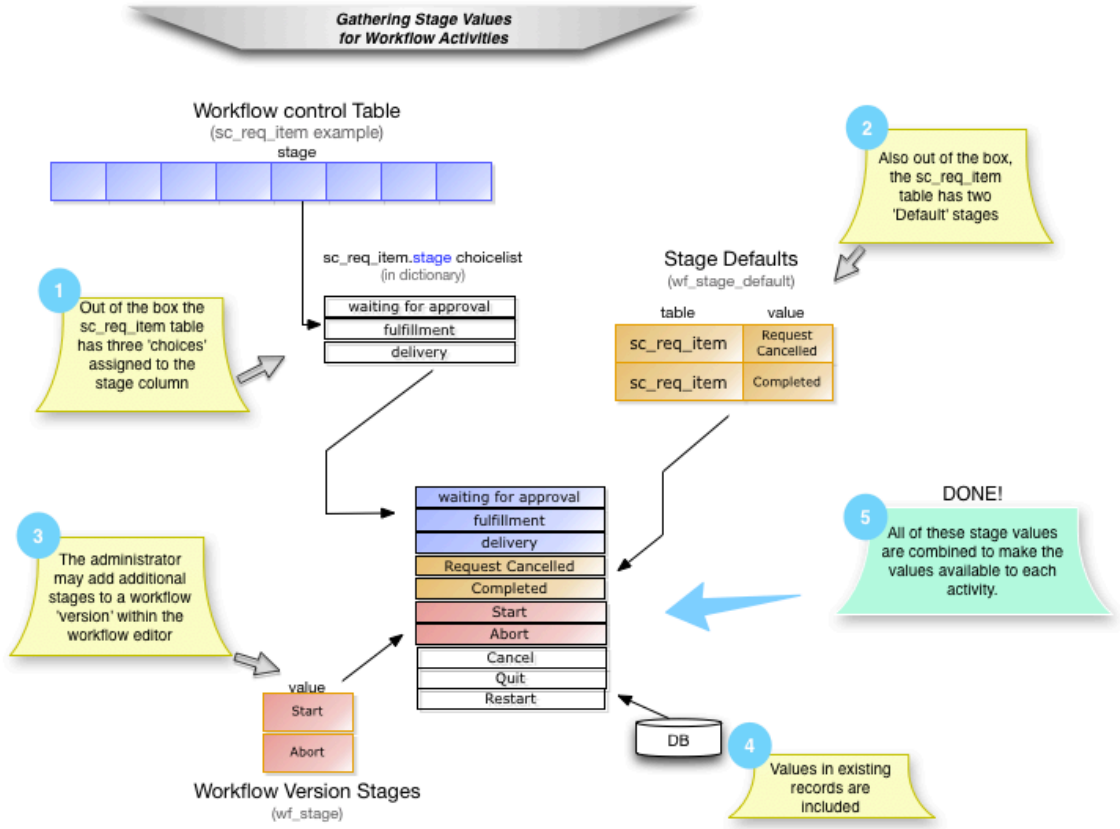
- Request Cancelled
- Completed

Note:
If an activity is cancelled, Request Cancelled appears in the **Stage** field. The "Cancelled" label set in the wf_stage table is a reserved word, and does not display in the **Stage** field.

When you edit available activity stages in the Workflow Editor, the list displays the following stage values:

- Waiting for Approval
- Fulfillment
- Delivery
- Request Cancelled
- Completed

The following diagram depicts the process used to gather stage values from the Request Item table to populate the **Stages** list in workflow activities.



Note:
If you are creating a workflow with a table other than Request Item [sc_req_item], you must select a **Stage field** in the workflow properties for the workflow to have stages.

Use workflow stages

You can add or modify workflow stages.

Before you begin

Role required: workflow_admin, workflow_creator, or admin

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Create a new workflow by clicking **New** or open an existing workflow.
3. In the Workflow Properties form, if the table is not the Requested Item [sc_req_item] table, select a field to display stages in the **Stage** field property.

   Available fields depend on the table selected for the workflow.
4. After assigning a list of stages to the workflow, you can set a stage value in any of the workflow activities that provide a **Stage** field in their dialog box.

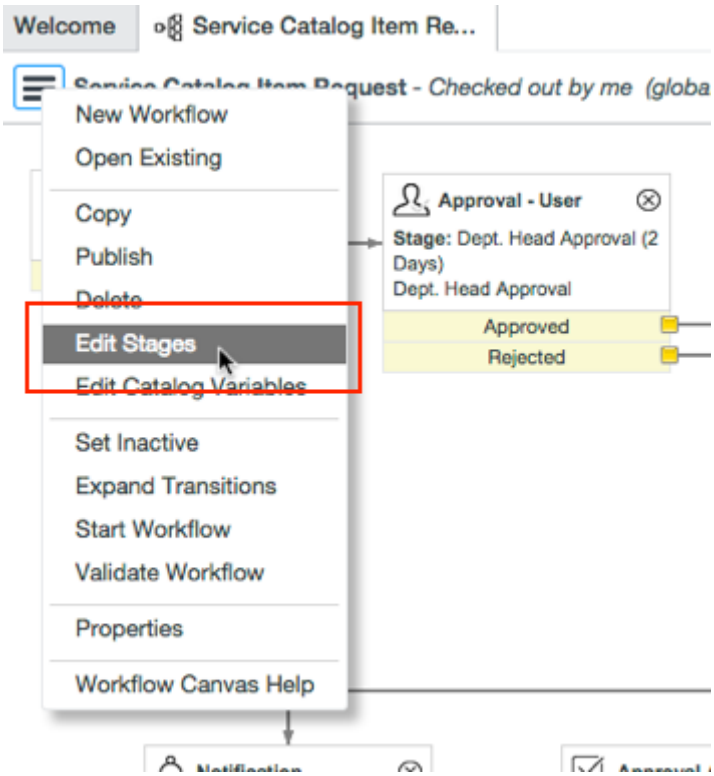
   Setting a stage value in a workflow activity tells the workflow engine to assign that stage to the record associated with the workflow context when the activity is encountered during a run.

   When an activity starts with a specified stage, the platform updates the **Stage** field with the current value. In workflows run against the Requested Item [sc_req_item] table, the `sc_req_item.stage` field is defined as a workflow type field.

When displaying the stages for a workflow on the Requested Item [sc_req_item] table, the stage state is based on the state of the workflow activities. If an activity has a stage specified for it, and the activity is currently active in the workflow, then the stage is shown with an **In progress** state. Similarly, if the activity is in the **Pending** or **Completed** state, the stage reflects this state.

5. To edit the workflow-specific stages for an existing workflow, click the gear icon in the header bar and select **Edit Stages**.

   Again, these stage values are combined with the choices, defaults, and existing values.



Example: Workflow stages

To optimize the use of stages, you can assign a stage to multiple activities in a workflow. For example, if your workflow uses the following activities to create tasks:

1. Get approval
2. Order equipment
3. Receive equipment
4. Add equipment to CMDB
5. Set up equipment
6. Install software
7. Configure software
8. Deliver to user

The following stages might be used:

- Approval
- Order

- Configure
- Deliver

A good practice is to assign stages to the activities as follows:

Examples of stages added to workflow activities

Activity	Assigned stage
Get approval	Approval
Order equipment	Order
Receive equipment	Order
Add equipment to CMDB	Order
Set up equipment	Configure
Install software	Configure
Configure software	Configure
Deliver to user	Deliver

When the **Order equipment**, **Receive equipment**, or **Add equipment to CMDB** activity is active, the **Stage** display shows that the **Order** stage is **In progress**.

Add and translate stages

You can add default stages to use for a table, and stages to existing workflow

Add default stages for a table

You can assign a stage set as a default set to any number of tables.

About this task

The stages in the assigned stage set pre-fill the workflow version stages when you create a new workflow for the associated table.

Procedure

1. Navigate to **All** > **Workflow** > **Default Stages (by table)**.
2. Click **New**.
3. Select a stage **Set**.
   You can create a new set by clicking **New**, typing a name, and clicking **Submit**.
4. Select a corresponding **Table**.
5. Click **Submit**.

   The stages in the selected stage set are automatically added to any new workflows that use the corresponding table.

Add a stage to an existing workflow

If a stage required for a workflow has not been imported or is not in the stage set assigned to the workflow table, you can add it to the workflow manually.

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open and check out the workflow.
3. In the title bar, click the menu icon and select **Edit Stages**.
4. In the Workflow Stages list, click **New**.
5. On the Workflow Stage form, fill in the fields as appropriate.

   Do not use a **Name** or **Value** field value that already exists in the base system.

   #### Workflow stage fields

Field	Description
Name	The name of the stage as it appears in workflow fields.
Value	The value of the stage when it is referenced from elsewhere in the system, such as in a script.
Duration	The default duration for the stage. Currently not used.

6. Click **Submit**.
7. Publish the workflow.

Add a stage to a workflow activity

After stages are added to a workflow, you can assign them to the workflow activities.

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open and check out the workflow.
3. Double-click the activity.
4. In the **Stage** field on the Activity Properties form, find and select the stage to display when the workflow reaches the activity.
5. Click **Update**.

Translate workflow stages

You can provide translations for workflow stage names so the names appear in the language selected for the instance.

Before you begin

The instance must already have language internationalization plugins installed. For more information, see Activate a language.

About this task

Language-specific text appears in Field types displayed in a list, workflow stage sets, and the Workflow Editor for users with that language selected. Language-specific text does not automatically appear when displaying stages on a form. To translate stages on a form, add translated text to the workflow field choices.

You can add translated text for any language enabled on the instance. For example, to translate workflow stages into French:

Procedure

1. Set the interface language to **French**.
2. Navigate to **Workflow** > **Workflow Editor**.
3. Open and check out a workflow.
4. In the title bar, click the menu icon and select **Edit Stages**.
5. Edit the **Name** field for each stage and enter the text to display to French users.

   Do not change the **Value** field.
6. Confirm that the workflow stages display the French text.

Result

Alternatively, to translate stages for multiple workflows at once, you can directly edit the Workflow Stages [wf_stage] table. For example, to translate workflow stages from multiple workflows into French:

1. Set the interface language to **French**.
2. In the application navigation filter, enter `wf_stage.list`.
3. Edit the **Name** field for each stage and enter the text to display to French users. Do not change the **Value** field.

Workflow stage sets

Stage sets are named groups of workflow stages commonly used together. Create a stage set and assign it as a default set to any number of tables. You can import the choice list values of a workflow field as stages for a workflow, and export a stage set to create a new one.

A single stage set usually represents a process, such as the stages required to display the progress of a service catalog request. You can also assign a stage set as the default set for workflows on specific tables.

In an active context, workflow stages provide summary-level feedback about the progress of a workflow. Stage icons display the status of each activity as it is being driven by a workflow. The system updates a **Stage** field, defined in the workflow properties, to indicate the progress of the workflow through the possible stage values.

Updates to workflow stage values

During an upgrade, the system makes changes to the **Value** field of records in the Workflow Stage [wf_stage] table.

- All entries are made lowercase.
- All spaces and special characters are replaced with underscores.

These changes support System Localization and enable workflows to display translated text in the **Stage** field.

Create a new stage set

You can create a new stage set by creating a stage set record and adding stage set entries manually.

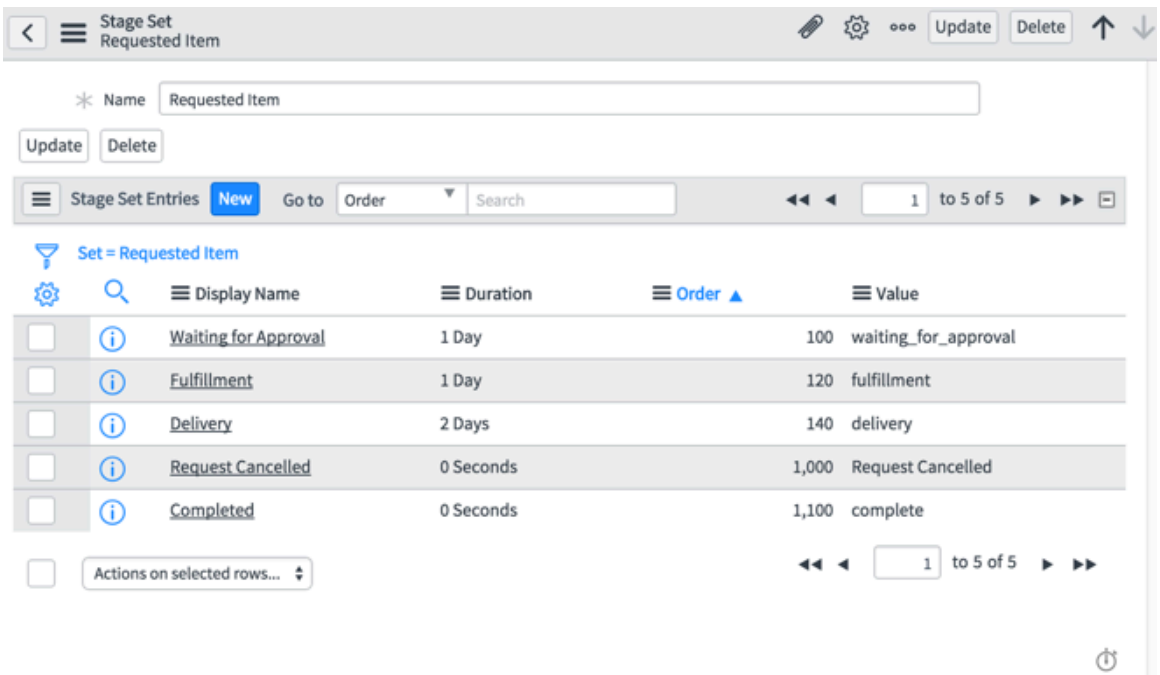
About this task

Create a stage set record only when you want additional stages that are not available on the table by default.

Procedure

1. Navigate to **All** > **Workflow** > **Stage Sets**.
2. Click **New**.
3. Enter a **Name** that indicates the purpose of the stage set.

   Example
   For example, you can create a Requested Item stage set to hold the stages commonly used by inventory tracking workflows or service catalog fulfillment workflows.
4. Click **Submit**.
5. Open the new stage set record.
6. In the **Stage Set Entries** related list, click **New**.
   Each stage set entry can be used as the **Stage** for an activity in a workflow that uses this stage set.
7. Enter a **Name** that indicates the stage name to appear in workflow fields.
8. Enter a **Value** to use when referencing the entry, such as in a script.
9. Click **Submit**.
10. In the **Stage Set Entries** related list, ensure that each entry has a unique **Order** value.
    Enter a low value for stages that should appear early in the workflow and a higher value for later stages.



What to do next

The stage set can be added to a workflow or assigned as the default stage set for workflows that are created for a specific table.

Use a default stage set

You can assign a stage set as a default set to any number of tables.

About this task

The stages in the assigned stage set pre-fill the workflow version stages when you create a new workflow for the associated table.

Procedure

1. Navigate to **All** > **Workflow** > **Default Stages (by table)**.
2. Click **New**.
3. Select a **Table**.
4. Select the **Set** you want to assign to the selected table.
5. Click **Submit**.

Import stages from a choice list

You can import the choice list values of a workflow field as stages for a workflow.

Before you begin

Create a choice list as follows:

- Add a custom field of Type **Workflow** to the table.
- Configure the custom field to use a choice list.
- If you are creating a new field, set the **Choice List type** to **Display without --None--** and create the choices for the newly created field.

For more information, see Create a workflow stage field.

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open and check out the workflow.
3. In the title bar, click the menu icon and select **Properties**
4. In the Workflow Properties dialog box, click the **Stages** tab.
5. From the **Stage field** list, verify that the correct workflow field is selected.
6. In the Related Links section, click **Import Stages from Choice List**.
7. In the dialog box asking you to confirm that you want to import the choice list, click **OK**.
8. Click **Update**.

What to do next

The stage set can be added to a workflow or assigned as the default set for workflows that are created for a specific table.

Export a stage set from a workflow

You can create a new stage set by exporting the stages from an existing workflow as a set, instead of manually adding stage set entries to a stage set record.

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open and check out the workflow containing stages that you want to export as a new stage set.
3. In the title bar, click the menu icon and select **Edit Stages** to open the Workflow Stages dialog box.
4. In the Related Links section, click **Export to Stage Set**.
5. Enter a unique **Name** for the new stage set.
6. Click **OK**.

What to do next
The stage set can be added to a workflow or assigned as the default set for workflows that are created for a specific table.

Add a stage set to a workflow

You can add any number of stage sets to an existing workflow.

About this task
When multiple stage sets have stage set entries with the same **Value**, the stage appears on the workflow only once.

Procedure
1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open and check out the workflow.
3. In the title bar, click the menu icon and select **Edit Stages** to open the Workflow Stages dialog box.
4. In the Related Links section, click **Import from Stage Set**.
5. Select the stage set.
6. Click **Ok**.
7. Import additional stage sets as needed for the workflow.

What to do next
After you add all necessary stage sets to the workflow, you can add them to the workflow activities. For more information, see Add a stage to a workflow activity.

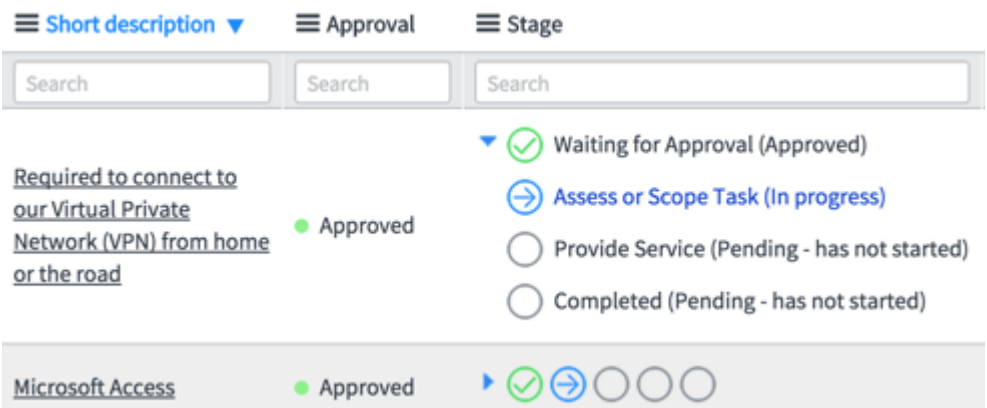
Create a workflow stage field

Workflows can provide a summary of workflow progress by updating any field of the **Workflow** type. If the field is a workflow field, it displays an icon to indicate the workflow stage progress.

Before you begin
Role required: admin

About this task
In a form, the workflow field displays the current stage as a choice list value.

In a list, the workflow field displays stage icons that represent the series of states, stages, phases, or tasks within a workflow. The stage icons are populated using the activity stages from the associated workflow. For more information, see Workflow stages.



To create a workflow stage field:

Procedure

1. Navigate to **All** > **Workflow** > **Default Stages (by table)**.
2. Right-click near a column header and select **Configure** > **List Layout**.
3. Create a new field in lower right.
4. On the form view of the table used by the workflow, create a field with the **Type** set to **Workflow**.
   For detailed steps, see Add and customize a field in a table.
5. Click **Save**.
6. Right-click the field label and select **Configure Dictionary**.
7. Open a dictionary entry.
8. In the Choice List Specification section of the Dictionary Entries form, select **Dropdown with --None--** from the **Choice** list.
9. Click **Update**.
10. To ensure that the workflow field properly displays stages on the form, define a choice for each workflow stage.
    - To create workflow stages using a choice list that already has choices, you can import the stages from the choice list. A stage is created for each choice defined for the **Stage field** selected in the workflow properties. For detailed steps about importing a choice list, see Import stages from a choice list.
    - The **Value** of each choice must match the **Value** of the corresponding workflow stage.
    - Do not use spaces in the **Value** field for either choices or stages. Use underscores in place of spaces.
    - Make the choice list read-only. If a user changes the stage value for a record from the choice list rather than allowing the workflow to control the value, the workflow-driven, legacy, and linear renders may not work as intended. You can also use business rules or events to ensure the workflow progresses accordingly.

    For detailed steps on defining choice lists, see Define an option for a choice list.
11. To see workflow stages as icons, add the field to the list layout of the workflow table.
    For detailed steps, see Configure the list layout.

What to do next

After you create or update the workflow field:

- Unless the workflow uses the Requested Item [sc_req_item] table, ensure that the workflow field is selected from the **Stage field** list in the workflow properties. For detailed steps, see Select a stage field. If a workflow uses the Requested Item table, the stage field is automatically set to the **Stage** field of the table and cannot be changed.
- Work through the workflow. Any other updates made to the workflow field, such as updates from business rules or other scripts, can interfere with displaying workflow stages.

Select a stage field

A **Stage field** allows the workflow context to show additional workflow information, such as the stage name and the estimated completion time for an activity.

Before you begin

Ensure that the workflow field you want to use as the stage field is configured to properly display stages. For detailed steps, see Create a workflow stage field.

About this task

Unless the workflow uses the Requested Item [sc_req_item] table, you can specify which field from the workflow table is the stage field. For workflows that use the Requested Item [sc_req_item] table, the stage field is automatically set to the **Stage** field of the table and cannot be changed.

To add or edit a workflow stage field:

Procedure

1. Navigate to **Workflow > Workflow Editor**.
2. Create or check out the workflow.
3. In the title bar, click the menu icon and select **Properties**.
4. In the Workflow Properties dialog box, click the **General** tab.
5. In the **Table** list, verify that the table containing the workflow field is selected.
6. Click the **Stages** tab.
7. From the **Stage field** list, select the workflow field.
8. Click **Update**.

Display approvers in workflow stage fields

Enable workflow stage fields to display approvers, change the number of approvers to display, or disable displaying approvers.

Before you begin

Role required: admin

About this task

By default, only workflow stage fields that use the Workflow-driven renderer can display a list of approvers. Only these workflow stage renderer types support displaying approvers.

- Linear renderer
- Main flow renderer
- Workflow-driven renderer

Procedure

1. Navigate to **All** > **Workflow** > **Administration** > **Properties**.
   The system displays the Workflow Properties page.
2. Set the following properties.

Property	Description
Number of approvers to show if approvers are displayed for workflow stages. Only valid for supported list v2 renderers. glide.workflow.renderer.show_approver_limit	Sets the maximum number of approvers to display within a workflow stage field. The system displays an ellipsis character when there are more approvers than the display limit allows. ○ Type: integer ○ Default value: 5
Show approvers when displaying workflow stages with the Linear renderer.	Enables (true) or disables (false) the Linear renderer to display approvers.

Property	Description
glide.workflow.renderer.linear.show_approver	○ Type: true | false ○ Default value: false
Show approvers when displaying workflow stages with the Main flow renderer. glide.workflow.renderer.mainflow.show_approver	Enables (true) or disables (false) the Main flow renderer to display approvers. ○ Type: true | false ○ Default value: false
Show approvers when displaying workflow stages with the Workflow-driven renderer. glide.workflow.renderer.workflowdriven.show_approver	Enables (true) or disables (false) the Workflow-driven renderer to display approvers. ○ Type: true | false ○ Default value: true

3. Click **Save**.
4. Add stages to workflows that have associated workflow stage fields.

   Note:
   If you add stages to a subflow, the parent workflow must also have stages.

5. For each approval activity you want to display approvers, select a **Stage** value.

   Example
   For example, the sample workflow **Service Catalog Item Request** has two approval activities. The first **Approval - User** activity has a **Stage** value of `Dept. Head Approval`. The second **Approval - User** activity has a **Stage** value of `CIO Approval`.

Result
The workflow stage field renderers you enabled display approvers up to the approver display limit. For example, a Workflow-driven stage field displays up to five approvers when the workflow reaches an approval stage.

Workflow stage field icons and tooltips

A workflow stage field displays icons to indicate the workflow stage.

Based on the stage renderer selected for the workflow, these icons may display a tooltip with additional information.

Workflow field icons

Icon	Workflow stage
	Current active step
	Approval pending
	Approval rejected
	Complete
	Late (Change/Request) or Canceled (Catalog)

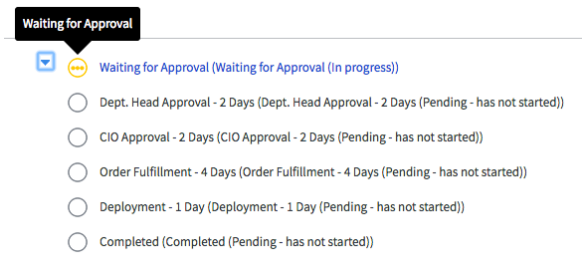
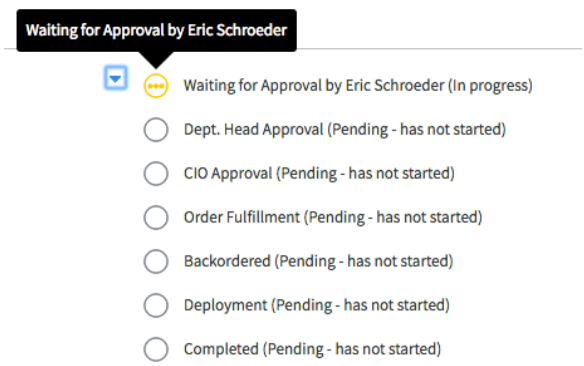
Workflow field icons (continued)

Icon	Workflow stage
	Skipped (Catalog only)

Stage tooltips

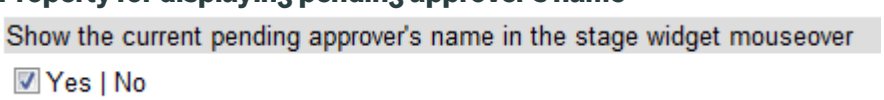
Based on the stage renderer selected for a workflow, workflow stage icons may display tooltips with detailed information about a stage.

Stage renderers and tooltip behavior

Renderer	Tooltip behavior
Legacy	When a user points to a stage, the tooltip displays the name of the stage. In the expanded view, the activity status appears in parentheses next to the stage. 
Other renderers	When a user points to a stage, the tooltip displays the name of the stage. If the stage is a gated approval, the tooltip also shows the name of the approver. In the expanded view, the activity status appears in parentheses next to the stage. 

If you do not want the approver's name included in the tooltip, navigate to **System Properties** > **Service Catalog** and clear the **Show the current pending approver's name in the stage widget mouseover** check box.

Property for displaying pending approver's name



Workflow stage renderers

Workflow stage renderers determine how a workflow displays stages in a workflow field.

There are multiple renderers available.

Note:
Most workflows should use the workflow-driven renderer. This renderer is used by default for all workflows. Use a different stage renderer only after careful consideration and to satisfy specific requirements for how the stages appear.

Workflow-driven

Use the workflow-driven renderer as much as possible. This renderer displays icons for stages using the stage state controlled by the workflow. Icons are displayed in a way that is meaningful for many situations. This renderer can display stages from a main workflow and subflows. The order of the icons is determined by the expected path of the executing workflow. As the workflow progresses, stages on paths that the workflow did not take are removed from the display. Stages from paths other than the expected path are not included unless the workflow follows that path.

The **Stage order** field on the **Stages** tab has two options:

- **Computed** uses the actual workflow path in order
- **User-specified** uses the order specified in the **Order** column on the Workflow Stages record

Note:
If the workflow context for a request item has been deleted, the stages for that request item can no longer be rendered. This stage history is stored on the workflow context.

Main flow

The main flow renderer displays icons for stages defined in the main workflow only. Use this renderer when you do not want to expose the details of the subflows. For example, a single main workflow may run several subflows to handle implementation details. The stages in these subflows do not provide useful information for the user who starts the workflow, but are useful when editing the subflow. In this scenario, using the main flow renderer leads to the best user experience. The stage field displays the high-level process of the workflow without exposing unnecessary details.

The **Stage order** field on the **Stages** tab has two options:

- **Computed** uses the actual workflow path in order
- **User-specified** uses the order specified in the **Order** column on the Workflow Stages record

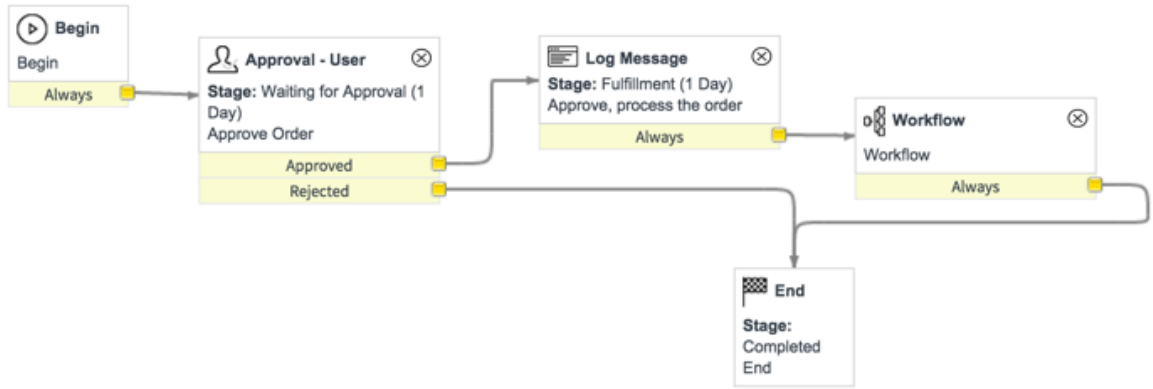
Linear

Linear rendering displays all stage icons from the main workflow and all subflows within a single workflow field on a list or form. It displays icons in a linear sequence regardless of the paths the workflow follows as it executes. This renderer uses stages defined in both the main workflow and any subflows that the main flow launches. The icons appear in the user-specified order. Skipped stages do not appear.

Use this renderer when the workflow stages must display in a consistent order and the actual details of how the workflow runs are less important. For example, a workflow may revisit or revert to previous stages based on one or more Condition Workflow activities. Displaying these loops in the stage field does not provide useful information to the user. In this scenario, using the linear renderer leads to the best user experience. The stage field displays the predefined process, including stages from subflows, in a consistent order.

For example, you can create a service catalog workflow that uses a subflow to complete delivery of the requested item. The following images illustrate the main flow and subflow for ordering corporate-branded supplies.

Linear main flow



Linear subflow

The linear renderer displays all these stages in one workflow field. The parent workflow specifies the **Waiting for approval**, **Fulfillment**, and **Completed** stages. The subflow specifies the **Delivery** and **Waiting for Task**, and **Delivery OK** stages.

Progress bar

The progress bar renderer displays a single progress bar instead of a sequence of stage icons. This renderer is particularly useful when you want to display the general progress of the workflow as a percent. Each stage in the workflow controls an equal percentage of the progress bar. For example, if a workflow has 10 stages, reaching each stage causes the workflow field to display an additional 10% of the progress bar. Stage names do not appear in the progress bar.

The progress bar renderer provides a way to display the status of a workflow that has a large number of stages. Other rendering options may provide a better user experience when rendering fewer than four stages in a single workflow.

The **Stage order** field on the **Stages** tab has two options:

- **Computed** uses the actual workflow path in order
- **User-specified** uses the order specified in the **Order** column on the Workflow Stages record

Workflow field progress bar

Legacy

The legacy renderer displays stages in the same way as releases prior to Dublin. When an instance is upgraded from a release prior to Dublin, all existing workflows are set to use the legacy renderer. Whenever possible, use a different stage renderer instead. If you want your workflow to maintain pre-Dublin behavior, use the Legacy option. The legacy renderer sometimes assigns incorrect data to and/or reports incorrect data from the workflow stage field. If possible, we recommend all workflows use the Workflow-driven renderer.

With the legacy renderer, you can create and reference a custom workflow field icon set by setting the icons attribute to a new script include. For example, to use the *WorkflowIconsSCR* script include to define which icons to use, add the attribute *icons=WorkflowIconsSCR* to the **Attributes** field of the dictionary entry for the workflow field. To use the default icon display behavior, use the attribute *icons=WorkflowIconsStages*.

The legacy renderer works with all tables except the Requested item [sc_req_item] table. In this case, use the requested item renderer instead.

Requested item

The requested item renderer functions the same way as the legacy renderer, but is for use with the Requested item [sc_req_item] table.

Only in Now Mobile, default stage renderer is used.

Select a stage renderer

In most cases, the default workflow-driven renderer should be used. If you have specific requirements for displaying stages, you can select a different stage renderer.

Before you begin

Consider the following when selecting a stage renderer:

- Use the workflow-driven renderer if possible. This is the default renderer that should be used in most cases.
- Use the legacy renderer only if your instance was upgraded from a release prior to Dublin and you want your workflow to maintain pre-Dublin behavior.

To use linear, main flow, or progress bar rendering, satisfy the following requirements.

Linear and progress bar renderer requirements

Renderer	Requirements
Linear	• The parent workflow and all subflows must modify the same current record. • In the properties for all subflows, the **Stage field** value for all subflows must match that of the parent workflow. • The parent flow and all subflows must contain the same stages. Use a stage set to ensure the parent flow and all subflows have the same stages. • On the workflow canvas, each workflow needs only the stages used directly by that workflow. It is not necessary to add stages from subflows to activities on the parent flow, or stages from the parent flow to activities on the subflows.

Linear and progress bar renderer requirements (continued)

Renderer	Requirements
Main flow	• The workflow contains subflows. • You do not want or need to reveal the details about the subflows.
Progress bar	• The workflow properties must have a **Stage ordering** value of **User-defined** • There must be workflow stages within the workflow. • Because stage names do not appear in a workflow field when using the progress bar renderer, you can simplify stage names to represent a percentage of the workflow. For example, if a process has four main steps, name the workflow stages as 25%, 50%, 75%, and 100%. Enter the numerical value of each stage, such as 25, in the **Value** and **Order** fields.

About this task

To select a stage renderer:

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open and check out the workflow.
3. In the title bar, click the menu icon and select **Properties**.
4. In the Workflow Properties dialog box, click the **Stages** tab.
5. From the **Stage rendering** list, select a stage renderer.
   - If you are using two workflows to update two unique workflow fields on a single record, both workflows must use a non-legacy renderer. You can select a different stage renderer for each workflow but do not select **Legacy** for either one.
   - If you want to use the linear renderer, make sure you select **Linear** in the properties for the parent workflow and all subflows.
6. Click **Update**.

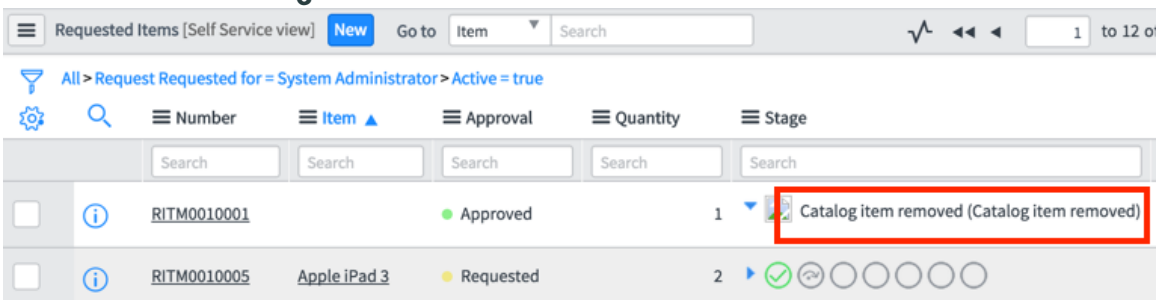
Workflow fields with deleted records

Workflow fields may indicate when a record required by the workflow is deleted.

After a referenced record is deleted, the reference in the primary record is no longer valid. If the stage renderer detects a reference that is no longer valid, the stage field displays a message about the deleted record.

Administrators can restore deleted records. For more information, see Use the Deleted Records module to restore a deleted record.

Workflows with missing records



The image shows a list with two workflows. The top request does not have an associated request item. The bottom request has an associated request item, but the item does not have an associated workflow context.

Workflow validation

Workflow validation examines different characteristics of a workflow to locate issues that might prevent the workflow from being published or cause it to fail. The validation report summarizes the results of each separate workflow validation.

Validation prevents workflows with critical flaws from executing and resulting in an unstable or incomplete state. There are a number of validators in the base system that notify workflow designers of potential problems. For example, multiple **End** activities, disconnected transitions, incorrect table references, missing subflows, and dependencies affected by update sets. A workflow validation report displays the results from each validator, including a message explaining what was found. The system automatically validates a workflow when you publish it. You can also run validation on a workflow directly from controls in the Workflow Editor.

Highlighting critical errors

The graphical Workflow Editor highlights critical errors when a workflow is loaded. In this example, a subflow is missing and is not available to the parent workflow for the current user. The graphical Workflow Editor indicates the error when the parent workflow loads by highlighting (in red) the activity that calls the subflow. To correct the error in the parent workflow, click the validate icon in the header bar and inspect the error description in the validation report.

Validate missing subflow

Validations at publishing

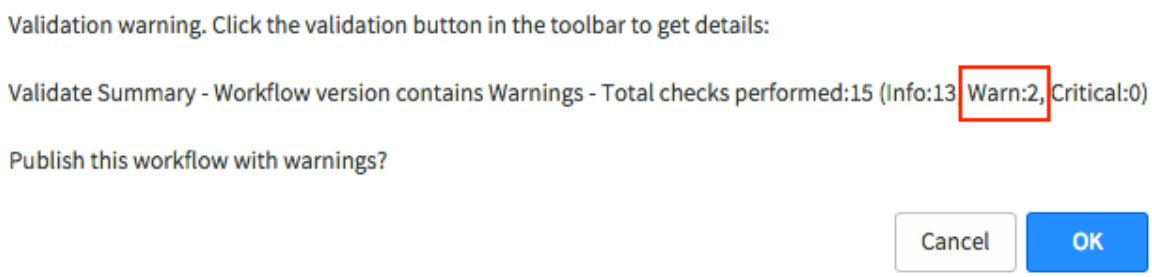
If you attempt to publish an invalid workflow or a workflow with potential problems, the system displays an error message and blocks the operation, if necessary. When validation error messages appear, click the validate icon in the graphical Workflow Editor to see the error report.

Validation warning

A validation warning notifies you that a potential problem exists in a workflow but permits you to publish the workflow. Validation warnings appear when:

- You edit and then attempt to publish a workflow that is included as a subflow in another workflow. The system cannot determine how your changes will affect the parent workflow and alerts you of the relationship.
- A workflow activity uses a different table than the table assigned to the workflow. The system alerts you of the potential conflict.

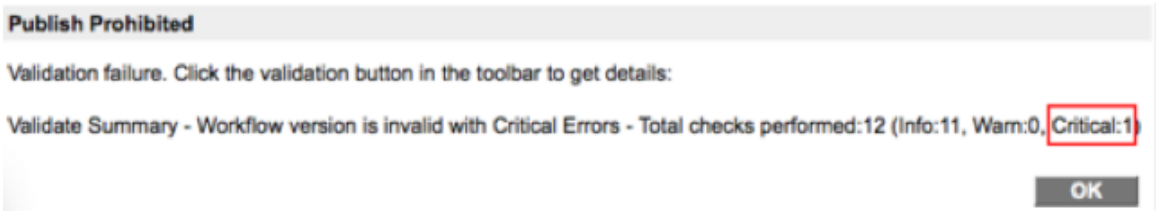
Validation warning



Validation failure

A validation failure notifies you that a critical error has occurred in the workflow that prevents you from publishing the workflow. An example of a critical error is a missing subflow.

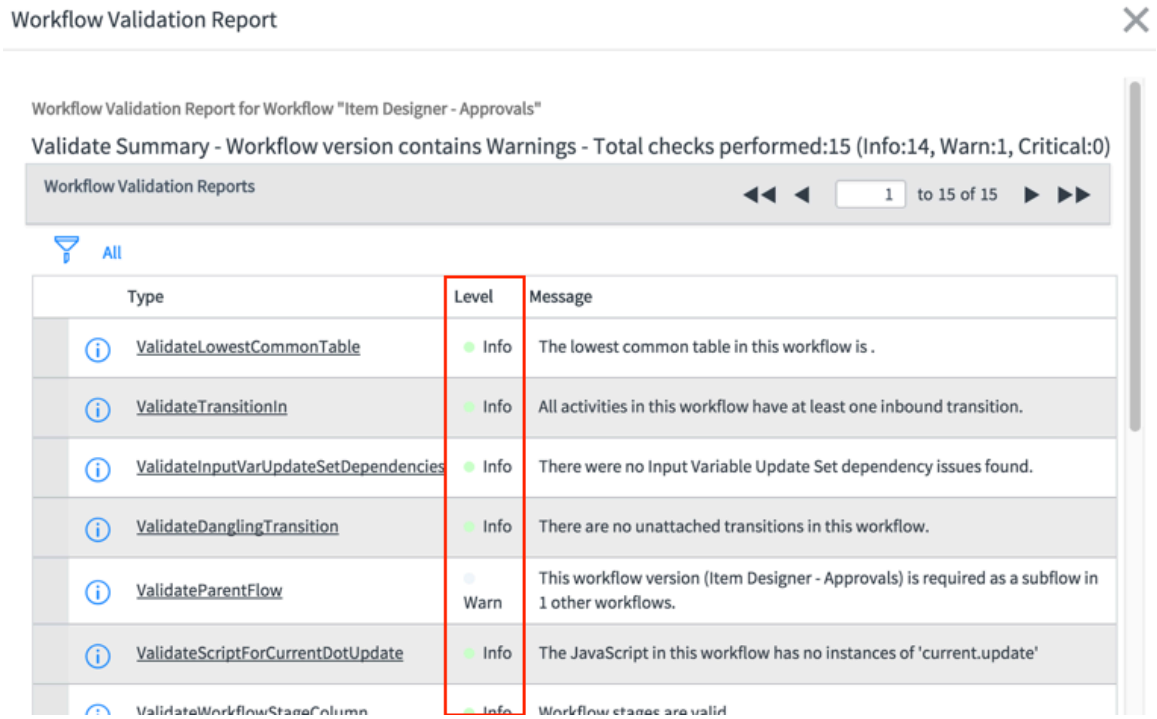
Validation failure



Workflow validation report

Validators display three notification levels: CRITICAL, WARN, and INFO. The designer can publish a workflow that returns WARN or INFO level validation, but not a workflow that returns an overall validation level of CRITICAL.

Workflow validation report



Header summary

The header of the validation report summarizes the entire validation run against the specified workflow.

- **Validate Summary:** The overall score reflects the most severe notification level encountered during the validation.
- **Total checks performed:** The total number of validations run is also broken down to show the number at each notification level.

Report columns

The body of the report displays the results of each individual validation check that was performed. The columns are **Type**, **Level**, and **Message**. You can sort and filter these columns as you would any list.

Workflow termination and external dependencies levels

Name	Implication
Info	Provides information about the current workflow version. An example of an information level message is one that names the lowest common table in the workflow. Workflows at this validation level are considered valid and publishable.
Warning	Alerts the user that the validator detected anomalies in the workflow that might compromise its ability to execute. An example of a warning level message is one that alerts you to a missing activity input transition. Workflows at this validation level are considered valid and publishable.
Critical	Names a workflow element containing a critical error that prevents the workflow from executing successfully. Examples of this are missing or invalid subflows and missing transitions. Workflows at this validation level cannot be published or run in production.

Message

The validation message provides a detailed description of the results, including table names, update sets, and other specifics.

For the procedure to validate a workflow and generate a validation report, see Validate a workflow

Workflow validator

ServiceNow offers several workflow validators for workflow designers to test their workflows.

This page lists all available workflow validators. See Workflow validation for information on using workflow validators and the workflow validation report to see the type of information that is returned.

Hanging workflows and update sets

Hanging Workflows	Update Sets
Identify workflow design decisions that can result in a hanging workflow	Identify related artifacts being moved in different update sets
ValidateTransitionOut	ValidateUpdateSetDependencies
ValidateTransitionIn	ValidateUpdateSetParentDependencies
ValidateDanglingTransition	ValidateInputVarUpdateSetDependencies
ValidateSubflows	
ValidateScriptForCurrentDotUpdate	

Workflow termination and external dependencies

Unexpected Workflow Termination	External Dependencies
Identify workflows that can end unexpectedly	Identify external artifacts that have potential workflow dependencies
ValidateSingleEnd	ValidateParentFlow

Workflow conflicts

Workflow Properties Conflicts
Identify workflow design decisions that conflict with workflow properties
ValidateLowestCommonTable
ValidateTableChange

ValidateTransitionOut

The **ValidateTransitionOut** validator finds activity conditions with no exit transitions.

Validation summary

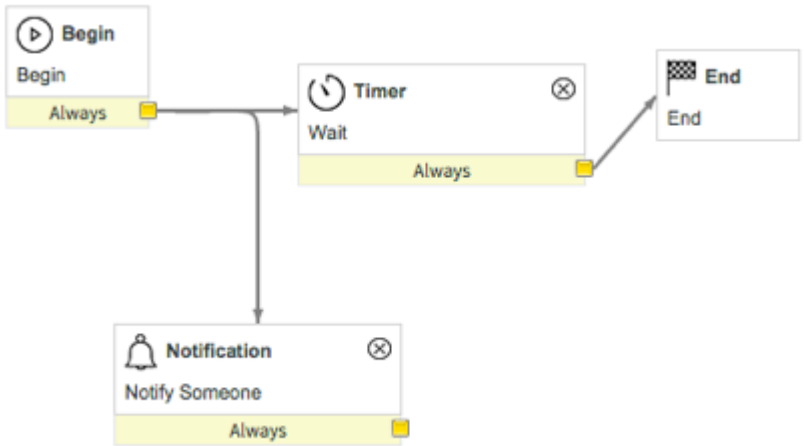
- Risk: Activity conditions might not transition to the next activity, which could cause the workflow to hang.
- Severity Level: Warning
- Valid Result: Valid

- Valid Message: All conditions have transitions.
- Invalid Result: Invalid
- Invalid Message: This workflow contains <condition count> activity conditions without an output transition.
- Suggested Action: If this is a conscious design decision, there is no corrective action. Otherwise, find the condition cited in the validator and add an appropriate transition to the next activity.
- Publishable: Yes
- Runnable: Yes
- Related Information: None

Troubleshooting

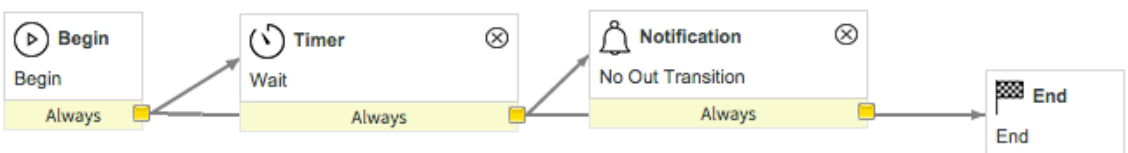
Design choices made when creating a workflow on the canvas might legitimately use an activity without an exit condition. In the first example, the **Notification** and **Timer** activities both execute at the start of the workflow. The **Timer** is the entity that decides when the workflow ends. In this situation, executing the **Notification**, but not transitioning away, keeps the design simple and adds no risk. The validator finds and reports the missing transition from the **Notification** activity as a **Warning** that the designer can ignore.

Condition with no valid transition



In the second example, the **Notification** activity has no exit transition. The designer missed this because of the layout. The transition from the **Timer** activity passes behind the **Notification** activity and appears to connect the exit from the **Notification** activity to the **End**. In workflows with more than 10 or 15 activities, it might be difficult to see all the transitions clearly. This workflow's designer intended for the **Notification** activity to transition to the **End**.

No condition out



This validator directs the designer to the specific activity and condition that does not have an exit transition. The designer then makes the decision whether or not to respond to the warning.

ValidateTransitionIn

The **ValidateTransitionIn** validator finds activities that do not have inbound transitions and cannot execute in the workflow.

Validation summary

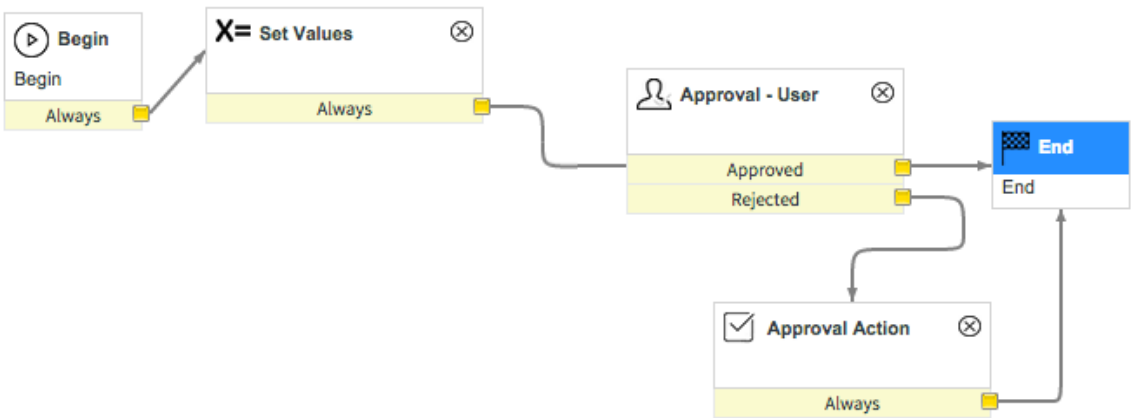
- **Risk:**Activities that do not have inbound transitions have no means of being executed in the workflow. If other logic builds from these activities, the workflow could hang, with no means of moving forward.
- **Purpose:** Find activities that do not have inbound transitions.
- **Severity Level:** Warning.
- **Valid Result:**Valid.
- **Valid Message:** All activities in this workflow have at least one inbound transition.
- **Invalid Result:** Invalid.
- **Invalid Message:**This workflow contains <activity count> activity conditions without an input transition.
- **Suggested Action:**Either remove the activities to reduce confusion and make the workflow easier to understand or provide the appropriate inbound transition.
- **Publishable:**Yes
- **Runnable:**Yes
- **Related Information:**None

Troubleshooting

Although the activities detected in this validator do no harm to the execution of the workflow, the designer needs to know that there are activities on the canvas that cannot execute and serve no purpose, particularly in a production system. This error is typically caused by a visual misinterpretation of the drawing.

This example workflow appears to be perfectly valid. Under closer inspection, however, we see that the transition from **Set Values** does not stop at **Approval - User** (there is no arrow end). Instead, that transition goes directly to **End**. As a result, **Approval - User** cannot execute. Because there is an activity in the workflow based on the approval's condition routing, it appears that the designer intended for the approval to execute and that a correction is needed.

TransitionIn invalid



This validator directs the designer to the specific activity and condition that has no inbound transition. The designer should establish a transition or remove the activity.

ValidateDanglingTransition

The **ValidateDanglingTransition** validator finds and reports any transitions that do not terminate on an activity.

Note:
These transitions are not drawn on the workflow canvas, but are still present in the database.

Warning:
This is a critical error that prevents a workflow from running.

Validation summary

- Risk: A workflow with dangling transitions will silently hang a workflow with no recovery options.
- Severity Level: Critical
- Valid Result: Valid
- Valid Message: Valid
- Invalid Result: Invalid
- Invalid Message: Invalid
- Suggested Action: Remove or connect the offending transition. Get the source activity name from the validation report details and resolve the issue. Then, run the validation again to test your changes.
- Publishable: No
- Runnable: No
- Related Information: None

Troubleshooting

On rare occasions, the destination of a workflow transition becomes null. The workflow canvas shows no evidence of the transition, but at run time, the workflow hangs when it encounters one of these dangling transitions. If the **ValidateDanglingTransition** validator reports this condition at publishing time, it blocks the publication action until the issue is resolved. If this condition is detected on a runtime check, the workflow is not allowed to execute against a current record's transaction. Instead, the system adds a critical log entry detailing the activity with the faulted transition to the current record's workflow context. To enable the workflow to execute on the next appropriate transaction, remove the faulted transition from the workflow model.

To find and remove the faulted transition:

1. Make note of the workflow version and activity that contains the faulted transition as indicated in the validator details.
2. Navigate to **Workflow** > **Administration** > **Workflow Version**.
3. In the list of workflow versions, select the workflow that has the faulted transition.
4. On Workflow Version form, add the workflow activities related list. Click the menu icon, select **Configure > Related Lists**, move **Workflow Activity-->Workflow Version** from the **Available** list to the **Selected** list, and click **Save**.
5. In the **Workflow Activities** related list, select the activity cited in the validator.

6. In the Workflow Activity form, view the **Workflow Transitions** section or tab and identify the transition in that list that has no value or a null value in the **To** column.
7. Delete this transition.
8. Return to the workflow version and re-run the validation check.

The **Critical** warning should disappear. The workflow should execute as expected on the next appropriate transaction.

ValidateSubflows

The **ValidateSubflows** validator detects any workflows included as subflows that are either inactive, deleted, or not available as a published workflow for the current user.

Any of these conditions cause the workflow to hang when the workflow activity in the main flow is encountered.

> ⚠ **Warning:**
> This is a critical error that prevents a workflow from running.

Validation summary

- Risk: A parent workflow that transitions to a deleted subflow hangs indefinitely, with no recovery options.
- Severity Level: Critical
- Valid Result: Valid
- Valid Message: This workflow contains <count> valid subflows.
- Invalid Result: Invalid
- Invalid Message: This workflow contains <invalid count> invalid subflow(s) of <total subflow count> total subflows.
- Suggested Action: Remove the link in the parent workflow to the questionable subflow, examine the subflow to ensure that it is valid and published, or that it is checked out to the current user. After making the correction to the state of the subflow, run the validation again to test your changes.
- Publishable: No
- Runnable: No
- Related Information: Workflows used as subflows

Troubleshooting

When a workflow runs, regardless of whether it is a subflow or a main flow, the script engine determines which version of a workflow should execute, given the current user and workflow conditions. When a workflow is checked out by the same user who is running the workflow, the checked out version is the version that executes. If the user is not the same person who has the workflow checked out, the published version of the workflow executes. If there is no published workflow, no workflow runs.

One scenario addressed by the **ValidateSubflows** validator is when a workflow:

- Is checked out to User A.
- Is a subflow in a parent workflow being run by User B.
- Has no published alternative to the subflow being run by User B.

When this occurs, the parent workflow runs to the execution of the unpublished subflow and then hangs at that activity, with no means to transition forward. Main flows that encounter this condition in a subflow are not permitted to execute against a current record's transaction. Instead, a critical log entry detailing the subflow's state is added to the current workflow's Workflow Context record. To correct the problem, remove the subflow from the main flow, or publish the subflow so it is available to User B. This allows the workflow to execute on the next appropriate transaction.

Another scenario addressed by the **ValidateSubflow** validator is when a workflow:

- Is a subflow in a parent workflow being run by any user.
- Has no published alternative to the subflow, because the workflow has been deleted or all versions of the workflow are unpublished or inactive.

Note:
You cannot delete a from a list or form workflow that is a subflow. However, you can create one of these unstable conditions with advanced scripting, SQL options, or incomplete update sets that contain main flows, but not the referenced subflows. When troubleshooting a workflow that triggered this validator, consider the history of the subflow while assessing the error condition.

ValidateScriptForCurrentDotUpdate

The **ValidateScriptForCurrentDotUpdate** validator finds workflow activities with scripts that use the `current.update()` function.

Calling `current.update()` causes significant performance delays in transaction processing and might cause an instance to hang.

Validation summary

- *Risk*: At best, a workflow that uses `current.update()` in scripts experiences degraded performance. In the worst case, the workflow enters an infinite, recursive loop that crashes the server.
- *Severity Level*: Warning
- *Valid Result*: Valid
- *Valid Message*: The JavaScript in this workflow has no instances of 'current.update()'.
- *Invalid Result*: Invalid
- *Invalid Message*: This workflow uses 'current.update()' in <count of current.update references> JavaScript statements.
- *Suggested Action*: Remove `current.update()` from the activity scripts cited by this validator. Workflows execute within a transaction, and current is updated, or possibly inserted, at the end of the transaction, as appropriate. There is no need to explicitly update the record during the transaction.
- *Publishable*: Yes
- *Runnable*: Yes

Problems with current.update() in workflow scripts

A workflow initiates execution in one of these ways:

- *Script Engine*: If a workflow is assigned to a specific table, and given a run condition, the workflow runs on INSERT.
- *Script*: Any business rule, script include, background script, or client script can initiate a workflow using the workflow script include and calling `startFlow()`.

The workflow engine initiates a workflow based on the matched criteria of the current record being inserted. The transaction for current is managed by the script engine and not the workflow. Workflows that progress on the `update()` of the current record are not invoked via the workflow engine, but as a call from either a client script or business rule. In either case, the script engine is invoked, and the current record is put in memory. Edits and modifications to any current fields are made and are available to other activities and scripts that are executed in the same transaction.

When appropriate, other engines that run in sequence with the workflow engine, such as the business rule engine or field normalization, are invoked against the same current record transaction. Any changes made to current through these scripts and activities modify the record in memory. These changing values are available for reference in any other transactions called from activities and scripts in the same INSERT transaction. When all expected changes are executed, the current record is inserted.

When one of these scripts calls `current.update()` on a record that has yet to be inserted, the action forces an unnecessary and error-prone database transaction. If a record is not yet in the database, it cannot be updated. Business rules that trigger on `update()` on a record that is in the process of being inserted can cause a very unstable and potentially infinite looping condition.

Troubleshooting

This validator detects the use of `current.update()` in any of the editable script fields. Do not call `current.update()` from within a workflow script. In the event of an INSERT or UPDATE of current, the changes made to current are available to all scripts executing in the same transaction, and the script engine stores all changes in the database. Leave the update of current to the engine. Use the scripts only for setting and referencing the current field values.

ValidateLowestCommonTable

The **ValidateLowestCommonTable** validator reports the lowest table in the Glide hierarchy that the workflow uses.

For example, the Requested Item [sc_req_item] table is the lowest table in a workflow containing a **Catalog Task** activity. This information is significant to a designer who wants to change the table against which an existing workflow runs after adding activities to the canvas.

Note:
This validator provides information only. It does not indicate an error or warning condition.

Validation summary

- Risk: This validator informs only and has no risk associated with it.
- Severity Level: Data/Information
- Valid Result: Valid
- Valid Message: The lowest common table in this workflow is <<table_name>.
- Invalid Result: N/A (informational only)
- Invalid Message: N/A (informational only)
- Suggested Action: None

- Publishable: Yes
- Runnable: Yes
- Related Information: Workflow activities

ValidateTableChange

The **ValidateTableChange** validator reports any activities in the workflow that are invalid given the table associated with the workflow version.

For example, a workflow version that is associated with the Change Request [change_request] table but has a **Catalog Request** activity on the canvas is invalid, since the activity is not compatible with the selected table.

Validation summary

- Risk: If the current record at runtime does not originate from the table specified by the lowest common table, the activities for the lowest common table cannot set specific values.
- Severity Level: Warning if the table that is associated with a workflow is higher in the table hierarchy than the lowest common table required for the workflow activities.
- Valid Result: Valid
- Valid Message: All activities are valid for the newly selected table
- Invalid Result: Invalid Activity
- Invalid Message: This workflow contains <count of invalid activities> invalid activities for the newly selected table.
- Suggested Action: Make one of these changes:
  - Change the workflow to not require the activities associated with the lowest table reported.
  - Modify the workflow to use a table that contains the lowest common table in its hierarchy.
  - Ensure that the current record meets the requirements of the at-risk activities.
- Publishable: Yes
- Runnable: Yes
- Related Information: Workflow activities

Troubleshooting

Navigate to **Workflow** > **Administration** > **Activity Definitions**. Note the **Table** column in the list. Each activity that is not global is associated with a table.

When you select a table in the Workflow Properties form, the activity menu presents only activities that are compatible with the selected table. Associating a table with a workflow activity enables the system to make certain decisions about the activity that affect value comparison, condition routing, or the setting and getting of field values in the current record.

When a workflow is set to the Global [global] table, it is possible that the process executing the workflow ensures that the current record meets the requirements of the activities identified by this validator. In that case, the activity still works as expected. If that assurance cannot be guaranteed by the user process, do not use the activity identified by the validator without assigning a table that meets the requirements of all the activities on the canvas.

This validation check ensures that the table specified by all the activities in a particular workflow is included in the hierarchy of the table selected for the workflow.

ValidateParentFlow

The **ValidateParentFlow** validator reports any workflows that use the workflow as a subflow.

Validation summary

- **Risk:** There is no risk in a workflow being a subflow. This is only a warning that other workflows are at risk from dramatic changes to a subflow.
- **Severity Level:**Warning
- **Valid Result:** Valid
- **Valid Message:** Currently <workflow version name> is not a subflow.
- **Invalid Result:** None
- **Warning Message:**This workflow version (<workflow version name>) is required as a subflow in <workflow version count> other workflows.
- **Suggested Action:** Exercise caution when modifying things like input parameters and return values to assure that parent workflows are not adversely affected.
- **Publishable:** Yes
- **Runnable:**Yes
- **Related Information:** Workflows used as subflows

Troubleshooting

The system warns the workflow designer at publishing time and during validation that a workflow is a subflow. This warning reminds the designer that changes to the current workflow have the potential to affect workflows cited in the validation report or other workflows already running in production. Regardless of how simple the change to a workflow that is a subflow, thoroughly test all parent flows cited in this validation report before publication.

When a workflow is a subflow, changes that can cause it to become invalid include:

- Changing the data types of input variables. Verify that all parent workflows cited in this validation report can pass the correct value type.
- Adding input variables. Verify that all parent workflows cited in this validation report are able to pass all variable values into the subflow.
- Changing or removing the return value of a workflow. Verify that any changes to the return value of a workflow are compatible with the requirements of all the parent flows.
- Changing the table on which the workflow runs. Verify that the table selection is compatible with all parent flows.

Note:
To delete a workflow that is a subflow, first remove the dependency by removing the subflow from all parent flows cited in this validation report. After the dependencies are cleared, a user with the proper role can delete the subflow.

ValidateSingleEnd

The **ValidateSingleEnd** validator finds and identifies multiple **End** activities in a single workflow.

Multiple **End** activities in a workflow might be intentional and have no affect on the workflow, or might be a mistake that the designer needs to correct.

Validation summary

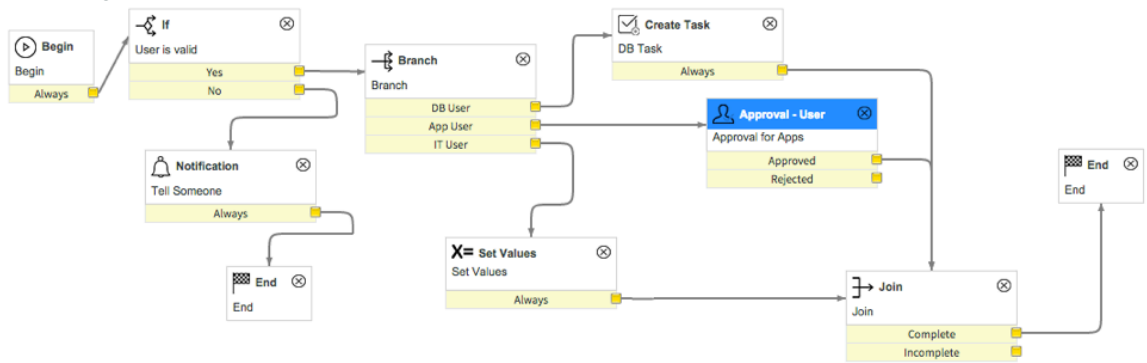
- Risk: If the execution paths to the **End** activities are not mutually exclusive, then the first **End** encountered completes the workflow and cancels all other executing activities.
- Severity Level: Warning
- Valid Result: Valid
- Valid Message: This workflow contains 1 End activity.
- Invalid Result: Invalid Activity
- Invalid Message: This workflow contains <count of ends> End activities.
- Suggested Action: Remove extraneous **End** activities that are not intended as part of the design.
- Publishable: Yes
- Runnable: Yes
- Related Information: None

Troubleshooting

As soon as an **End** activity is encountered in the workflow, the workflow completes even if there are other viable execution paths leading to a second **End** activity that is still executing. Those executing activities are canceled as part of the **End** activity's clean up actions. Therefore, the results of designing workflows with multiple **Ends** must be carefully considered.

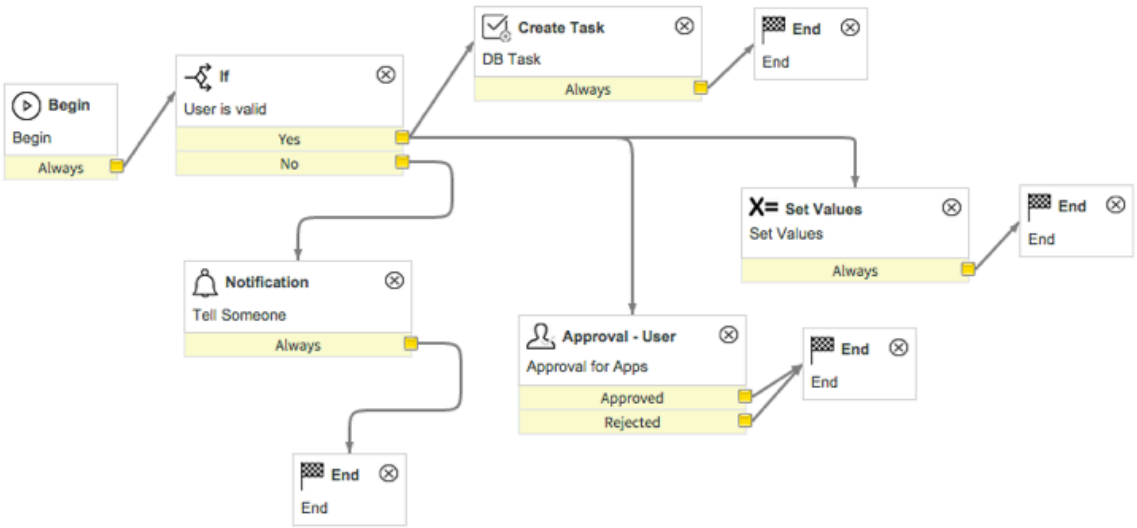
In the case of large workflows, it is often more intuitive to read the workflow when there are multiple **End** activities. In the following example, the paths to the two **Ends** are mutually exclusive execution paths. If this was a large workflow, with many activities between **Branch** and the second **End**, the value of the multiple ends becomes apparent. Tracing a **No** response from **User is invalid** to a single **End** behind 33 other activities would be significantly more difficult. There is no risk in this workflow design because there is no reason for other activities to execute if the **End** after the **Notification** activity terminates the workflow.

Mutually exclusive execution paths



The next example has multiple **End** activities in execution paths that are not mutually exclusive. A **Yes** response from **User is valid** causes the **Set Values** activity to finish immediately. By reaching its **End** activity first, this execution path cancels the **Approval for Apps** and the **DB Task** activities, which might not be the desired outcome. If the paths are all expected to complete before **End**, the activities should come to a **Join** (as in the previous example) that transitions to a single **End**.

Mutually inexclusive execution paths



Note:
To add the second **End**, right-click to copy the original **End** activity and paste it onto the canvas. In most cases, a single **End** is the best and most reliable way to ensure that all activities expected to execute prior to workflow completion, do so successfully.

ValidateUpdateSetDependencies

The **ValidateUpdateSetDependencies** validator identifies all the subflows called in the current workflow and determines if any of those subflows are being edited in a different (in progress) update set.

This warning informs the user that this workflow and one or more of its dependencies are being actively modified in a way that will not deploy concurrently to another instance without additional effort.

For information about update sets, see Create and select an update set.

Validation summary

- Risk: If a parent workflow is edited in one update set and its dependent subflow is edited in another, the two workflows might not be compatible when moved to a different instance. Making independent changes, such as to common or expected values, can make the two workflows incompatible.
- Severity Level: Warning
- Valid Result: Valid
- Valid Message: There were no Update Set dependency issues found.
- Invalid Result: Invalid
- Invalid Message: This workflow has dependent workflows that are in a different update set.
- Suggested Action: Modify and deploy both workflows in the same update set. If you must modify dependencies in separate update sets, use one of these methods:
  - Ensure that all update sets migrate concurrently.
  - Prior to deploying the main flow update set, merge the dependencies into one update set before completing that update set.
- Publishable: Yes

- Runnable: Yes
- Related Information: Workflow movement with update sets

Troubleshooting

A workflow is added to an update set only when the workflow is published. This validator issues a warning when either of the following conditions exist:

- A published subflow is in a different update set than the parent workflow and that update set is In progress.
- A subflow is checked out by another user, who is working in a different update set than the current user.

Note:
This validator does not look for update sets that have been closed. It looks only at update sets that are In progress or at the update sets of all subflows being used by the current workflow that are checked out to users who are working in a different update set.

Example

Following is an example of an at-risk development scenario in which two users create dependencies between workflows in different update sets.

User A:

1. Sets Update Set A to the current update set.
2. Checks out Workflow A.
3. Changes the return value of the **String** type in Workflow A to a **Reference/User** type.
4. Publishes Workflow A, causing an entry into Update Set A.

User B:

1. Sets Update Set B to the current update set.
2. Checks out Workflow B.
3. Includes Workflow A as a subflow.
4. Uses the user reference return value from Workflow A as an approval assignment.
5. Publishes Workflow B, causing an entry into Update Set B.

Risks

- User B moves Update Set B to a different instance that has an older version of Workflow A. The return value is not a user reference, which causes the outcome of Workflow B to be different than it was when tested in development.
- User B moves Update Set B to a new instance that does not have a version of Workflow A. Workflow B experiences a validation failure at runtime and cannot execute. A log entry is added to the workflow log of the current record.

Possible solutions

Solution 1

Migrate the parent workflow and all dependent workflows to a new instance together using the same update set.

1. Set the update set to the one you want to migrate to new instances.
2. Check out and republish the workflows that need to be included, this action forces an entry into the current update set.
3. Complete the update set with all dependencies.
4. Follow standard procedures for migrating update sets to local instances.

Solution 2

Move dependent workflows between update sets.

1. Identify the update set containing the main workflow to be migrated.
2. Navigate to **System Update Sets > Local Update Sets**.
3. Find and select the update set that contains the dependencies to the main workflow.
4. In the **Customer Updates** related list, select the workflow version of the subflow you want to move.
5. Select the update set containing the parent workflow in the **Update set** field. If this field is not on the Customer Update form, configure the form and add the field.
6. Click **Update** and the base system moves the dependent subflow to the update set selected.
7. Repeat steps 4-6 to add additional dependent subflows to the parent flow update set.

ValidateUpdateSetParentDependencies

The **ValidateUpdateSetParentDependencies** validator identifies all the workflows that call the current workflow as a subflow and determines if any of those parent workflows are being edited in a different update set that is in progress.

This warning informs the user that this workflow and one or more workflows that depend on this workflow are being actively modified in a way that will not deploy concurrently to another instance without additional effort.

Validation summary

- Risk: If a parent workflow is edited in one update set and its dependent subflow is edited in another, the two workflows might not be compatible when moved to a different instance. Making independent changes, such as editing common or expected values, can make the two workflows incompatible.
- Severity Level: Warning
- Valid Result: Valid
- Valid Message: There were no Update Set dependency issues found.
- Invalid Result: Invalid
- Invalid Message: This workflow has dependent workflows that are in a different update set.
- Suggested Action: Modify and deploy both workflows in the same update set. If you must modify dependencies in separate update sets, use one of these methods:
  - Ensure that all update sets migrate concurrently.
  - Prior to deploying the main flow update set, merge the dependencies into one update set before setting that update set to complete.
- Publishable: Yes

- Runnable: Yes
- Related Information: Workflow movement with update sets

Troubleshooting

A workflow is added to an update set only when the workflow is published. This validator issues a warning when either of the following conditions exist:

- A published subflow is in a different update set than the parent workflow and that update set is In progress.
- A subflow is checked out by another user, who is working in a different update set than the current user.

Note:
This validator does not look for update sets that have been closed. It looks only at update sets that are In progress or at the update sets of all parent workflows that call the current workflow and are checked out to users who are working in a different update set.

Example

Following is an example of an at-risk development scenario in which two users create dependencies between workflows in different update sets.

User A:

1. Sets Update Set A to the current update set.
2. Checks out Workflow A.
3. Changes the return value of the **String** type in Workflow A to a **Reference/User** type.
4. Publishes Workflow A, causing an entry into Update Set A.

User B:

1. Sets Update Set B to the current update set.
2. Checks out Workflow B.
3. Includes Workflow A as a subflow.
4. Uses the user reference return value from Workflow A as an approval assignment.
5. Publishes Workflow B, causing an entry into Update Set B.

Risks

- User B moves Update Set B to a different instance that has an older version of Workflow A. The return value is not a user reference, which causes the outcome of Workflow B to be different than it was when tested in development.
- User B moves Update Set B to a new instance that does not have a version of Workflow A. Workflow B experiences a validation failure at runtime and cannot execute. A log entry is added to the workflow log of the current record.

Possible solutions

Solution 1

Migrate the parent workflow and all dependent workflows to a new instance together using the same update set.

1. Set the update set to the one you want to migrate to new instances.
2. Check out and republish the workflows that need to be included.

   Note:
   This action forces an entry into the current update set.

3. Complete the update set with all dependencies.
4. Follow standard procedures for migrating update sets to local instances. For information about update sets, see System update sets.

Solution 2

Move dependent workflows between update sets.

1. Identify the update set containing the main workflow to be migrated.
2. Navigate to **System Update Sets** > **Local Update Sets**.
3. Find and select the update set that contains the dependencies to the main workflow.
4. In the **Customer Updates** related list, select the workflow version of the subflow you want to move.
5. Select the update set containing the parent workflow in the **Update set** field. If this field is not on the Customer Update form, configure the form and add the field.
6. Click **Update**. The base system moves the dependent subflow to the update set selected.
7. Repeat steps 4-6 to add additional dependent subflows to the parent flow update set.

ValidateInputVarUpdateSetDependencies

The **ValidateInputVarUpdateSetDependencies** validator examines update sets to ensure that workflow input variables for a given workflow have not been deleted in different update sets than those currently **In progress**.

Validation summary

- Risk: Workflows and their input variables are not moved together in a single update set. The deletion of input variables is captured in a different update entry. If these two entries are not in the same update set, the workflow execution can be unstable.
- Purpose: Determine whether input variables that belonged to this workflow were deleted in a different update set.
- Severity Level: Warning
- Valid Result: Valid
- Valid Message: There were no Input Variable Update Set dependency issues found.
- Invalid Result: Invalid
- Invalid Message: There are input variables that have been deleted and logged in a different update set.
- Suggested Action: If the deletion is not intended to be separate from the workflow, ensure that both update sets are committed concurrently to the new instance, or merge both payloads into a single update set.
- Publishable: Yes

- Runnable: Yes
- Related Information: Workflow movement with update sets

Troubleshooting

Workflow input variables get individual entries in the **Customer Update** related list in the current user's update set. This validator reports to the user when workflow input deletions have happened in an update set other than the current user's update set.

Follow the instructions for Input variable removal when the validator issues this warning.

ValidateWorkflowEndStages

The **ValidateWorkflowEndStages** validator checks that in workflows with stages, the end activity of the workflow has a stage named Complete or Completed.

If the workflow has stages associated with it, but does not have the completed stage on the end activity, then the stage indicator will not show that the workflow completed.

Validation summary

- Risk: The stage indicator will not show the workflow is completed.
- Severity Level: Warning
- Valid Result: Valid
- Valid Message: Workflow end stages are valid.
- Invalid Result: invalid
- Invalid Message: End activity A found with invalid stage "S."
- Suggested Action: If this is not by design, make the appropriate changes.
- Publishable: Yes
- Runnable: Yes
- Related Information: Workflow activities

Troubleshooting

Check the workflow for an end activity. Ensure that this activity assigns a stage named Complete or Completed.

ValidateWorkflowStageColumn

The **ValidateWorkflowStageColumn** validator detects and reports when the stage field (stage column) for a workflow is not correct or is unusable.

Validation summary

- Risk: The stage indicators may not display appropriate information.
- Severity Level: Warning
- Valid Result: Valid
- Valid Message:Workflow stage values are valid.
- Invalid Result: Warning
- Info Summary: Stage warnings found.
- Invalid Messages:

- Table T does not have a column named C.
- Workflow for table T has stages, but no stage column.
- Table T has a stage column "C," but no stages are set by activities.
- Workflow on table T has stage column "C" that is not type=workflow.

- Suggested Action: If this is not by design, make the appropriate changes.
- Publishable: Yes
- Runnable: Yes
- Related Information: Workflow activities, Workflow stages

Troubleshooting

To check the stage column:

1. Open and check out a workflow.
2. Open the workflow version properties dialog by clicking the menu icon and selecting **Properties**.
3. View the **Stages** tab or section.
4. Check that the assigned stage column is actually a column in the table to which the workflow is associated.
5. Check that the column is type=workflow.

Tip: Stage columns should not be choice lists. If they are, the list appears read-only in form views, since changing that column value outside the workflow engine does not ensure safe tracking of stage states.

ValidateWorkflowStateValues

The **ValidateWorkflowStateValues** validator checks a number of stage aspects in workflow activities for correctness.

This validator has multiple possible error messages.

Validation summary

- Risk: The stage indicators may not display appropriate information.
- Severity Level: Warning.
- Valid Result: Valid.
- Valid Message: Workflow stage values are valid.
- Invalid Result: invalid.
- Warning summary: Stage warnings found.
- Invalid Messages:
  - Stage with empty name is not allowed.
  - Stage with empty value is not allowed.
  - Cannot have more than one stage with the same name: x.
  - Cannot have more than one stage with the same value: y
- Suggested Action: If this is not by design, make the appropriate changes.
- Publishable: Yes.

- Runnable: Yes
- Related Information: Workflow activities.

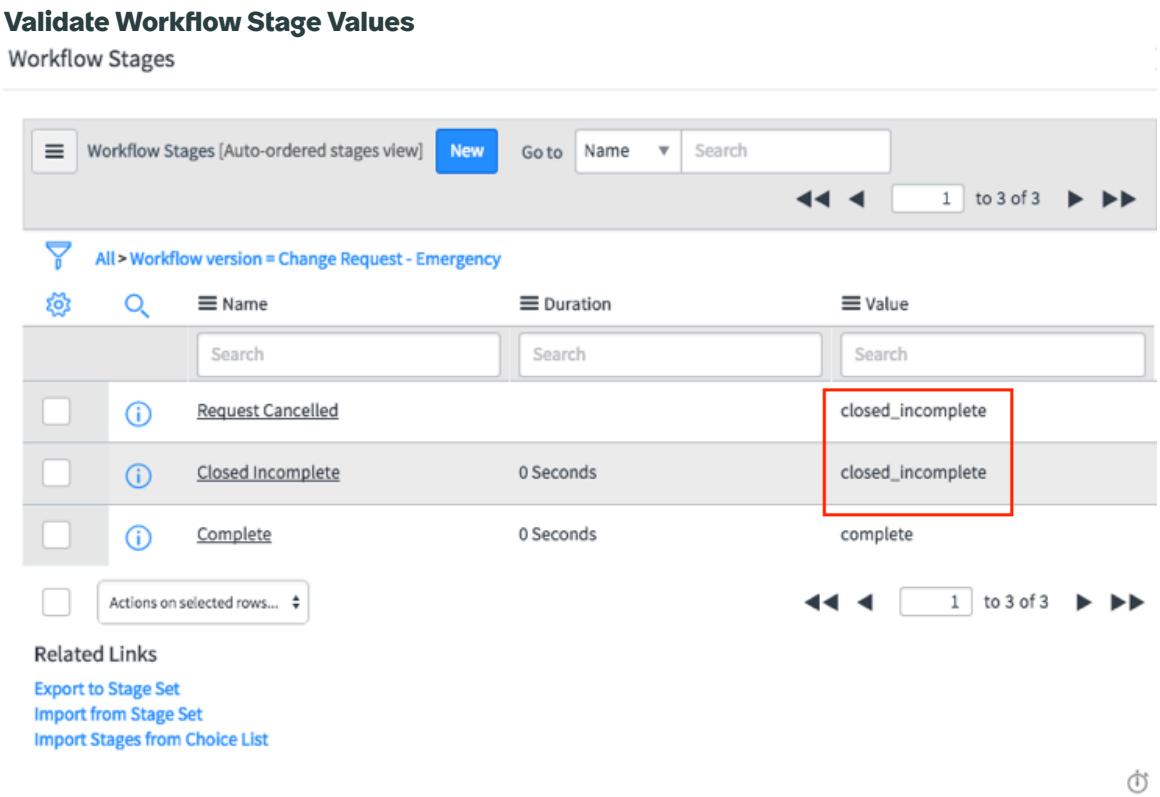
Troubleshooting

Use the following procedure to troubleshoot this validator:

1. Open and check out a workflow.
2. Open the list of stages for the workflow by selecting the menu icon and selecting **Edit Stages**.
3. Check the names and values.

   Ensure that the names and values are unique and are not empty.

4. If the same stage name or value apprears more than once, remove one of the rows. It is then very important to go through the workflow and reassign stages in the activities that used the removed stage.



Tip: Make a list of which activities assign which stages.

Workflow concepts

You can do many things using the Workflow Editor.

- Modify core activities and exit conditions.
- Create custom activities and reuse the data for other workflows.

  See Orchestration activity designer.

- Download activity packs from the ServiceNow Store and create packs for upload.
- Edit workflows graphically.

- Define transitions between workflow activities.
- For the table that corresponds to the workflow, customize business rules.
- Summarize workflow progress through stages.
- Validate workflows to identify potential problems.
- Publish workflows for other users.
- Edit multiple tables without needing to directly modify them.

Workflow versions

To prevent users from making changes to a workflow that affect other users of the system, workflows must be checked out before they can be edited.

Only one user can check out a workflow at a time. When a workflow is checked out, changes apply only to the user who has the workflow checked out. Other users can continue to use the published workflow. After the changes are complete, the workflow can be published so that it is available to all users.

Note:
Because each workflow has a unique sys_id, different workflows can have the same name. This is typically expected in a domain-separated environment where users in different companies cannot see each other's workflows because they are in different domains. However, this can lead to confusion in other environments. In general, give each workflow a unique name to prevent workflow designers from making changes to the wrong workflow.

When a new version of an existing workflow is published, the changes are not applied to running workflow contexts. Any currently running workflow context continues using the workflow version that was available when the workflow started. The next time the workflow runs, it uses the updated, published version.

Workflow scope

Workflow application scope determines the access that an application has to the information in a workflow, specifically to the data contained in the activities in that workflow.

When a workflow is created, it inherits the application scope from the gear menu for the logged in user. This scope cannot be changed in the Workflow Editor. When the workflow executes, it runs in this scope and can only be called from a different application if the workflow's accessibility setting permits access to all scopes (public). Otherwise, the workflow's application scope is private to the application.

Note:
Any script that is created in the Workflow Editor, such as an advanced script in an *If* activity, runs in the scope of the workflow. All core activities provided in the base system or for Orchestration run in the scope of the workflow.

Custom activities run in their own scope, even if it is different from that of the workflow. The scope of a custom activity can be private or public. Any script that runs inside a custom activity with a scope can only access outside artifacts that are within the scope of that activity or artifacts that are configured to run in any scope. Conversely, an outside artifact can only access the script inside that private activity if the outside artifact is running in the same scope. Activities with public scopes can interact with outside artifacts in any application scope.

You can use private activities as part of a workflow that has a public application scope. These activities are protected from reaching outside of the workflow or from being reached from outside

the workflow. For details about setting application scoping for custom activities, see the field description table for the appropriate activity template.

Note:
Custom activities uploaded to the ServiceNow Store must be configured as accessible to all application scopes.

Workflow scope restrictions

There are some restrictions to public and private application scopes.

During runtime, publicly scoped workflows can access other application resources, as long as these resources are set to be accessible to all application scopes. Privately scoped workflows in a private application scope can only access resources private to its scope. Due to scope access boundaries, any privately scoped workflows that make calls out to other scoped resources fail with either an exception or a hung activity while waiting for returned results. This occurs when making calls to these common global resources:

- ECC queues
- Tasks
- Approvals
- Events
- SLA timers
- Timers
- Script includes
- Business rules
- Workflow APIs

As you design workflows, validate the visibility and accessibility of all resources prior to deployment.

See Application scope.

For information on how to configure the scope for a workflow, see Workflow properties.

Domain separation and Workflow

Domain separation is supported in the Workflow application. 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 **Basic** level
- Business logic: Processes can be created or modified per customer by the service provider. The use cases reflect proper use of the application by multiple service provider customers in a single instance.
- The owner of the instance needs to be able to configure MVP business logic and data parameters per tenant as expected for the specific application.

Use case: As an admin, I need the ability to make comments mandatory on close of a record for one tenant, but not for another.

Overview of domain separation and Workflow

When domain separation is enabled, workflows and workflow activities inherit the domain of the user who publishes or creates them.

How domain separation works in the Workflow application

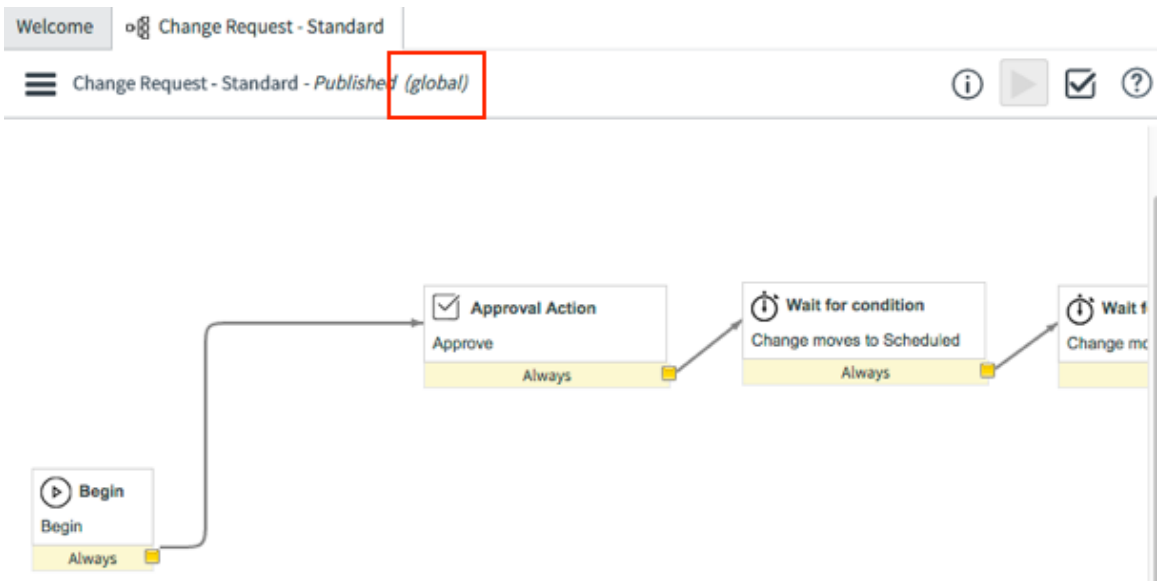
While workflows are managed by multiple tables, only the following tables are used for domain separation features:

- **Workflow [wf_workflow]** and **Workflow Version [wf_workflow_version]**: used for Process administration or process separation.
- **Workflow Context [wf_context]**: used for Understanding domain separation.

Note:
The Workflow Version table [wf_workflow_version] table does not contain a domain field; Workflow Version records inherit their domain from the parent Workflow record.

The Workflow Editor displays a workflow's domain in the title bar after the workflow name.

Workflow editor domain



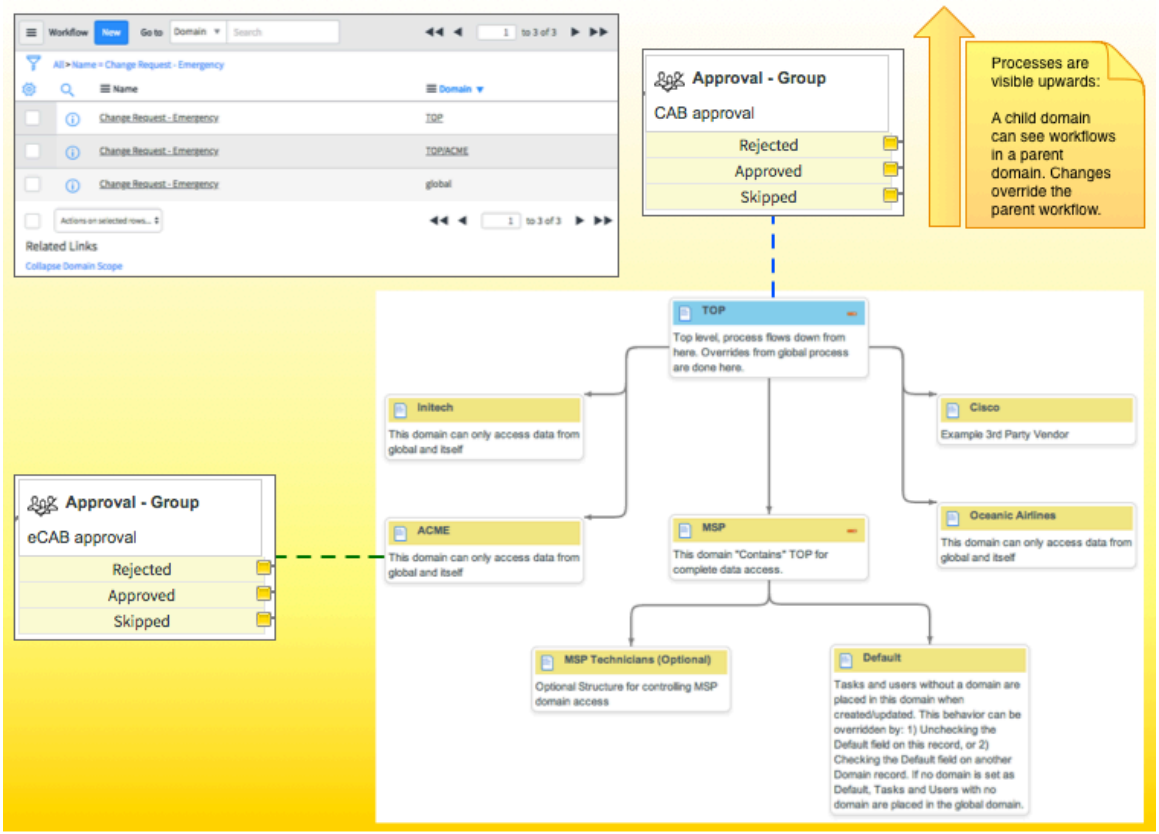
Related topics

Domain separation for service providers

Workflows and delegated administration

Delegated administration allows child domains to inherit workflows from higher up the domain hierarchy and to override them with domain-specific versions if necessary.

Workflow and delegated administration



Workflow records in the Workflow [wf_workflow] and Workflow Version [wf_workflow_version] tables are considered processes. A user in a child domain may check out but not copy a workflow from a parent domain. When a user in a child domain checks out a workflow from a parent domain, the system creates a version of the workflow in that user's domain. This new version is a unique record in the Workflow [wf_workflow] table. After the user publishes this new workflow, other users in the child domain use the new workflow, which overrides the workflow from the parent domain. The original workflow in the parent domain is no longer visible to users in the child domain.

For example, a managed service provider (MSP) hosts ITSM services for several companies, including ACME and Initech, on a single instance. As administrators, the MSP creates a Change Request - Emergency workflow that applies to all domains because it was created in the TOP domain, which is the highest domain in the domain hierarchy. This workflow overrides the global Change Request - Emergency workflow and specifies that emergency change requests require approval from the CAB approval group. Because of delegated administration, every domain in the hierarchy sees and uses this workflow. Now suppose the ACME domain requires a different approval policy where emergency change requests require approval from the emergency CAB approval group. The MSP creates another version of the Change Request - Emergency workflow in the ACME domain. This workflow overrides the version in the TOP domain and only applies to users in the ACME domain.

Workflow permissions

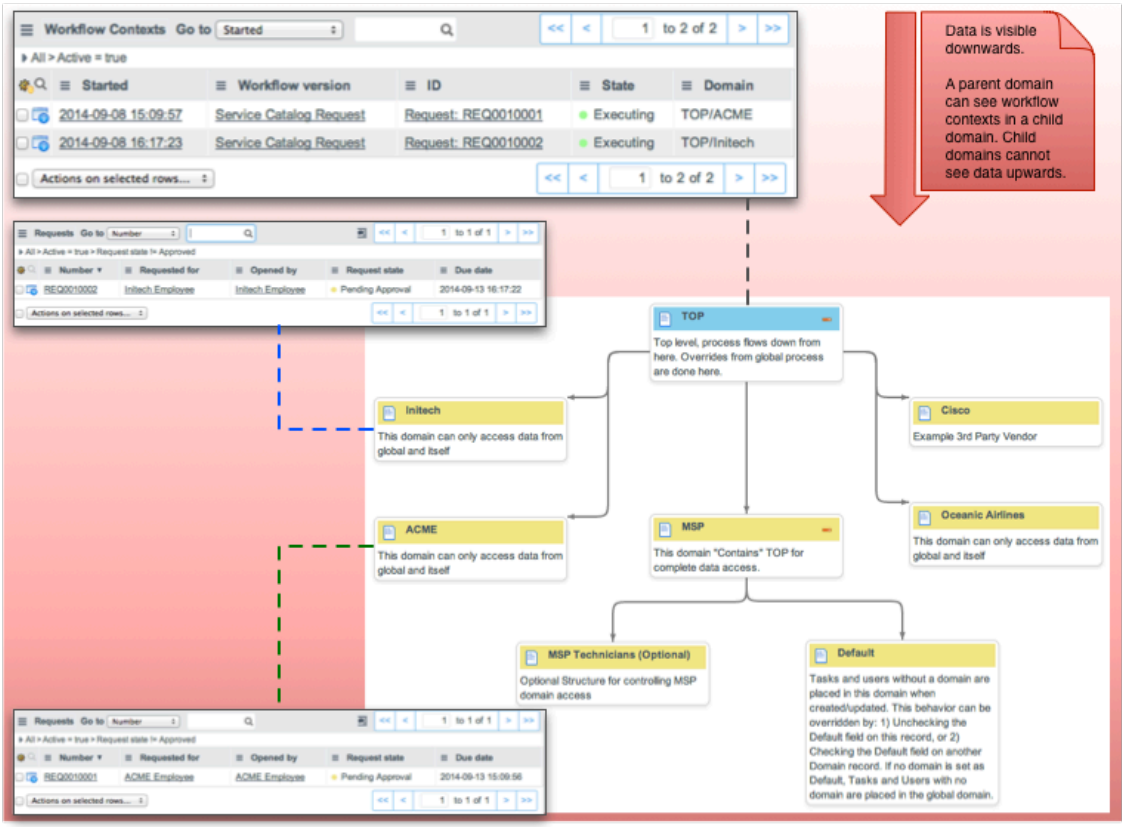
When a user starts a new workflow, the workflow runs with that user's domain and credentials.

The workflow preserves a user's domain and credentials until an activity causes the workflow to wait, such as an approval activity waiting for approval or rejection. When the stopped workflow resumes, such as when a user approves a request, the workflow uses the credentials of the approving user, but continues to run within the domain of the original user.

Workflows and data separation

Data separation restricts workflow contexts to users who are either in the same domain of the workflow or are members of a parent domain.

Workflow and data separation



Workflow records in the Workflow Contexts [wf_contexts] table are considered data. Data separation restricts workflow contexts to users who are either in the same domain of the workflow or are members of a parent domain. While a user in a parent domain can see running workflows in a child domain, a user in a child domain cannot see running workflows in a parent domain. If necessary, administrators can use visibility or contains domains to expand who can see domain-specific data.

For example, when an ACME user requests something from the service catalog, a Service Catalog Request workflow context is created in the ACME domain. Similarly, a service catalog request from an Initech user creates a workflow context in the Initech domain. An MSP user in the TOP domain can see both workflow contexts because it is the parent domain for both the ACME and Initech domains. However when an ACME or Initech user logs in, data separation prevents them from seeing each other's service catalog requests. This is expected behavior because each workflow context contains data specific to that domain, such as the item requested and the request's approval history.

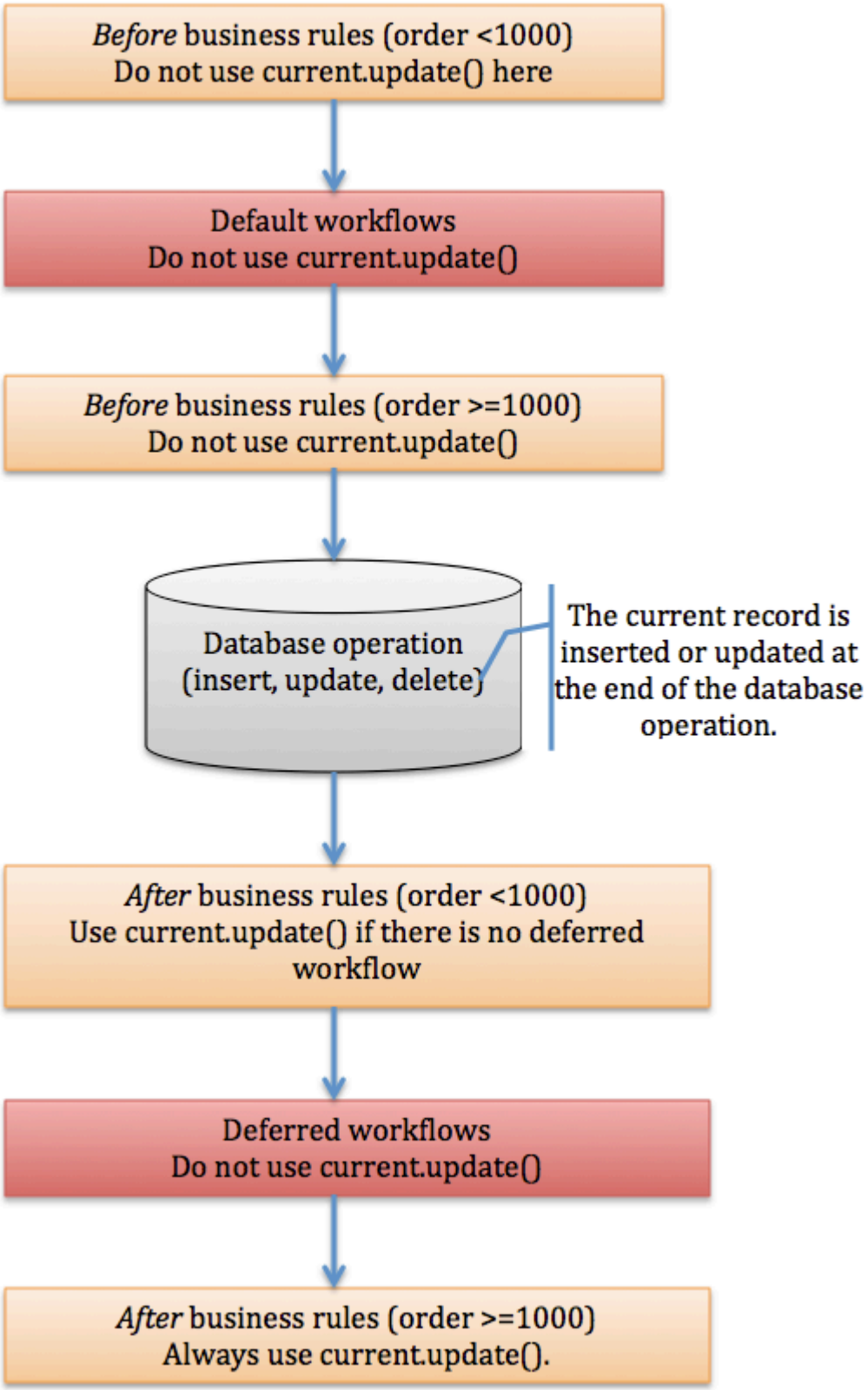
Workflow engine operation order

The workflow engine runs in a predefined order relative to business rules and database operations. It caches commonly-used published workflows to improve performance.

The **Run after bus. rules run** workflow property defines if a workflow is **Default** or **Deferred**.

The diagram below shows the workflow engine order of operations and when Default and Deferred flows are executed. For a more general overview of engine operation order, see Execution order of scripts and engines.

Workflow engine order diagram



Workflow caching

The workflow engine caches commonly-used published workflows to improve performance.

Caching significantly reduces the number of database queries per workflow. By default, the engine caches up to 300 unique workflow versions. Caching very large workflows may reduce this number as the cache size cannot exceed the Java Virtual Machine (JVM) heap size.

To change the maximum number of cached workflow versions, navigate to **Workflow** > **Administration** > **Properties** and modify the value of the **The max number of models that will be concurrently held in the LRU cache** (*glide.workflow.model.cache.max*) property. You must restart the instance to apply this change.

Workflow tables

For full flexibility, workflows store information over a number of different tables.

Usually tables containing workflow information are not edited one-by-one. Instead, use the Workflow Editor to edit workflows. The following lists are provided for reference purposes.

Workflow tables

Table	Description
Workflows	
Column Renderer [column_renderer]	A renderer widget for a stage column. Stage renderers are written in Jelly as a UI Macro. The default is Workflow-Driven; it covers most workflow related stage scenarios.
Version [wf_versionable]	Tracks different versions of element definitions [wf_element_activity].
Workflow [wf_workflow]	The primary records of workflows.
Workflow Binding [wf_workflow_binding]	History of workflows run and the triggering record. Workflow Binding records prevent the system from running workflows again when the associated Workflow Context record has been deleted.
Workflow Context [wf_context]	Individual instances of a workflow being used.
Workflow Execution [wf_workflow_execution]	Synthetic "current" records for workflows that run on Global.
Workflow Instance [wf_workflow_instance]	Connections of workflows to subflows.
Workflow Version [wf_workflow_version]	Particular versions of a workflow, either published versions or versions that have been checked out.
Activities	
Activity Designer [wf_element_activity]	Custom activity definitions.
Activity Variables [wf_activity_variable]	Variables for activities.
Workflow Activity [wf_activity]	Activities as they are being used in workflows.
Workflow Activity Definition [wf_activity_definition]	Definitions of activities that can be used in a workflow.

Workflow tables (continued)

Table	Description
Workflow Executing Activity [wf_executing]	Individual instances of activities being performed in active contexts.
Workflow components	
Element Provider [wf_element_provider]	Template definitions for custom activities.
Group approval [sysapproval_group]	Group-level approvals.
Variable [item_option_new]	
Workflow Condition [wf_condition]	All of the defined conditions in workflows.
Workflow Element Definition [wf_element_definition]	Parent table for activity definitions.
Workflow Estimated Runtime Configuration [wf_estimated_runtime_config]	Runtime performance data for completed workflows.
Workflow Queued Command [wf_command]	Temporary internal storage for workflows that are currently executing.
Workflow SC Variable [wf_variable]	The Service Catalog variables for a workflow.
Workflow Schedule [wf_workflow_schedule]	Definitions of the times to run specific workflows.
Workflow Timing [wf_workflow_timing]	Timing performance data for workflows.
Workflow Transition [wf_transition]	All of the defined transitions in workflows.
History	
Workflow Activity History [wf_history]	The history of executed activities.
Workflow Log Entry [wf_log]	All of the events and history of the workflow.
Workflow Transition History [wf_transition_history]	The history of executed transitions.
Stages	
Stage Default [wf_stage_default]	Definitions of default stage fields for tables to use.
Stage Set [stage_set]	A named set of stages that can be used to populate workflow stages for multiple workflows.
Stage Set Entry [stage_set_entry]	The stages that belong to a named stage set.
Stage Set for Table [stage_set_table]	Defines a relationship of a stage set to a table so that the stage set can be used as the default stages when a new workflow is created for the table. This replaces the wf_default_stage table

Workflow tables (continued)

Table	Description
	and is the view that shows when you click **Default Stages (by table)** in the menu.
Workflow Stage [wf_stage]	Definitions of stages used by workflows.

Workflow administration

Tailor workflows exactly the way you want them.

Workflow roles

Certain roles are required to use workflows.

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

Workflow roles

Role title [name]	Description
Activity creator [activity_creator]	Creates and edits custom workflow activities, reuses custom activity data, and manages activity packs downloaded from the ServiceNow Store.
Web service administrator [web_service_admin]	Accesses and uses REST and SOAP messages in the Orchestration activity designer. Creates and edits custom activities that use the REST web service and SOAP web service templates.
Workflow administrator [workflow_admin]	Checks out, creates, edits, publishes, and deletes graphical workflows. **Warning:** Granting this role to a user is equivalent to giving the user the admin role, because workflow Script activities bypass access controls and grant access to all tables and database operations. Script activities do not bypass application scope settings.
Workflow creator [workflow_creator]	Checks out, creates, edits, and deletes graphical workflows. **Warning:** Granting this role to a user is equivalent to giving the user the admin role, because workflow Script activities bypass access controls and grant access to all tables and database operations. Script activities do not bypass application scope settings.
Workflow publisher [workflow_publisher]	Checks out with force checkout option, validates, publishes, and deletes graphical workflows.

Administering workflow contexts

The workflow context performs the activities and transitions defined in the workflow with the new record as current.

Workflow in ServiceNow names a running workflow a Workflow Context. The Workflow Context maintains the state of the overall process in the Workflow Context record. The Workflow Context maintains the state of the individual activities as they execute in a series of related lists. These lists maintain the state of currently executing activities, the result of finished activities, and the execution path the workflow took through the process model.

The Workflow Context canvas provides a visual representation of the execution path the workflow took through the process model. The state of each activity (finished, executing, cancelled, error) is represented using the color palette. The executed paths are represented in the color blue; the non-executed paths are represented in grey. Active and historic workflow contexts, as well as the activities within them, can be viewed using the Live Workflows section of the Workflow application menu.

Viewing a workflow context

Workflow contexts can be found in two places:

- From the **Workflow Context** related link on the form of the task being powered by the workflow.
- By navigating to **Workflow** > **All Contexts** and selecting an active context.

Displaying workflow progress

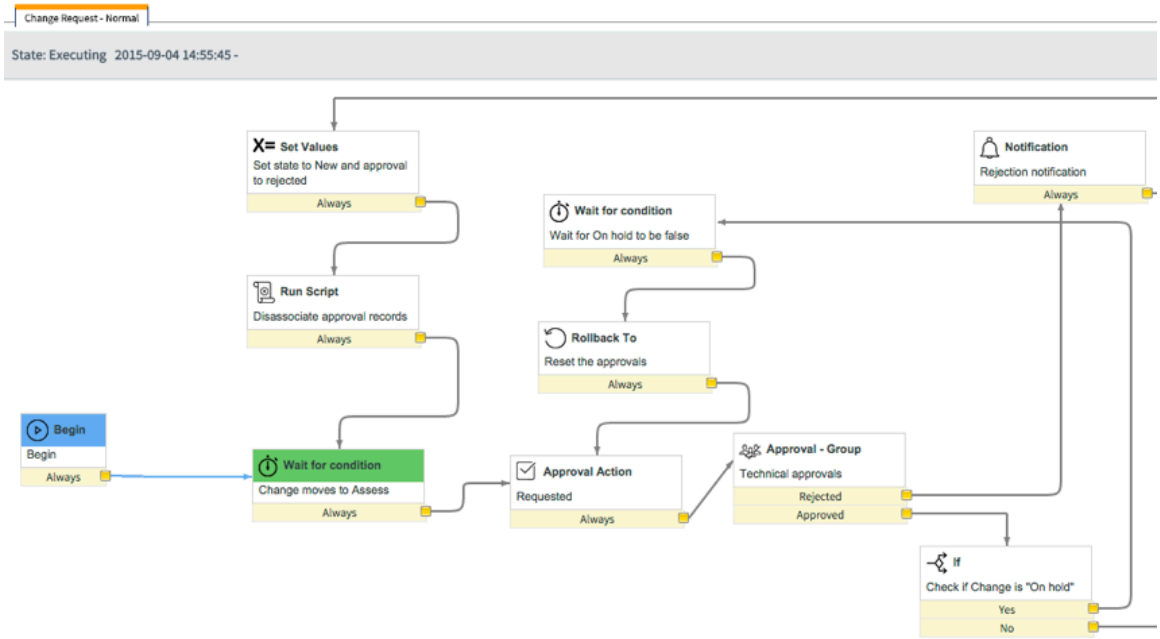
Two related links on the Workflow Context form allow you to view the progress of a workflow in different formats.

- **Show Timeline** displays the workflow context as a timeline.
- **Show Workflow** displays the workflow context in the graphical Workflow Editor.

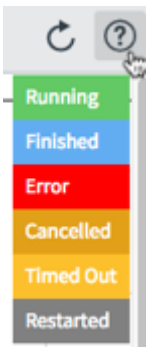
Graphical interface

To view the workflow context in the graphical Workflow Editor interface, click the **Show Workflow** link from either the workflow context record or the current record.

Show workflow



In the top right hand corner are two controls:

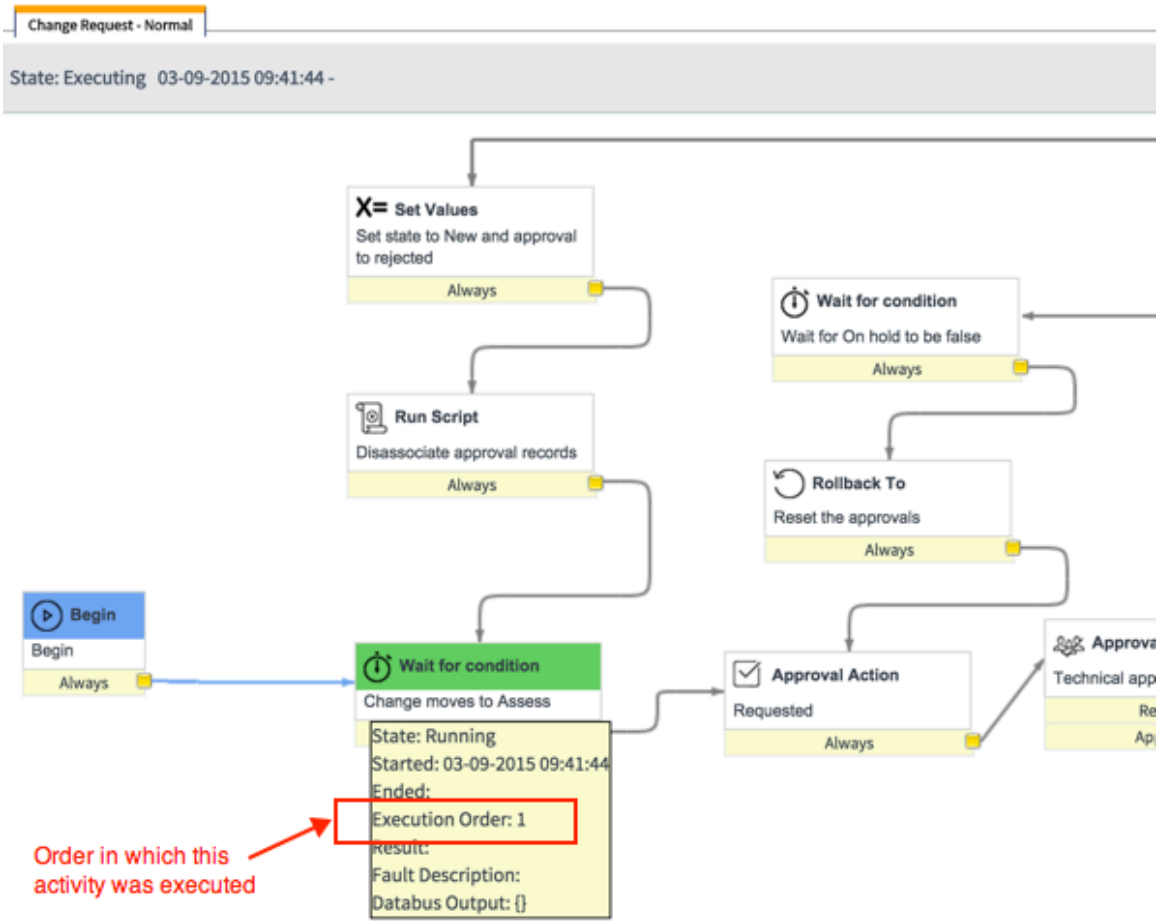
	Refreshes the workflow context.
	Displays a key of the colors used in the workflow to denote the state of activities and transitions: 

Execution order

View tooltip text in the workflow context graphical view to see the execution order of individual activities.

In **Workflow** > **Live Workflows** > **Active Contexts** or **All Contexts**, Open the context you want to examine. Click **Show Workflow**, and point to a finished or executing activity. The tooltip shows error data, execution time, and the order in which the activity executed in the workflow. Use this data to help troubleshoot activities in an error state.

Execution order workflow



Cancel a workflow

Canceling a workflow stops the workflow from executing and sets the workflow context **State** to **Canceled**. To cancel an executing workflow, you can use the `cancelContext(context)` script. You can define an onCancel script to clean up unresolved workflow activities.

Before you begin

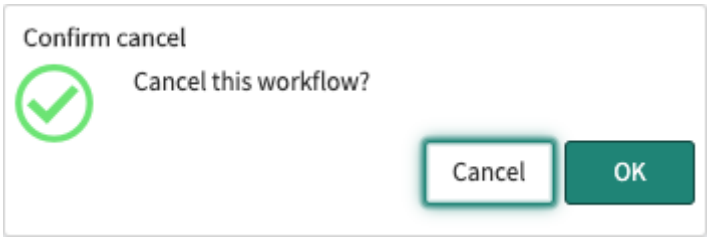
Role required: admin or workflow_admin

About this task

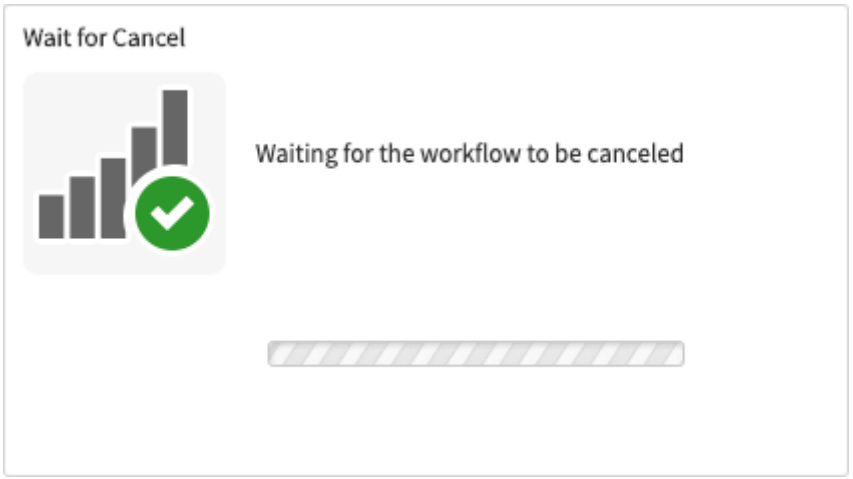
Canceling a workflow attempts to stop the workflow gracefully by injecting a cancel command into the workflow engine.

Procedure

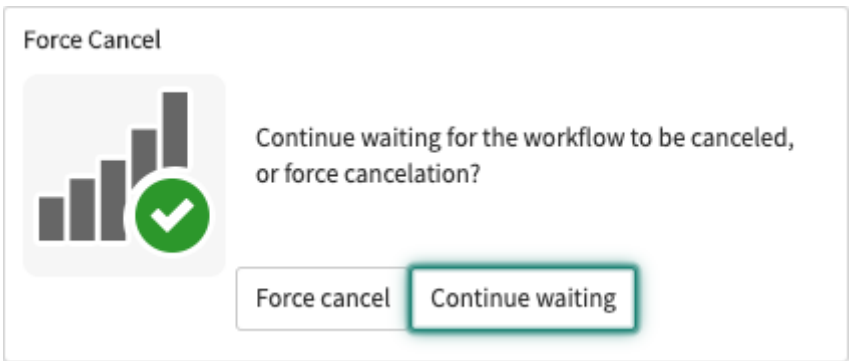
1. Navigate to **All** > **Workflow** > **Active Contexts**.
2. Select a workflow context record.
3. Configure form layout to add **On-cancel script** field to form.
   For detailed information about configuring form layout, see Configuring the form layout.
4. Select the **Cancel** related link.
   A confirmation appears.



5. Click **OK**.
   The workflow engine attempts to cancel the workflow gracefully.



If the workflow does not respond to the cancel command, the Force Cancel option appears.



6. Click **Force cancel** to interrupt the thread the workflow is actively executing or click **Continue waiting** to continue waiting for the workflow to cancel gracefully.

> **Warning:**
> Whenever possible, allow a workflow to cancel gracefully. Forcing a workflow to cancel can leave related workflows and scripts in an unresolved state. You can use an on-cancel script to clean up unresolved artifacts from a cancelled workflow.

Cancel a workflow with the cancelContext(context) script

To cancel an executing workflow, you can use the `cancelContext(context)` script. This script can be useful in cases where a workflow must be canceled in response to an event or where a user must manually cancel a workflow.

Before you begin
Role required: admin

About this task

For more information, see the Workflow - cancelContext(GlideRecord context).

Define an on-cancel script

Canceling a workflow can leave records or scripts in an unresolved state. For example, canceling a service catalog workflow may leave catalog items in the requesting user's cart. An administrator can specify an On-cancel script that runs when the workflow transitions to the Canceled state. This script can notify users, log information, or resolve the state of any scripts run within a workflow activity. The sys_id of the workflow context is available in this script using the context_sys_id variable.

About this task

On-cancel scripts run asynchronously from the global scope. Your instance workload determines when the system schedules and runs the on-cancel script.

Important:
Since the system runs on-cancel scripts from the global scope, they cannot call or run scoped script includes.

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Versions**.
2. Select a workflow version that you have checked out.
   Workflow versions that are not checked out are not editable.
3. Edit the **On-cancel script** field.
   You may need to configure the form to add this field.
4. Click **Update**.

Example:

This example script adds a comment to a Requested Item [sc_req_item] record indicating the workflow for that request has been canceled.

```
var grContext = new GlideRecord("wf_context");
grContext.get(context_sys_id);
var grReq = new GlideRecord("sc_req_item");

// The current record may not exist, make sure it exists before
 modifying it.
if (grReq.get(grContext.id)) {
    grReq.comments = "The workflow processing this item was
 Canceled. Contact your system administrator for further
 information.";
    grReq.update();
}
```

Scheduling a workflow

In addition to being run based on conditions, workflows can also be scheduled to perform at a particular time, similar to a scheduled job.

Scheduled workflows do not have a defined current record and do not run on specific records within a table. Due to this behavior, certain activities that depend on a current record, such as **Create Task** or **Catalog Task** are unavailable on scheduled workflows. If your workflow requires one of these activities, consider using a scheduled job that inserts a record to start the workflow instead of using a scheduled workflow.

To schedule a workflow, navigate to **Workflow** > **Scheduled Workflows** and click **New**. Populate the following fields:

Field	Description
Name	A unique name for the scheduled workflow.
Workflow	Select an existing published workflow to be triggered at the specified date, time, or interval.
Active	If selected, the scheduled workflow will be triggered at the appropriate time.
Application	Specifies the type of application, such as Global.
Run	A choice list to determine when the workflow should be triggered. Options are: • **Daily:** At a particular hour every day. • **Weekly:** On a particular day of the week. • **Monthly:** On a particular day of the month • **Periodically:** After every set duration. • **Once:** At one specific date and time.
Time	For daily/weekly/monthly scheduled workflows, the time of day to run the workflow.
Day	For weekly scheduled workflow, the day of the week to run the workflow. For monthly scheduled workflows, the day of the month to run the workflow.
Repeat Interval	For periodical workflows, the interval between workflows, beginning from the **Starting** date and time.
Starting	For periodical workflows, the first date and time to run the workflow. For scheduled workflows run once, the date and time to run the workflow.

Related topics

Create a scheduled job

Workflow movement with update sets

The system tracks workflows in update sets differently than other records because workflow information is stored across multiple tables.

Changes made to a workflow version are not added to the update set until the workflow is published, at which point the entire workflow is added into the update set. Update sets store workflows as a single Workflow [wf_workflow] record and only retain the latest version with the update type of Workflow.

For information about update sets, see System update sets.

Workflow update set migration use case - simple

Create a new workflow with no dependencies and then migrate the workflow in an update set.

1. User A selects Update Set A.
2. User A creates a new workflow called Workflow A.
3. User A publishes Workflow A.

   A customer update set record is added to Update Set A containing an XML payload, including the published Workflow A and all activity dependencies. The XML payload also contains the workflow input variables associated with the workflow.
4. User A completes Update Set A and migrates it to the production instance.
5. Update Set A commits successfully.
6. Workflow A works as expected.

Workflow update set migration use case - subflow dependency (success)

Successfully edit and migrate an existing workflow and its dependent subflow.

1. User A selects Update Set B.
2. User A checks out Workflow A.
3. User A adds a subflow called Workflow B to Workflow A.

   Assume that Workflow B was previously published and migrated to the production instance.
4. User A publishes Workflow A.

   A customer update set record is added to Update Set B containing an XML payload, including the published Workflow A and all activity dependencies. The XML payload also contains the workflow input variables associated with the workflow.
5. User A completes Update Set B and migrates it to the production instance.
6. Update Set B commits successfully.
7. Workflow A works as expected with Workflow B as a subflow.

Workflow update set migration use case - subflow dependency (failure)

Edit and migrate an existing workflow from a test instance to a production instance that fails to run on the production instance because of a missing dependent subflow.

1. User A selects Update Set C.
2. User A checks out Workflow A.

3. User A adds a subflow called Workflow B to Workflow A.

   Assume that Workflow B was previously published, but has not been migrated to the production instance.

4. User A publishes Workflow A.

   A customer update set record is added to Update Set C containing an XML payload, including the published Workflow A and all activity dependencies. The XML payload also contains the workflow input variables associated with the workflow.

   Notably absent from Update Set C is the subflow called Workflow B. Workflow B was published before User A selected Update Set C.

5. User A completes Update Set C and migrates it to the production instance.
6. Update Set C commits with warnings.
7. Workflow A is invoked on the production instance with the following results:

   Workflow A fails the runtime validation check and is prevented from running on the production system. The system adds to the workflow context a workflow log entry detailing the cause of the failure, notably the absence of a dependent workflow.

   To learn more about the validation checks on workflow dependencies and update sets see ValidateUpdateSetDependencies.

Workflow update set migration use case - subflow dependency (risk)

Multiple users migrate a workflow from a test instance to a production instance without proper coordination. This use case can succeed, but only when each user understands the dependencies and properly migrates the dependent parts of the workflow to the new instance.

This example does not represent an update set failure, although update sets are most often blamed in this use case. Validation increases the visibility of workflow dependencies across multiple update sets and provides designers with better information. In most cases, the warnings do not prevent an action, but only identify risk. The designer is responsible for taking action on advice given in the validation checks.

1. User A selects Update Set C.
2. User A checks out Workflow A.
3. User A adds a subflow called Workflow B that returns a **User ID**.

   Note:
   Assume that Workflow B was previously published and migrated to the production instance.

4. User A uses the return value of Workflow B to generate approvals.
5. User B selects Update Set D.
6. User B checks out Workflow B (the subflow in Workflow A).
7. User B modifies the return value of the workflow by changing it from a **User ID** to a **String Message**.
8. User A publishes Workflow A.

Note:
A dialog box displays warnings associated with Workflow A and encourages User A to validate the workflow before publishing.

9. User A cancels publishing and validates Workflow A.
10. User A is warned that Workflow B was modified by a user in a different update set.
11. User A ignores this warning and publishes Workflow A.

Note:
A customer update set record is added to Update Set C containing an XML payload, including the published Workflow A and all activity dependencies.The XML payload also contains the workflow input variables associated with the workflow.

12. User A completes Update Set C and migrates it to the production instance.
13. Workflow A is invoked on the production instance and runs successfully using the older version of Workflow B already on the system.
14. User B publishes Workflow B.

Note:
User B is not warned of the Update Set C dependency, because the update set is no longer In progress. However, User B is informed via a dialog box that there are warnings associated with the workflow version and is instructed to validate Workflow B. If User B cancels publication and validates the workflow, User B is warned that there are workflows that use Workflow B as a subflow. Knowing the return value was changed, User B should test those workflows as well. See ValidateUpdateSetDependencies to understand the parameters of update set warnings.

15. User B finally publishes Workflow B.

Note:
A customer update set record is added to Update Set D containing an XML payload, including the published Workflow B and all activity dependencies.

16. User B completes Update Set D and migrates it to the production instance.
17. Update Set D commits without warnings.
18. Workflow A is invoked on the production instance and fails to run successfully, because the return value of Workflow B no longer generates a User ID.

Input variable movement

You can add input variables to existing workflows and add them to update sets.

When you submit the new variables, an entry is made into the current update set that reflects the addition of a variable to the Variables [var_dictionary] table. Unlike the workflow version that only writes to the update set when the workflow is published, the variables write individual update entries into the currently selected update set immediately upon submission.

Input variable movement use case - two input variables

An existing workflow already contains two input variables.

1. User A checks out the workflow.
2. User A adds two input variables.

   ServiceNow adds to the current update set one customer update record for each new variable.

   The current workflow now has 4 input variables: the two that were present prior to check out and the two new ones.
3. User A publishes the workflow.

   There are now three related customer update records: two for new variables, and one for the published workflow. The XML payload of the new workflow version now includes all input variable database entries. So while the two original input variables do not have individual customer update records, all four variables are migrated to the local instance with the payload of the newly published workflow version.
4. Verify variables included in a specific workflow.
5. User A completes the update set.
6. Adding Input Variables - Success

   User A migrates and commits the update set to a local instance where the original workflow version had previously been committed.
   - The two existing input variables are already present because of the earlier version.
   - The system adds the two new input variables when the user commits the update set.
   - The system preserves the two legacy input variables on the instance receiving the update set. The update set does not overwrite these variables.
   - The new published workflow version uses all four variables.
   - The user tests the new workflow version and it runs as expected.

Verify variables included in a specific workflow

You can verify the variables that are included in a specific workflow.

Procedure

1. Navigate to **All** > **System Update Sets** > **Local Update Sets**.
2. Select the active update set.
3. Select the customer update entry for the workflow.
4. View the **XML Payload**.
5. Search for the name of one of the columns or search for var_dictionary.

   There is one var_dictionary entry for each input variable.

Input variable removal

Deleting workflow input variables, like insert and update actions, creates a customer update record in a user's current update set.

These deletions migrate to a new instance with the update set, regardless of whether the workflow that owns the input variables is published in the same update set. Plan carefully and use caution when editing a workflow and selecting update sets.

Input variable removal risk

An existing workflow already contains two input variables.

1. The workflow was migrated to a production instance with the two variables.
2. On a development instance, User A selects Update Set A and checks out the workflow.
3. User A removes one input variable and all references to it in the workflow.

   The system enters into Update Set A one customer update record reflecting the deletion of the input variable. No record is added for the new workflow version which no longer depends on the input. This does not happen until the workflow is published.
4. User A continues working on other features in Update Set A that need to be moved to production.
5. User A completes Update Set A and migrates it to the production instance without publishing the workflow.

   The update set entry that deletes the workflow input variable now applies to the production instance. The prior version of the workflow is running on this instance and still references the missing variable.

Input variable removal solution

When editing workflows, particularly when deleting input variables, be sure to use a single update set for all variable editing and workflow publishing.

If necessary, merge the update set into a more general set targeted for deployment after the workflow is published. For information about update sets, see System update sets.

Note:
If a workflow version is already running on a production system and input variables are deleted from a newer version, those deletions could affect transactions already running against the earlier version. Use extreme caution when deleting workflow input variables and plan the migration carefully.

Input variable removal prevention

Prior to publishing a workflow version, the system validates the workflow model to assist the designer in planning for deployment.

This validation warns of critical errors that can prevent a workflow from running successfully, but also warns of dependencies and conflicts in update sets. See ValidateUpdateSetDependencies for more details.

Avoiding duplicate workflows

Update sets manage the published state of all versions of a workflow prior to committing the workflow version on a local instance.

The last version of a workflow committed as an Insert or Update using an update set becomes the currently published version, regardless of the publishing sequence for the workflow versions.

Commit a workflow in an update set

Follow the steps in this page to commit a workflow in an update set.

Procedure

1. Workflow A - Version 1 is created and published in Update Set A.
2. Update Set A is completed and migrated to a local instance.
3. When the update set is committed, the system sets all prior versions of Workflow A to published = false.

   In the first migration, there are no prior versions.
4. Workflow A - Version 1 becomes the only published version of the workflow.

Update set migration example

It is not possible to have multiple published versions as a result of update set commits. However, this does not eliminate risk, and care should be taken when migrating update sets.

Consider this example:

1. Workflow A - Version 1 is migrated and committed to the production instance.
2. Update Set B is created.
3. Update Set C is created.
4. Workflow A - Version 2 is published in Update Set B.

   A customer update record is added to Update Set B with the Version 2 payload.

   A customer update record is added to Update Set B with the Version 1 workflow left unpublished.
5. Update Set B is completed.
6. Workflow A - Version 3 is published in Update Set C.

   A customer update record is added to Update Set C with the Version 3 payload.

   A customer update record is added to Update Set C with the Version 2 workflow left unpublished.
7. Update Set C is completed.
8. Update Set C is migrated and committed to the production instance.

   Workflow A - Version 1 is set to unpublished.

   Workflow A - Version 2 update is skipped since Update Set B, which contains Version 2, was never migrated.

   Workflow A - Version 3 is committed and becomes the only published version of the workflow.

Update set migration risk

Update Set B is migrated and committed to the production instance.

1. Workflow A - Version 3 is set to unpublished.
2. Workflow A - Version 1 remains unpublished.
3. Workflow A - Version 2 is committed and becomes the only published version of the workflow.

   The workflow has gone back a version, perhaps unintentionally. The regressed version becomes the currently published version.

Workflow timelines

The system provides a timeline view of history activities associated with a workflow context.

Timelines display a linear calendar of activities, such as tasks and approvals, defined by their start and end dates. Each activity on the timeline is represented by a span, which is displayed as a horizontal, colored bar. Each span has a label and a tooltip that contains additional information about the activity. The left pane displays all the activities in the context (or contexts) in an expandable hierarchy. You can change the timeline's perspective for a more granular view of the data.

Note:
Workflow timelines reflect context history only and are not real-time gauges of workflow activity.

Use a timeline

By default, the timeline displays all activities and transitions requested when first opened.

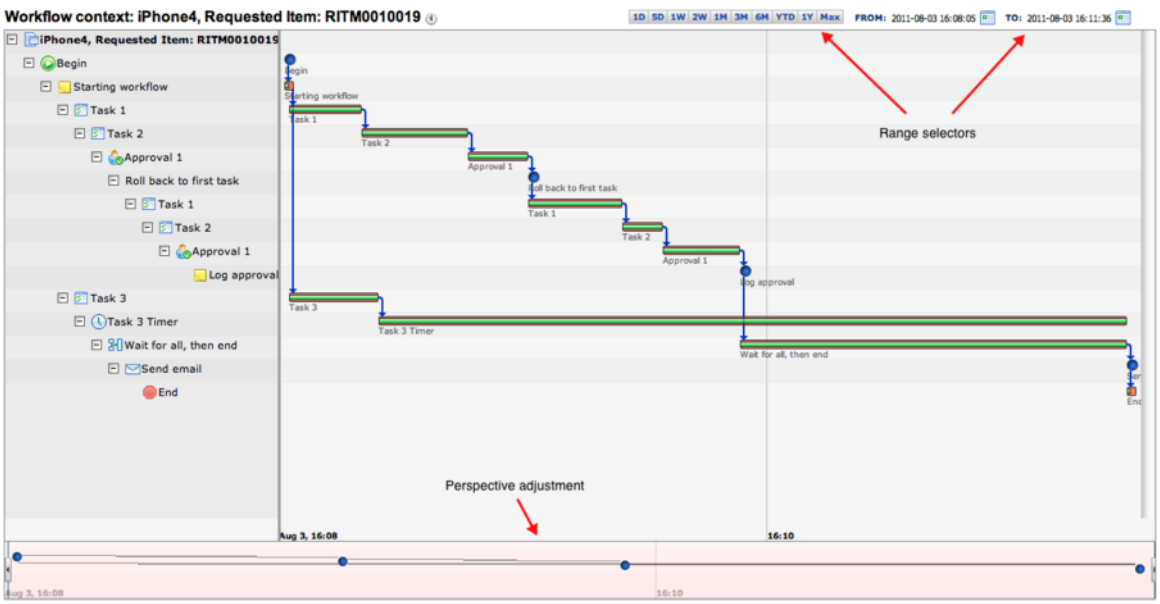
About this task

Collapse any part of the hierarchy in the activity pane and the timeline adjusts automatically. Date/time and duration controls enable you to scale the timeline to view all the elements at once. To display a timeline, click a UI action within a Workflow Context record.

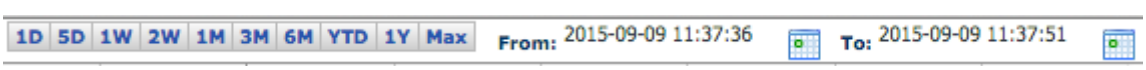
Procedure

1. Navigate to**All** > **Workflow** > **Live Workflows** > **Active Contexts**.
2. Select a context.
3. In **Related Links**, click **Show Timeline** to display the timeline for the entire context.

   The timeline opens with all activities expanded and the view set to **Max**, which displays the entire timeline at the width of the pane. The title of the timeline is in the form Workflow context: <context name>, Requested Item: <requested item number>.



4. Use the Range Selectors at the top of the timeline to change the perspective.



The increments go from one day to one year. To limit the timeline to an increment between the start date of the first span and the end date of the last span, click **Max**.

5. Use the starting and ending calendar fields to select the timeline perspective.

   These fields control the same perspective as the slider at the bottom of the timeline.

6. Use the pink slider at the bottom of the timeline to change the perspective.

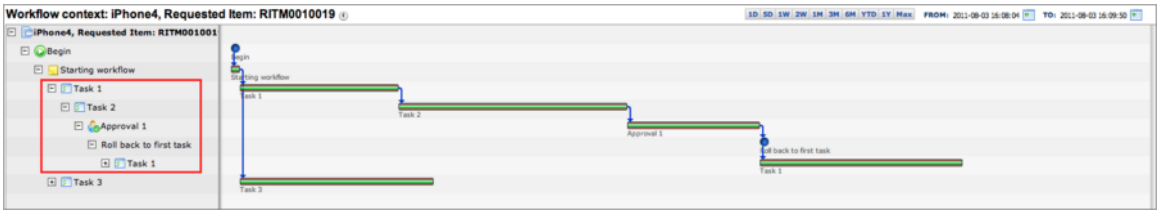
   a. Move the slider from right to left to view all the spans on a long timeline.

   b. Adjust the end points of the slider to make arbitrary changes to the magnification.

      A narrow slider zooms in on the spans and provides a more detailed view of complex timelines. A wide slider pulls the view out and makes more of the timeline visible on the screen.

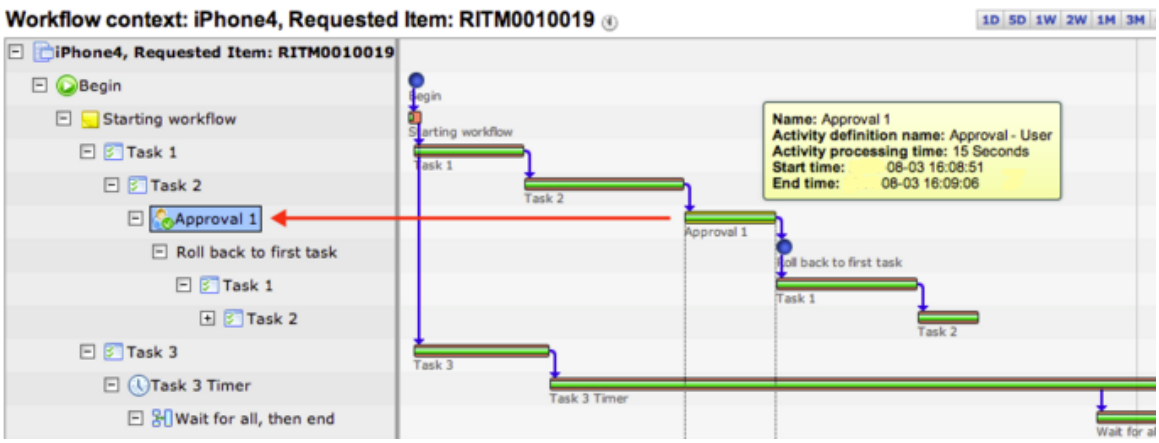
7. To focus the timeline view on selected activities, expand or collapse the activity tree.

   Spans not visible in the activity tree are not shown in the timeline pane.

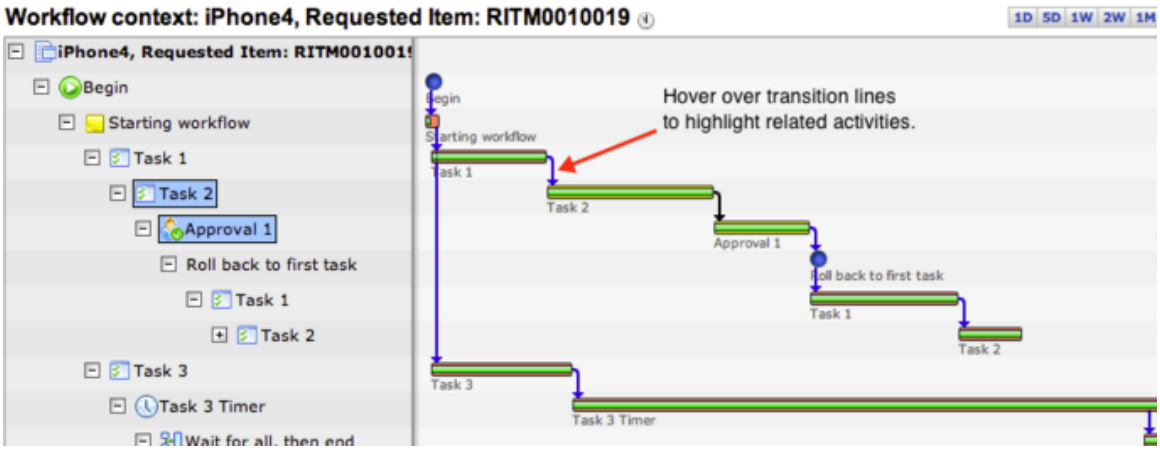


8. Hover over an activity span to display a tooltip with information about the activity.

   This action highlights the activity in the activity pane.

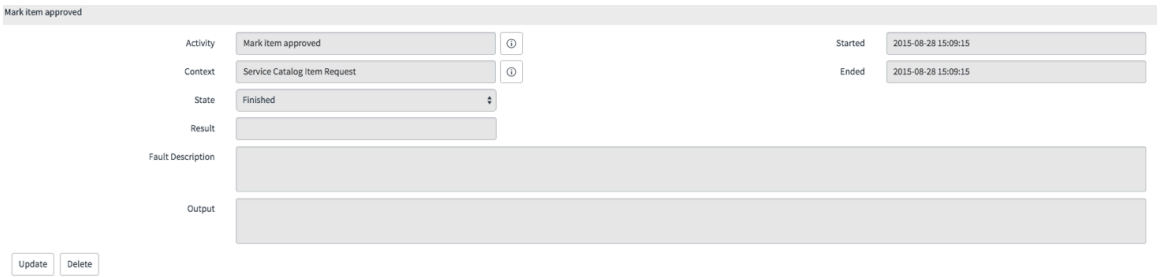


9. Hover over the transition between two activities to highlight the activity and the predecessor activity in the activity pane.



10. Double-click a span to display a history record for that activity.

    History records shows information such as the **State** and the starting and ending times.



Timeline for a selected activity

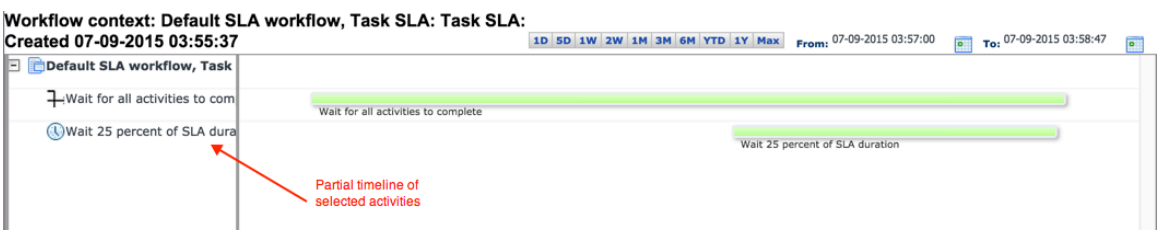
You can display a partial workflow timeline.

Procedure

1. Navigate to**All** > **Workflow** > **Live Workflows** > **Active Contexts**.
2. Open a Workflow Context record.
3. On the **Workflow Activity History** related list, select one or more individual activities.
4. Click **Show Timeline** from the action menu.

   The resulting view is a snapshot of the timeline, showing only the selected activities and their transitions, if any.

5. Collapse the tree to confine the view even further.



6. To view a timeline displaying activities from different contexts:
   You might use this feature to display a subflow's context with the parent workflow context.

a. Navigate to **Workflow** > **Live Workflows** > **History**.

b. Select individual history items from the list.

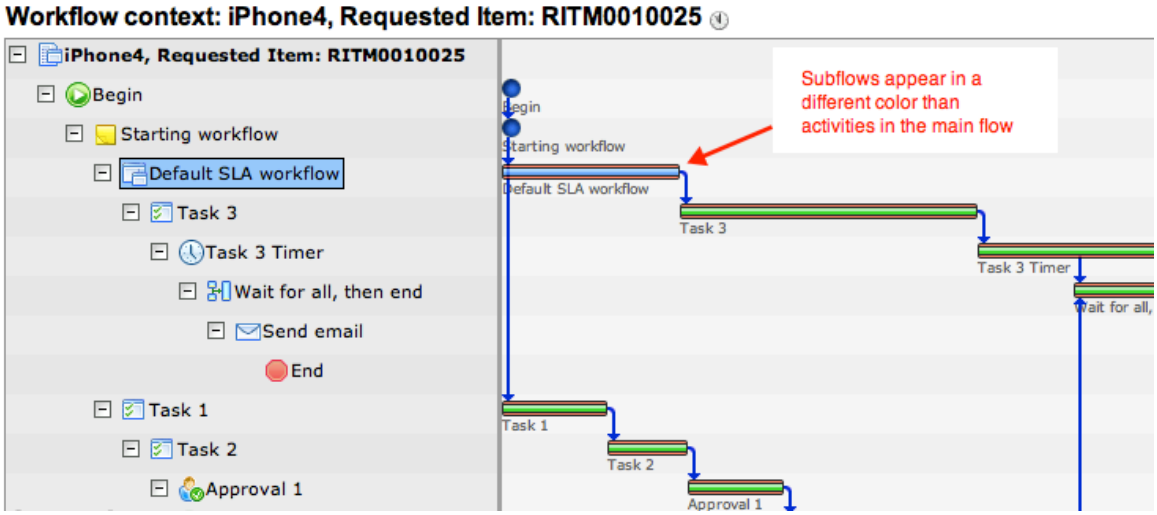
c. Select the **Show Timeline** option from the actions menu.

History items are arranged in a hierarchy in the activities pane under their contexts. The timeline title is **Multiple Contexts**. The timeline draws only the activities and relationships of the history items selected.

View subflows in a workflow timeline

In a workflow timeline, subflow spans appear as a different color than the activities of the main workflow.

Workflow timeline subflow



Workflow error handling

The personalized JavaScript that users create in workflow activity variables is vulnerable to run-time syntax errors. Available error information is available in a tooltip when you point to a workflow activity in an error state.

In the base system, workflow activities do not provide condition routing on the error state. As a result, the workflow progresses based on the state of the current record. For example, a workflow contains an **Approval - User** activity that uses an advanced script to add additional approvers. A syntax error in the script results in no approvers being added. Because a state of **no approvers** is a valid return, even without the syntax error, the approval activity is skipped and the workflow progresses along a positive path. However, this might not be a valid response for the workflow designer who does not want the workflow to progress along the positive path without approvers.

Workflow error handling detects and logs syntax errors and provides a state that the workflow designer can use to add error conditions to the workflow. Use error handling to locate syntax errors in advanced script fields for these workflow activities:

- Approval - User
- Approval - Group
- Catalog Task
- Create Task
- If

- Run Script
- Notification

Available error information

This table shows which activities support error exits.

Available error information

Activity	Workflow log	Red error indicator	Activity state	Activity result	Fault description	Reroute on error
Approval - User Approval - Group	Yes	Yes	Error	Skipped	Yes	Yes
Catalog Task Create Task	Yes	No	Finished	none	No	No
If	Yes	Yes	Error	none	Yes	Yes
Run Script	No	Yes	Error	error	Yes	Yes
Notification	Yes	Yes	Error	error	Yes	Yes

Workflow error tracking features

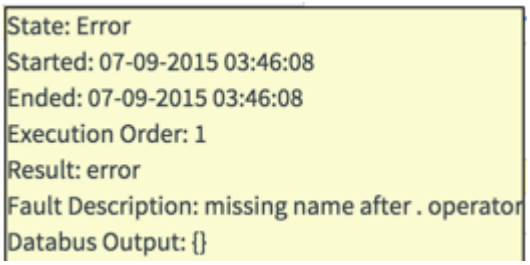
Error handling provides visual cues within the workflow, such as error descriptions for activities in pop-ups, and detailed log records.

Banners

Look for an activity with a red banner, indicating that a syntax error has occurred in a script field. All activities that provide error handling, with the exception of **Catalog Task** and **Create Task**, display a red banner for this error.

Tooltips

Point to the activity displaying a red banner to view information about the error. A tooltip shows the **State** and **Result** of the activity and provides a brief **Fault Description** (except for task activities). Note that this approval continued as skipped despite the error given in the fault description. See Workflow error handling for the information available to each activity.



Execution order

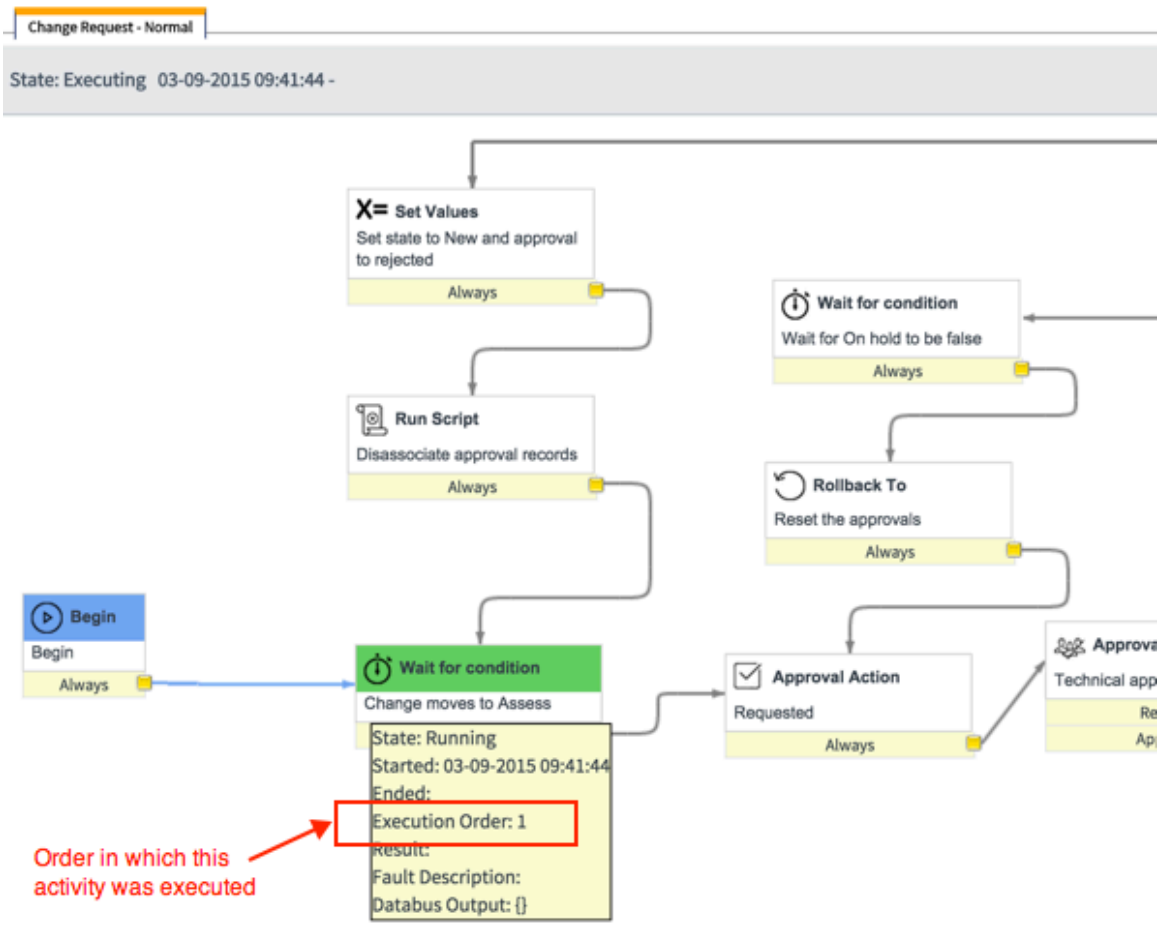
Tooltip text in the Workflow Context graphical view displays the execution order of individual activities, which assists in troubleshooting.

To view the order in which a workflow activity was executed:

1. Navigate to **Workflow** > **Live Workflows** > **Active Contexts** or **All Contexts**.
2. Open the context you want to examine.
3. Click **Show Workflow**.
4. Hover the cursor over a finished or executing activity.

   A tooltip appears showing error data, execution time, and the order in which that activity executed in the workflow. You can use this data to help troubleshoot activities in an error state.

Activity execution order



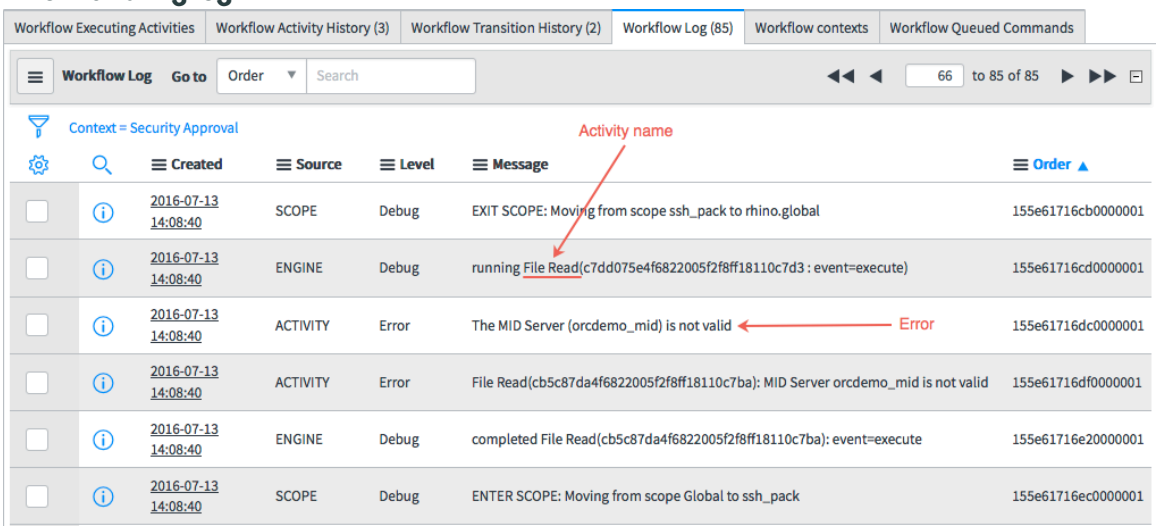
Workflow log

View the log in the Workflow Context form for more information about the syntax error in the activity. Since task activities do not display a red banner when a syntax error has occurred, you must view the log if you suspect the workflow has not run properly. Examine the error description in the log, and then inspect the script in the activity named in the log.

To view the activity by name, navigate to **Workflow** > **Administration** > **Properties** and enable the **Log workflow debug messages** property.

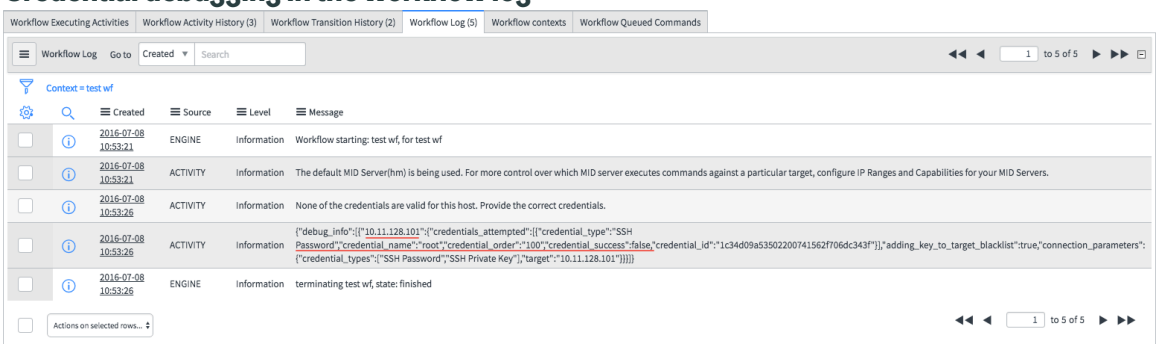
In this example, an SSH activity named File Read specifies an invalid MID Server.

Error handling log



If the credentials used by an activity in the workflow fail, and the activity cannot authenticate on the target, a message describing the failure appears in the **Workflow Log** related list. The message displays the target IP address and the credential details.

Credential debugging in the workflow log



Create an error condition exit

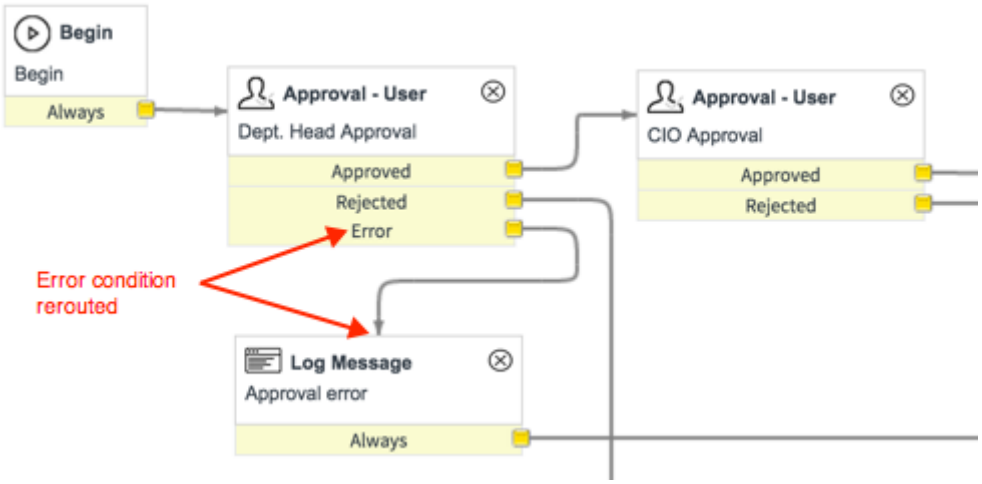
An administrator can reroute the workflow when a script error occurs by creating an error condition exit for specific activities within the workflow. This allows the workflow to process script errors in a predictable way and not create undesirable results.

Procedure

1. Open and check out a workflow.
2. Right-click in the top portion of the activity for which you want to create an error exit.
3. Select **Add condition** from the context menu.

4. Add a condition exit with the following values:
   - Name: `Error`
   - Condition: `activity.state=='faulted'`
5. Click **Submit**.

   The **Error** exit appears on the activity.
6. Connect the **Error** exit to another activity for handling the error state, such as **Notification** or **Log Message**.



Reconfigure an approval condition

Approval activities react differently to script errors than the other activities.

About this task

Approval activity script errors can prevent an approval from being processed successfully. This, in turn, can cause the approval to complete as **Skipped**, which can appear to be an **Approved** state. To prevent this from happening, reconfigure the **Approved** exit as follows:

Procedure

1. Open an approval workflow and make sure it is checked out.
2. On an approval activity, double-click the **Approved** exit.
3. Add the following script to the **Condition** field:

   ```
   && activity.state != 'faulted'
   ```

   This prevents the activity from continuing down the normal path in an error state and ensures that **Skipped** or **Approved** is the desired state.
4. Click **Update**.

Workflow run time metrics

You can enable the collection of workflow run time metrics to determine whether workflows are performing as expected or consuming additional resources.

Outlying run times for a workflow are identified by comparing actual run times to an outlier range calculated with the outlier threshold and estimated run time defined in the workflow properties.

You can monitor the results of these metrics on the Workflow Operations Dashboard and custom homepages with workflow gauges.

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.

Enable workflow run time metrics

Provide an estimated run time that can be compared to actual workflow run times.

About this task

For baseline workflows, you must also manually enable the collection of run time metrics. The system automatically enables the collection of run time metrics for new workflows.

Procedure

1. Navigate to **All** > **Workflow** > **Workflow Editor**.
2. Open and check out the workflow.
3. In the title bar, click the menu icon and select **Properties**.
4. In the Workflow Properties dialog box, click the **Estimated Runtime** tab.
5. To enable the collection of run time metrics, check that the **Requires ERT** option is selected.
6. Open a configuration from the **Estimated Run Time** column.
7. In **Estimated Run Time**, enter an initial estimate for the workflow's run time.
   The system compares this initial estimate to actual run time results to create outlier reports. The system can automatically update this field in certain circumstances. Workflow designers can also manually update this field.
8. In the **Outlier Percentage Threshold for ERT** field, enter the percentage deviation from the estimated run time that identifies an outlier workflow run time.
   The system uses a default value of **20**.
9. Click **Update**.

Outlying workflow run times

Workflow run times are identified as outliers when they are longer or shorter than the outlier range that is computed for the workflow.

The outlier range is automatically computed with the **Estimated Run Time** and **Outlier Percentage Threshold for ERT** values in the workflow properties. These values are used in the following formulas.

Formulas for computing workflow outlier ranges

Value computed	Computation used	Example
Outlier Value	**Estimated Run Time** * (**Outlier Percentage Threshold for ERT** / 100)	10 seconds * (20 / 100) 10 seconds * 0.2 = 2 seconds
Outlier Range	(**Estimated Run Time** - *`Outlier Value`*) to (**Estimated Run Time** + *`Outlier Value`*)	(10 seconds - 2 seconds) to (10 seconds + 2 seconds) = 8 to 12 seconds

When a workflow runs within the outlier range, its estimated run time is automatically updated.

If a workflow has an outlying run time, it appears in any outlier workflow gauges on the Workflow Operations Dashboard and custom home pages.

Workflow estimated run time updates

When a workflow runs within the outlier range, its estimated run time is automatically updated.

The estimated run time is updated with the cumulative moving average of the latest run time value in relation to previous run times. The computed value is rounded to the nearest second and stored as a GlideDateTime.

For example:

Data point	Latest value	Cumulative running average (CRA)	CRA after rounding to the nearest second
1	10 seconds	10 seconds	10 seconds
2	12 seconds	11 seconds	11 seconds
3	9 seconds	10.333 seconds	10 seconds

Note:
Because the system rounds to the nearest second, the calculation is less precise with short durations.

You can also manually update the estimated run time in the workflow properties.

Workflow pause utility

Workflow Pause Utility provides functionality you can use to pause or resume all workflow contexts, a subset of qualified workflow contexts, or individual workflow contexts. You designate the data and time of day at which the paused workflows should resume, and can manually resume individual workflows as needed.

Once activated, the Workflow Pause Utility plugin (com.glideapp.workflow.pause) updates several tables when you pause or resume workflows. Each table stores specific data related to each paused or resumed workflow.

wf_pause_request

Records every workflow pause request. The system automatically updates and inserts records in this table whenever you pause workflow contexts. It tracks user-

specified resume time, whether the pause request is still active, and the total number of workflow activities that paused or resumed.

wf_pause_status

Records the status of each workflow context you pause. When you pause workflow contexts, the plugin waits until the current executing activity finishes, and pauses it before the next activity starts. It tracks the specific activity on which the workflow context was paused and whether it has resumed. If the workflow is paused, it tracks the time at which it is scheduled to resume.

wf_pause_group_request

Pauses the specific set of workflow contexts you designate using filtering in Group Pause Requests. You can manually pause all currently active workflows by clicking the **Pause All** check box.

wf_pause_snapshot

Before pausing and after resuming a workflow, the table records a snapshot of the current state of the workflow context. The table also records the state of the currently executing activity on the paused workflow context.

Use examples

When you take down an instance for maintenance, you can pause all or selected active workflow contexts, and then resume them after you complete the maintenance.

You have an integration workflow context that hits an internal service that is down for maintenance. You can pause all instances of that workflow context from progressing to the next activity, allowing time for the internal service to be restored. Then, the paused workflow contexts can be resumed.

Request workflow pause utility

You can request the Workflow Pause Utility plugin (com.glideapp.workflow.pause) if you have the admin role. This plugin may include demo data and activates related plugins if they are not already active.

Before you begin

Role required: admin

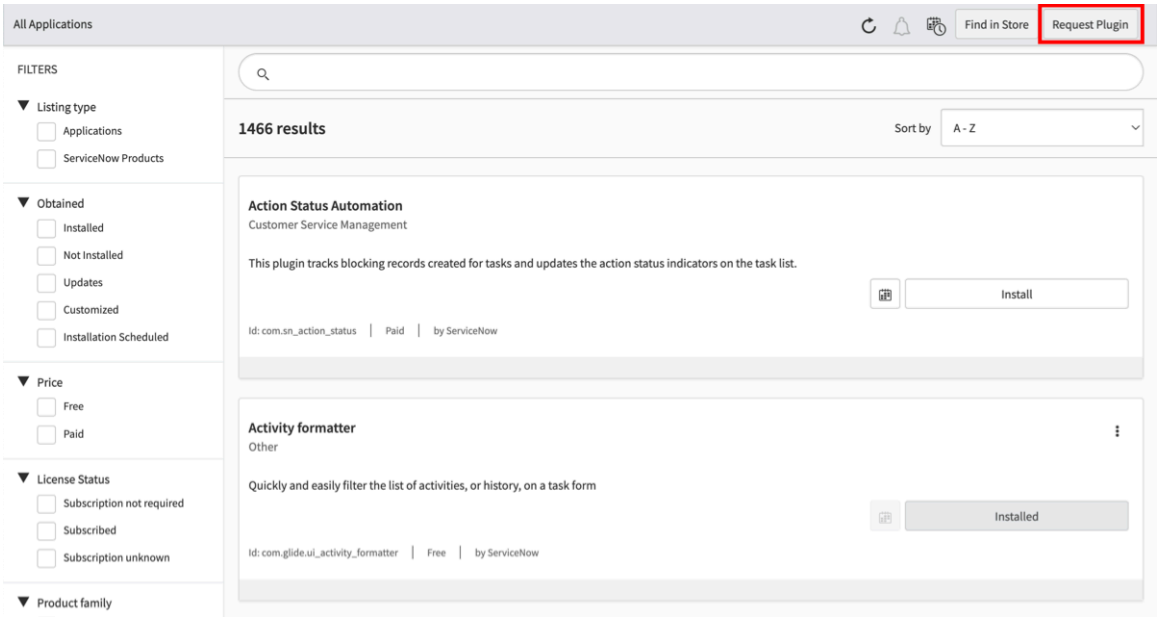
About this task

There are two ways to request a plugin:

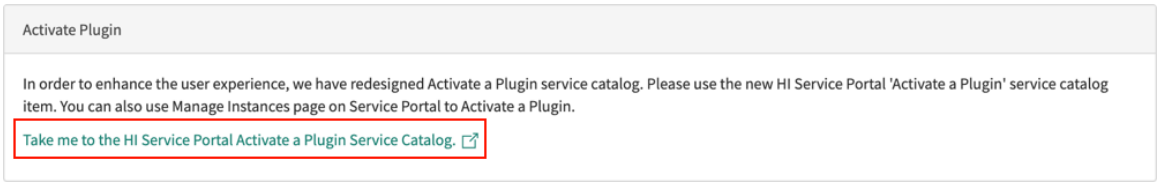
- Access the Now Support Service Catalog directly by selecting **Automation Store** > **Service Catalog** > **Activate Plugin** on Now Support.
- Access the Now Support Service Catalog through the All Applications page on your instance by following these steps.

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. On the All Applications page, select **Request Plugin** to open the **Activate Plugin** form on Now Support.



3. On Now Support, select the link to access the Now Support Service Portal Service Catalog.



4. Select your instance.
5. Select **Actions > Activate Plugin**.
6. On the **Activate Plugin** form, provide the following information.

Activate Plugin form

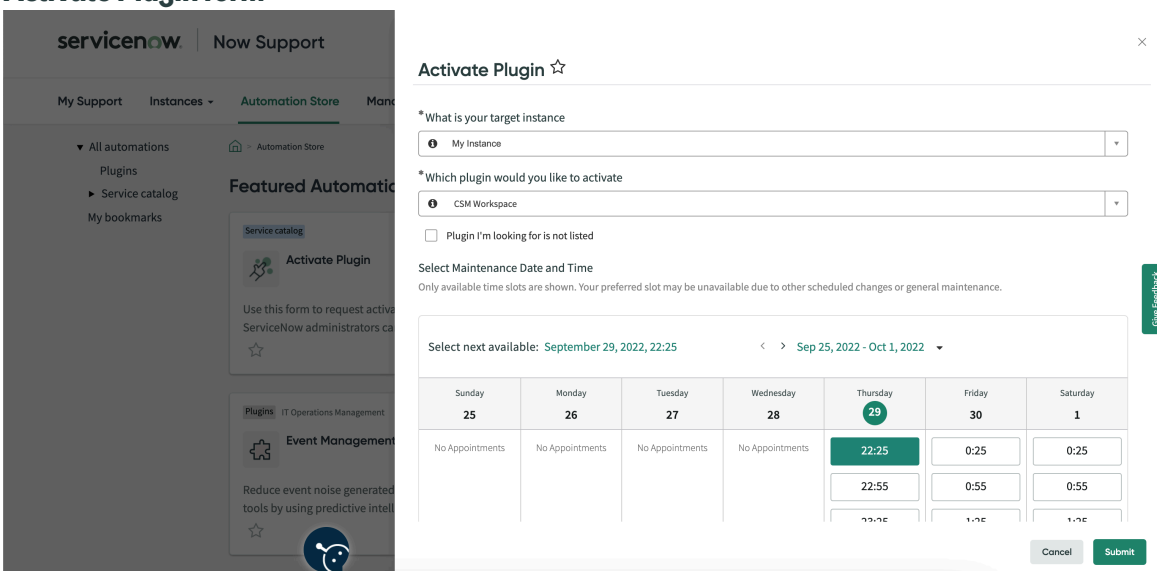
Field	Description
What is your target instance	Instance on which to activate the plugin.
Which plugin would you like to activate	Name of the plugin to activate. **Note:** If the system does not list the plugin you want or if you are activating the plugin on an OEM or on-premise instance, select the **Plugin I'm looking for is not listed** check box and then enter the name of the plugin.
Select Maintenance Date and Time	The date and time to activate the plugin.

Field	Description
	Note: Plugins are activated in two batches, once in the morning and once in the evening, on every business day in the US Pacific time zone. If the plugin must be activated at a specific time, enter the request in the **Reason/Comments** field.

Example

For example, see the following form to activate the CSM Workspace plugin on an instance named My Instance.

Activate Plugin form



7. Select **Submit**.
   For additional details about requesting a plugin, see Requesting a Plugin from the Service Catalog [KB0751715] article in the Now Support Knowledge Base.

Pause and resume individual workflows

Pause individual workflow contexts. Specify the date and time of day at which the workflow context should resume.

Before you begin

Install and activate the Workflow Pause Utility plugin.

Role required: Workflow_admin or workflow_publisher, workflow_creator

About this task

When a workflow context is paused, the system saves basic information about the pause request, such as the resume date and time, in the wf_pause_request table. A workflow context automatically resumes at a specified date and time of day, but the paused workflow context can be manually resumed as needed.

Procedure

1. Navigate to **All** > **Workflow** > **Live Workflows** > **Active Contexts**.
2. Select the individual workflow context that you are pausing.
   Workflow Context appears.
3. In **Related Links**, click **Pause**.
   A **Workflow Pause Inputs** dialog appears.
4. In the **Workflow Pause Inputs** dialog:

   a. Click the calendar icon () to select the date, and enter the time of day (in hours, minutes, and seconds) at which the workflow should resume.
   You can select the current date or future date, but you must enter a time of day in the future. For example, to pause the workflow context at 13:30:00 (1:30 pm), select the date from the calendar, and then enter `13`, `30`, and `00` into the respective **Time:** fields.

   b. Click the green check mark icon () when finished.
   The selected date and time of day appear in the **Resume At** field.

   c. Click **OK** to return to Workflow Contexts.
   The selected workflow is now paused. Use Workflow pause request to monitor its status.

5. Manually resume a currently paused workflow context:
   Paused workflows resume on the date and time of day specified in the **Resume At** field.

   a. Navigate to **Workflow** > **Live Workflows** > **Active Contexts**.

   b. Select the individual workflow context that you are pausing.
   Workflow Context appears.

   c. In **Related Links**, click **Resume**.

Pause and resume all or multiple workflows

Use Workflow Pause Group Requests to pause, or resume, groups of workflows, or all active workflows. If pausing a group of workflows, use filtering functions to select the workflows. If pausing all active workflows, select the Pause All check box to indicate that all currently active workflows should be paused.

Before you begin

Role required: Workflow_admin or workflow_publisher, workflow_creator
Install and activate the Workflow Pause Utility plugin.

Procedure

1. Navigate to **All** > **Workflow** > **Operations** > **Group Pause Requests**.
   Workflow Pause Group Requests appears and displays existing workflow pause group requests.
2. Click **New**.
   Workflow Pause Group Request appears, and assigns a workflow pause group request ID.
3. To pause all active workflows, select **Pause All**, then click **Update**. To pause a filtered group of active workflow contexts, skip this step.
   All currently active workflow contexts in the instance are paused until you resume them. Use Workflow pause request to monitor their status.

4. To pause a filtered group of active workflow contexts, use the **Filter** field, specify the conditions for the search, then click **Update**.
   Workflow Pause Group Requests appears, and displays the results of the workflow group search. It assigns a workflow pause group request ID and indicates if the group is active (true = paused) or inactive (not yet paused).
5. To update, resume or delete a specific workflow pause group request, select it.
   Workflow Pause Group Request appears, and displays the selected workflow pause group request.
6. If the selected workflow pause group request is active (paused), and you do not want to pause it, click **Do Not Pause Incoming Workflows**. If the selected workflow pause group request is inactive (not yet paused), click **Pause** to pause it.
7. For the paused workflow group request, use the **Resume At** field to specify the date and time of day at which the paused workflow contexts should resume.

   a. Click  to select the date, and enter the time of day (in hours, minutes, and seconds) at which the workflow contexts should resume. You can select the current date or future date, but you must enter a time of day in the future.
   For example, to pause the workflow context group at 13:30:00 (1:30 pm), select the date from the calendar, and then enter `13`, `30`, and `00` into the respective **Time:** fields.

   b. Click  when finished.

   c. The selected date and time of day appear in the **Resume At** field.

   d. Click **OK** to return to Workflow Pause Group Request.

8. Click **Submit**.
9. Paused workflow contexts automatically resume on the date and time of day specified in the **Resume At** field. To manually resume currently paused workflow contexts:

   a. Navigate to **Workflow Pause Group Requests**.

   b. Select the workflow pause group request to resume.
   Workflow Pause Group Request appears.

   c. Click **Resume**, and then click **Update**.

10. Workflow Group Pause Request also contains the following information fields.

Workflow group pause request form

Field	Description
Active	Indicates if workflow group is paused.
Completed Resume Activity Count	Number of resumed workflow context activities that are completed.
Filter	Standard filtering fields used to specify the conditions for selection of a group of workflow contexts.

Field	Description
Log	Activity work notes generated by the group pause request.
Not Transitioned Workflow Count	Number of workflow contexts that were not transitioned when the group pause request was placed. When a group pause request is placed, the affected workflow contexts are paused before start of the next activity.
Resume Act Count	Number of workflow context activities resumed after the group pause completed.
Pause Act Count	Number of paused workflow context activities.
Requester Name	Name of the person requesting the workflow context pause.
Pause percentage	Total percentage of workflow contexts that are paused.
Resume Percentage	Total percentage of workflow contexts that have resumed.
Total Workflow Count	Total number of workflows matching the specified filter conditions that are being paused.
Paused Workflow Count	Number of current paused workflow contexts.

Monitor workflow pause requests

You can monitor the status of workflow pause requests using Workflow Pause Request.

Before you begin

Role required: Workflow_admin or workflow_publisher, workflow_creator
You must install and activate the Workflow Pause Utility plugin.

Procedure

1. Navigate to **All** > **Workflow** > **Operations** > **Pause Requests**.
   Workflow Pause Requests appears and displays existing workflow pause requests.
2. Select the workflow pause request.
   Workflow Pause Request appears, and displays the selected workflow pause request.

Workflow pause request fields

Field	Description
Req Number	Unique assigned pause request number.
Completed Resume Activity Count	Number of workflow context activities resumed after the pause completed.
Pause Active	Selected indicates the workflow context is currently paused. Cleared indicates it is not.
Transitioned During Pause	Number of workflow contexts that transitioned when the pause request was placed. When a pause request is placed, the affected workflow contexts are paused before start of the next activity.

Field	Description
Workflow Context	Reference number for the paused workflow context.
Log	Activity work notes generated by the group pause request.
Paused Act Count	Date and time of day (h:m:s) at which the paused workflow context is resuming.
Requester Name	Name of the person who requested the workflow pause.
Resume Act Count	Number of workflow context activities resumed after the pause completed.
Resume At	Date and time of day (h:ms:s) at which the workflow contents are scheduled to resume.
Table	Related record table name.
Application	Application in which the workflow context is executing.
Exec Act Count	Number of activities executing when the workflow context was paused.
Group Pause Request	Reference number for the associated group pause request, if any. Appears only if pause request was the result of a group pause request.
Is Workflow Complete	Selected indicates the workflow context is complete. Cleared indicates it is not.
Is Workflow Stuck	Selected indicates the workflow context is stuck. Cleared indicates it is not.
State	Current state of the workflow context - Paused or Resumed.
Stuck Activities Count	The number of workflow context activities that are stuck.
Stuck Act Count To be subtracted for Pause Percentage	Writer to whom the documentation request is assigned.

Encrypted workflow scratchpad

The Encrypted Workflow Scratchpad plugin (com.snc.encrypted.scratchpad) provides encrypted scratchpad support for workflow context and workflow executing activities.

The platform supports encryption on most fields, and workflows can execute on tables with encrypted fields. Encrypted data is typically stored in the workflow scratch pad for workflows that execute using encrypted fields and must access this data after processing an approval, timer, or create task activity.

Data in the workflow scratchpad is not encrypted by default. The Workflow engine supports scratchpad encryption and executing activity records only when the Encrypted Workflow Scratchpad plugin is activated. Once activated, data stored in the wf_context and wf_executing scratchpads is protected by a private workflow engine encryption module, and is prevented from being stored in plain text.

Incompatibility with domain separation

The Encrypted Workflow Scratchpad plugin is incompatible with domain separation. Activating the Encrypted Workflow Scratchpad plugin on a domain separated instance can produce encryption errors when running some workflows. Consider using a Workflow Studio flow instead.

Request encrypted workflow scratchpad activation

Request activation of the Encrypted Workflow Scratchpad plugin (com.snc.encrypted.scratchpad) from ServiceNow personnel

Before you begin

Role required: admin

The Encrypted Workflow Scratchpad plugin must be activated via a Customer Service and Support request. You should activate and thoroughly test the plugin in a non-production instance. Only after you satisfied with the results should you request activation in a production instance.

When requesting activation in a production instance, you should schedule a low transactional volume time to do so. Prior to its activation, first activate the Workflow Pause Utility (com.glideapp.workflow.pause), and then pause all active workflows. Refer to Workflow pause utility. After activation of the Encrypted Workflow Scratchpad plugin, resume all paused workflows.

About this task

There are two ways to request a plugin:

- Access the Now Support Service Catalog directly by selecting **Automation Store** > **Service Catalog** > **Activate Plugin** on Now Support.
- Access the Now Support Service Catalog through the All Applications page on your instance by following these steps.

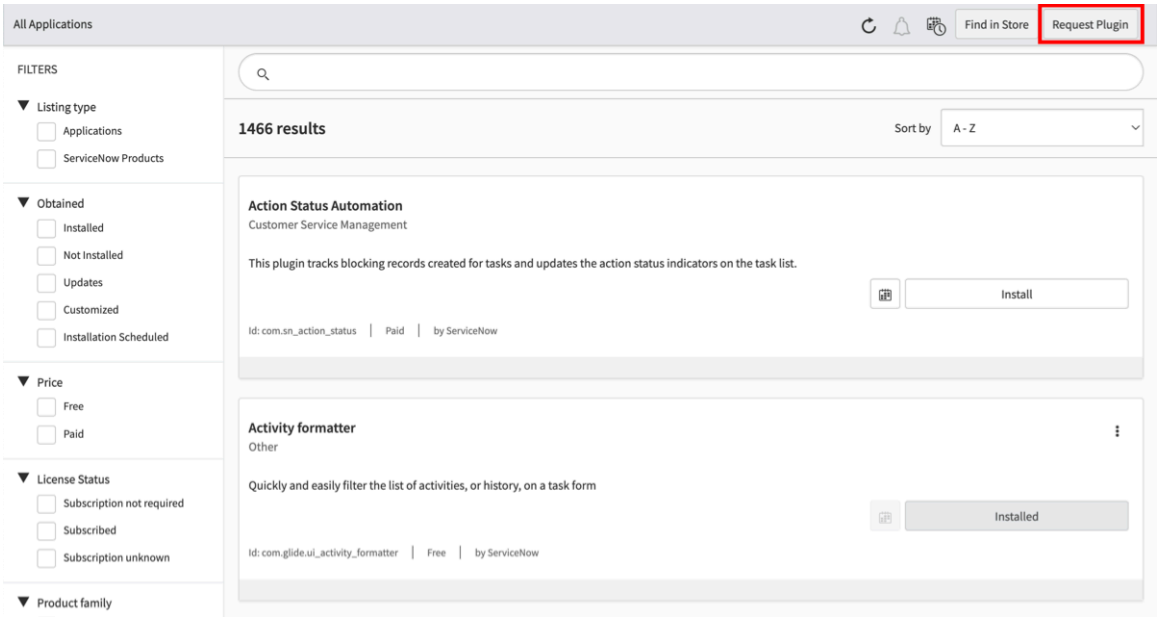
Encrypted Workflow Scratchpad activates these related plugins if they are not already active.

Plugins for Encrypted Workflow Scratchpad

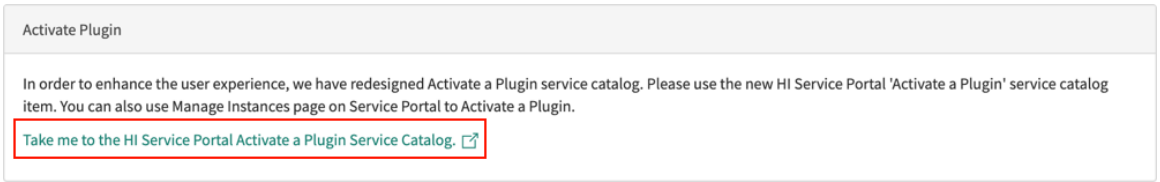
Plugin	Description
Encryption Support [com.glide.encryption]	Allows text fields and attached files to be encrypted.

Procedure

1. Navigate to **All** > **System Applications** > **All Available Applications** > **All**.
2. On the All Applications page, select **Request Plugin** to open the **Activate Plugin** form on Now Support.



3. On Now Support, select the link to access the Now Support Service Portal Service Catalog.



4. Select your instance.
5. Select **Actions > Activate Plugin**.
6. On the **Activate Plugin** form, provide the following information.

Activate Plugin form

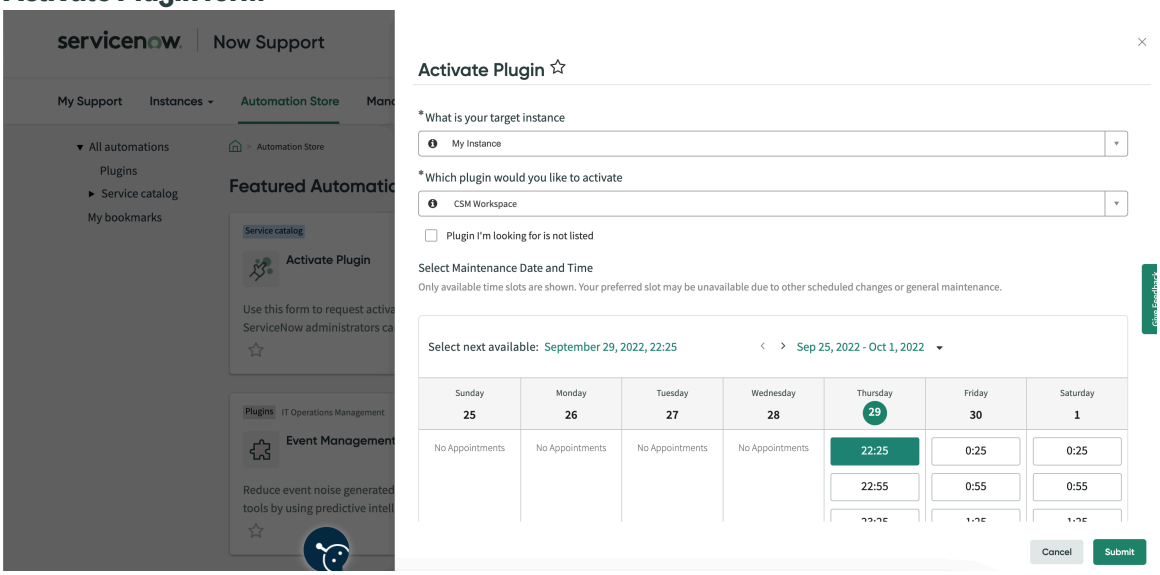
Field	Description
What is your target instance	Instance on which to activate the plugin.
Which plugin would you like to activate	Name of the plugin to activate. **Note:** If the system does not list the plugin you want or if you are activating the plugin on an OEM or on-premise instance, select the **Plugin I'm looking for is not listed** check box and then enter the name of the plugin.
Select Maintenance Date and Time	The date and time to activate the plugin.

Field	Description
	Note: Plugins are activated in two batches, once in the morning and once in the evening, on every business day in the US Pacific time zone. If the plugin must be activated at a specific time, enter the request in the **Reason/Comments** field.

Example

For example, see the following form to activate the CSM Workspace plugin on an instance named My Instance.

Activate Plugin form



7. Select **Submit**.
   For additional details about requesting a plugin, see Requesting a Plugin from the Service Catalog [KB0751715] article in the Now Support Knowledge Base.

What to do next

After plugin activation, resume all paused workflows. Refer to Pause and resume all or multiple workflows.

Related topics

List of plugins

Troubleshoot workflows

Troubleshooting tools for workflows enable administrators to isolate execution paths, compare contexts, and track incomplete activities.

The workflow timeline provides a visual representation of the workflow, including transitions and the elapsed time for each activity. A troubleshooting tool for highlighting execution paths helps users perform forensics on a workflow. The highlighting feature can group multiple execution paths in various colors and can isolate rollback processing. Use the workflow highlighter to

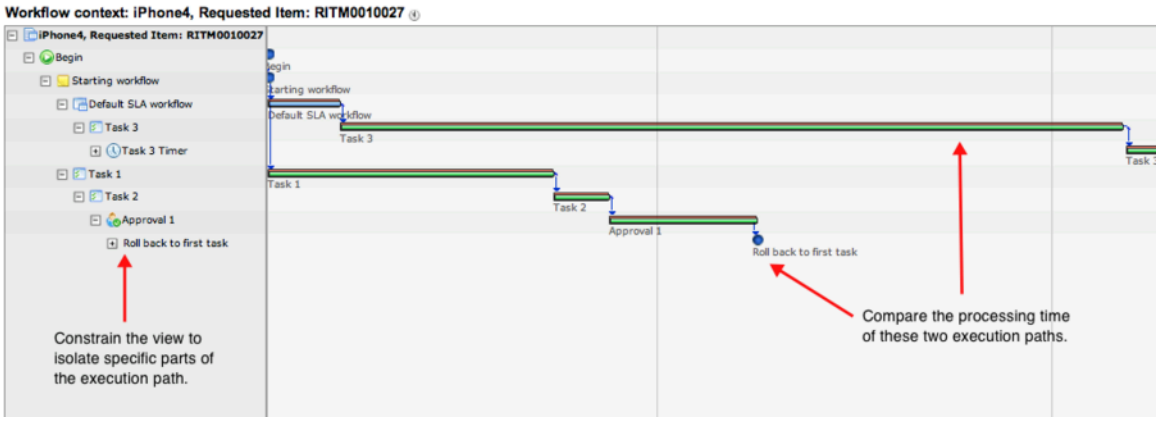
isolate incomplete tasks and approvals. You can also enable a workflow performance timing feature to troubleshoot slow workflows.

Execution path troubleshooting in timelines

Use the controls in a workflow timeline to isolate specific execution paths or compare multiple execution paths over time.

Timelines show how the activities in the workflow progressed in relation to one another over time. Isolate execution paths and follow transition lines between subflows and the main flow. Processing times provide a view of the workflow that you cannot get from the workflow diagram. Tooltips give precise information about each activity, such as duration.

Workflow timeline troubleshooting



Execution history table

Highlight execution paths and rollbacks to locate activities that may have been left in an unresolved state.

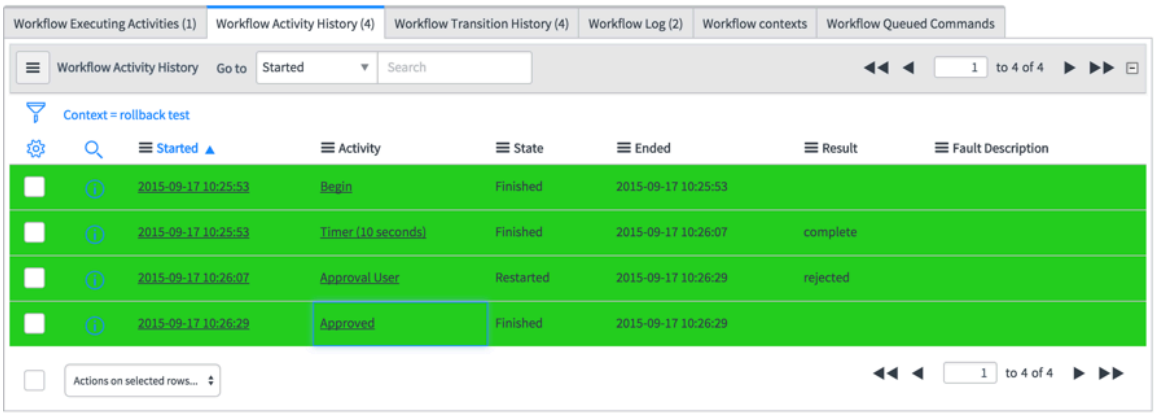
About this task

Rollbacks, cancels, and deletions during the execution of a workflow can prevent some activities from fully completing. Use highlighting in the execution history table to determine which activities in the workflow were left in an unresolved state.

Procedure

1. Run the workflow.
2. Navigate to **Workflow** > **All Contexts**.
3. Select a context to troubleshoot.
4. In the Workflow Context form, select the **Workflow Activity History** related list.
5. Right-click an activity and select **Workflow Debug** > **Toggle Execution Path Highlighting** from the context menu.

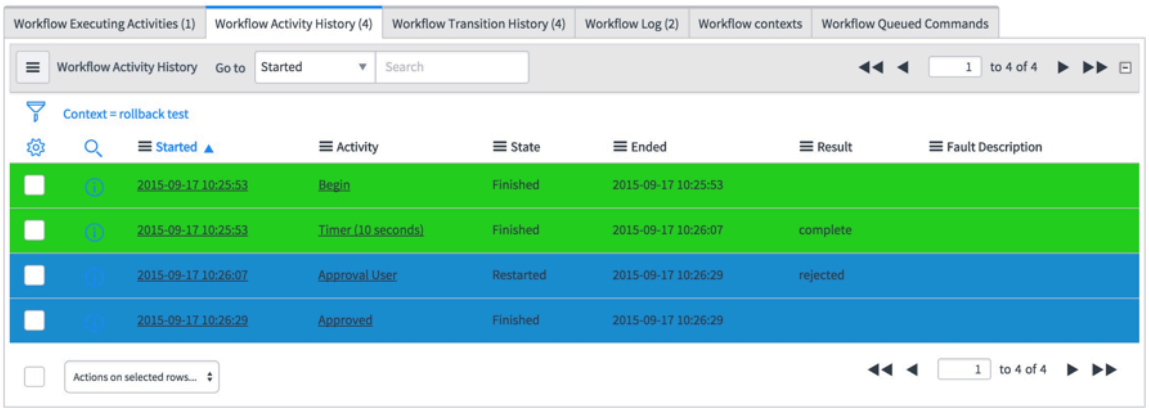
   All activities in that execution path are highlighted in a color selected by the platform. The debug path shows all activities that completed successfully during the workflow.



6. Right-click a rollback activity and select **Workflow Debug** > **Toggle Rollback Highlighting** from the context menu.

   The platform highlights the rollback path (restarted activities) in a different color. Each color represents a group of activities that were part of the same rollback execution. The highlighting includes the activity that initiated the rollback. If you right-click an activity that was not part of a rollback, no rows are highlighted.

   Note:
   The rollback activity itself appears in both execution path and rollback highlighting.



7. To remove highlighting, right-click in the list and select an option to clear execution path or rollback highlighting.

   You can clear individual rollback paths or all rollback highlighting.

Use the Workflow Operations Dashboard

On the Workflow Operations Dashboard, view and add widgets to help you monitor workflows. Review the performance of workflows and determine which workflows must be improved.

Before you begin

Role required: workflow_admin
The different levels of access are:

- **View**: View the dashboard and refresh report widgets.
- **Customize**: Refresh, add, delete, and rearrange widgets.

About this task

On the Workflow Operations Dashboard, you can click widget elements to view the records they represent. You can also add new report widgets that are not displayed by default.

Procedure

1. View the Workflow Operations Dashboard by navigating to **Workflow** > **Operations** > **Workflow Operations Dashboard**.

   The default reports on the dashboard include:

   - Workflow contexts providing run time metrics
   - Active workflows started during a specified time period
   - Workflows run between yesterday and today
   - Workflows by state for the current month
   - Aged workflow contexts for the past month

2. Find and add more widgets by selecting the add content icon (+) in the corner of the dashboard.

What to do next

If a workflow consistently appears as an outlier and there is no error causing the run time values, you can use its historical run time values to calculate a new estimated run time value. Then update the estimated run time in the workflow properties.

Workflow gauges

Multiple gauges are available to help you review the performance of workflows. You can add these gauges to the Workflow Operations Dashboard or custom homepages.

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.

Workflow gauges

Content	Description
ERT Dashboard Controls >	
Outlier Finished Workflows for ERT (Percentage Outlier)	Identifies workflows that finished, but did not finish within estimated runtime (ERT) values. Enter an outlier percentage to see workflows that ran outside the specified runtime range.

Workflow gauges (continued)

Content	Description
Outlier Long Running Workflows for ERT (Percentage Outlier)	Displays workflows running longer than the configured runtime threshold.
Workflow Dashboard >	
Workflows Without Current Record	Displays workflow contexts that do not have an associated current record.
Gauges > **Workflow Context**	
Number of Active Workflows Started Hourly Over Time (Yesterday)	Displays the total number of running workflows per hour over a given time period. By default, it displays the number of workflows that ran per hour over the previous day (yesterday).
Outlier Finished Workflows Not Cumulated to ERT	Displays workflow contexts that are finished outside the estimated runtime outlier value and not cumulated to estimated run time value.
Running Workflow Contexts	Displays the total number of running workflows.
Successfully Finished Workflows Cumulated to ERT	Displays the total number of successfully completed workflows whose running duration is cumulated to the estimated run time value.
Workflows by State (This Month)	Displays the total number of workflows run in a month grouped by the current state.
Aged Workflow Contexts (Running Since Last Month)	Displays the total number of workflow contexts running for a given period of time by workflow name. By default, it displays the total number of workflow contexts running over the last month.
Workflows Run Between Yesterday and Today (by Table)	Displays workflows that have run in the last day grouped by table name.

Workflow performance timing

The workflow engine can generate detailed performance timing data that is useful for troubleshooting slow workflows.

An administrator must enable this functionality.

When workflow performance timing is enabled, the workflow engine tracks key performance data, including various execution speed metrics. The Workflow Timing [wf_workflow_timing] table stores the data, with a record for each workflow context. The workflow timing record is updated when the workflow engine completes the workflow, waits for an activity to complete, or otherwise exits the workflow execution.

Workflow estimated run time properties

Administrators can enable the collection of workflow run time metrics by setting Estimated Run Time (ERT) properties.

Administrators can use ERT metrics to determine if workflows are running longer or shorter than expected and to identify errors in workflow processing. The system displays run time metrics on the Workflow Operations Dashboard.

Enable workflow performance timing

Workflow performance timing is disabled by default. You can create a system property to enable it.

Before you begin
Role required: admin

Procedure

Add a system property with the following specifications.

Field	Value
Name	glide.workflow.show_timing
Type	true | false
Value	true

Workflow activities

Workflow activity properties reference

Each activity performs a different task, such as running a script, sending notifications, or requesting approvals. Activities can succeed or fail, which can result in actions performed by other activities.

For information about configuring different types of activities, click an activity name in the list below or see Workflow activities reference.

Core activities provided in the base system

- Approval and rollback workflow activities
- Condition Workflow activities
- Notification workflow activity
- Notify workflow activites
- Subflow activities
- Task workflow activities
- Timer workflow activities
- Utility workflow activities

Activities provided with Orchestration

The following activities are included with Orchestration.

- Active Directory activity pack
- Orchestration activities

- PowerShell activities
- Puppet activities

Templates provided for creating custom activities

If Orchestration is active on your system, users with the proper roles can create custom activities using the ServiceNow Orchestration activity designer. For information about the templates Orchestration provides for creating custom activities that you can upload to the ServiceNow Store, see Activity designer components.

Workflow activities reference

Workflow activity reference, organized by category.

Each activity performs a different task, such as running a script, sending notifications, or requesting approvals.

Workflow runs activities as the user session that starts or advances them. Workflows started from record operations will run activities as the user session that performed the record operation. Workflows started from schedules or restarted from timers run activities as the System user. Workflows started from script calls run activities as the user session that started the script.

Approval and rollback activities

Approval and rollback activities generate and manage approvals. Not all workflows can include approval activities. For more information, read Approval and rollback workflow activities.

Note:
Approval activities run as the user whose actions match the approve or reject conditions the workflow was waiting for and advances the workflow.

Approval and rollback activities

Activity	Description
Approval Action workflow activity	The **Approval Action** activity performs an approval action on the current task.
Approval Coordinator workflow activity	The **Approval Coordinator** activity creates an approval whose outcome depends on the outcome of one or more child activities, including one or more **Approval - User**, **Approval - Group**, and/or **Manual Approval** activities.
Approval - Group workflow activity	The **Approval - Group** activity creates approval records for each member of a specified group.
Approval - User workflow activity	The **Approval - User** activity creates one or more individual user approvals.
Generate workflow activity	The **Generate** activity immediately creates task or approval records from any task or approval activities placed after the **Generate** activity in the workflow path. These pre-generated tasks and approvals start when the task and approval activities are reached during flow execution. This allows a task to have a set of associated pre-generated sequential tasks or approvals, but still require them to be completed in order.

Approval and rollback activities (continued)

Activity	Description
Manual Approvals workflow activity	The **Manual Approvals** activity watches and manages any approvals that users add manually outside of the workflow process. This activity only selects approvals that are in the Not requested state.
Rollback To workflow activity	The **Rollback To** activity transitions directly to the activity specified by the outgoing transition line arrow.

Condition activities

Condition activities provide conditional branching and logical operation functionality for workflows.

Note:
Condition activities run as the user whose actions match the conditions the workflow was waiting for and advances the workflow.

Condition activities

Activity	Description
If workflow activity	The **If** activity checks a condition or script to determine if a **Yes** or **No** transition should be taken.
Switch workflow activity	The **Switch** activity checks if the value of a passed field or variable is equivalent to one of several case values.
Wait for condition workflow activity	The **Wait for condition** activity causes the workflow to wait at this activity until the current record matches the specified condition.
Wait for WF Event workflow activity	The **Wait for WF Event** activity causes the workflow to wait at this activity until the specified event is fired.

Notify activities

Notify workflow activities manage calls and SMS messages in Notify.

Notify activities

Activity	Description
Forward call workflow activity	The **Forward Call** activity forwards a Notify call to an E.164-compliant phone number.
Input workflow activity	The **Input** activity creates a phone menu by presenting a list of options on a Notify call.
Hangup workflow activity	The **Hangup** activity disconnects an active Notify phone call.
Play workflow activity	The **Play** activity plays a sound file on a Notify call.
Record workflow activity	The **Record** workflow activity records audio from a user on a Notify call.

Notify activities (continued)

Activity	Description
Reject workflow	The **Reject** workflow activity rejects an incoming Notify call.
Say workflow activity	The **say** workflow activity allows you to play a message, using text to speech, on a Notify call.
Forward to notify client workflow activity	The **forward to notify client** workflow activity connects a phone call to a Notify WebRTC client.
Call workflow activity	The **Call** activity makes outbound phone calls using a Notify workflow. This workflow activity can be added to any table.
Join conference call workflow activity	The **Join Conference Call** activity connects an incoming or outgoing call to a Notify conference call.
Send SMS workflow activity	The **send SMS** workflow activity to send short text messages using Notify to users' phones. This workflow activity can be added to any table.
Queue workflow activity	The **Queue** activity places an active Notify call in a queue.

Notification activities

Notification workflow activities notify users of events that occur during the workflow.

Notification activities

Activity	Description
Create Event workflow activity	The **Create Event** activity adds an event to the event queue, but does not immediately fire the event.
Notification workflow activity	The **Notification** activity sends an email or SMS message to specified users or groups.

Subflow activities

Subflow activities run and manage workflows from a parent workflow.

Subflow activities

Activity	Description
Parallel Flow Launcher workflow activity	The **Parallel Flow Launcher** activity launches multiple subflows in parallel.

Task activities

Task activities create and modify workflow tasks.

Note:
Task activities run as the user whose actions complete the task the workflow was waiting for and advances the workflow.

Task activities

Activity	Description
Add Worknote workflow activity	The **Add Worknote** activity adds text to the *Worknotes* field of the current incident record.
Attachment Note workflow activity	The **Attachment Note** activity adds an attachment to the current record.
Catalog Task workflow activity	The **Catalog Task** activity creates a service catalog task record.
Create Task workflow activity	The **Create Task** activity generates a record on any of the tables that extend Task [task].

Timer activities

Timer activities pause workflows for set periods of time.

Note:
Timer activities run as the System user because the system scheduler advances the workflow.

Timer activities

Activity	Description
SLA Percentage Timer workflow activity	The **SLA Percentage Timer** activity pauses the workflow for a duration equal to a percentage of an SLA.
Timer workflow activity	The **Timer** activity pauses the workflow for a specified period of time.

Utility activities

Utility activities provide controls over the path of the workflow, and other useful tools.

Utility activities

Activity	Description
Branch workflow activity	The **Branch** activity splits the workflow into multiple transition paths from a single activity.
Join workflow activity	The **Join** activity unites multiple execution paths into one transition.
Lock workflow activity	The **Lock** activity prevents other instances of this workflow from continuing past this activity until the lock is released.
Log Message workflow activity	The **Log Message** activity writes a message to the workflow log.
Log Trace Message workflow activity	The **Log Trace Message** activity writes a trace message to the workflow log.

Utility activities (continued)

Activity	Description
REST Message legacy workflow activity	The legacy **REST Message** activity enables an administrator to override the REST endpoint or supply the variables configured in the REST Message module.
Return Value workflow activity	The **Return Value** activity returns a value to a parent workflow, when run from a subflow.
Run Script workflow activity	The **Run Script** activity runs the specified script in the scope of the workflow version.
Set Values workflow activity	The **Set Values** activity sets values on the current record when the workflow quiesces or ends.
SOAP Message legacy workflow activity	The legacy **SOAP Message** activity uses SOAP messages defined in the System Web Services plugin and can call the messages using a MID Server.
Turnstile workflow activity	The **Turnstile** activity limits how many times a workflow can pass through the same point.
Unlock workflow activity	The **Unlock** activity releases a lock that was previously placed by the **Lock** activity.

Activities provided with Orchestration

The following activities are included with Orchestration.

- Active Directory activity pack
- Orchestration activities
- PowerShell activities
- Puppet activities

Templates provided for creating custom activities

If Orchestration is active on your system, users with the proper roles can create custom activities using the ServiceNow Orchestration activity designer. For information about the templates Orchestration provides for creating custom activities that you can upload to the ServiceNow Store, see Orchestration custom activity templates

Approval and rollback workflow activities

Approval and rollback activities generate and manage approvals.

Approval and rollback activities are not available in some workflows.

- With two exceptions, approval and rollback activities are only available when the workflow runs on a table that extends Task. The exceptions are the **Approval - User** and **Approval Action** activities, which are available globally.
- Approval and rollback activities are available only if approval engines are turned off for the table on which the workflow runs. If approval engines are enabled for the table, approval activities appear greyed out and cannot be selected. To learn more about how workflow and approval engines interact, read Approval workflow activities and approval engines.

Approval Action workflow activity

The **Approval Action** activity performs an approval action on the current task.

Use this activity to mark the current task record as approved or rejected.

Note:
When an **Approval Action** activity is used to mark a task approved, the activity marks all pending approvals as **No Longer Required**. This activity behaves differently from **Set Values** or **Run Script** when used to set the **Approval** field's value.

Results

The result value of the activity is the final approval disposition selected by the approver. The result value can be **Approved** or **Rejected**. A workflow designer can assign a result value using the *`activity.result`* variable from within a script field of the activity.

Input variables

Input variables determine the initial behavior of the activity.

Approval Action activity input variables

Field	Description
Action	The action to perform on the task. Options are: • Mark task approved • Mark task rejected • Mark task requested • Disregard pending approvals: the system sets approval records to **no longer required** and marks the activity as **approved**.

Conditions

The conditions determine which transition runs after this activity.

Note:
Approval activities run as the user whose actions match the approve or reject conditions the workflow was waiting for and advances the workflow.

Approval Action activity conditions

Condition	Description
Always	The event or condition that causes the approval to move to the next workflow step.
Error	The event or condition that generates an error.
Skipped	The event or condition that allows a skipped approval.

States

The activity state tells the workflow engine what to do with the activity.

Approval Action activity states

State	Description
Executing	The workflow engine starts the `execute` function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Approval Coordinator workflow activity

The **Approval Coordinator** activity creates an approval whose outcome depends on the outcome of one or more child activities, including one or more **Approval - User**, **Approval - Group**, and/or **Manual Approval** activities.

Note:
This activity is only available when the workflow runs on a table that extends Task.

To create an **Approval Coordinator** activity, first drag the activity onto the workflow canvas, causing the activity form to display. On the activity form, fill in the appropriate fields, then click **Submit**.

After you click **Submit**, the activity appears on the workflow canvas. From there, specify the child activities by clicking the links that appear on the body of the activity.

When the **Approval Coordinator** activity completes, all pending approvals that were created by any of the **Approval Coordinator** approval activities are immediately set to **No Longer Required**. If a single user is called as an approver twice by the same workflow, such as when a single user is both a product approver and an executive approver, any approvals for that user after the first are skipped.

Results

The result value of the **Approval Coordinator** activity depends on the approval actions taken by the approvers and the approval conditions specified in the **Wait for** field. Possible result values are:

- Approved
- Rejected
- Deleted
- Cancelled

Input variables

Input variables determine the initial behavior of the activity.

Approval Coordinator activity input variables

Field	Description
Wait for	Indicate what to wait for to indicate that the approval activity is approved or rejected. Options are:

Approval Coordinator activity input variables (continued)

Field	Description
	• **Any child activity to be approved:** Any child activity (User, Group, or Manual Approval) that completes with a result of **approved** causes the **Approval Coordinator** activity to complete with a result of **approved**. • **All child activities to be approved:** All child activities of the **Approval Coordinator** activity must complete with a a result of **approved** to cause the **Approval Coordinator** activity to complete with a result of **approved**. • **The first approval or rejection from any child activity:** The first child activity that completes with a result of **approved** or **rejected** causes the **Approval Coordinator** activity to complete with the same result. • **Condition based on script:**Call a script to determine how to manage an approval or rejection.
When a rejection occurs Only appears if **Wait for** is set to **All child activities to be approved** or **Any child activity to be approved**.	Specify what the coordinator should do when it sees a rejection from any one of the child activities. Options are: • **Reject the approval:** Immediately complete the **Approval Coordinator** activity with a result of **rejected**. • **Wait for other responses before deciding:** Wait until we get other responses from other child activities before making an approval or rejection decision. This allows users to change their minds until a decision is made. In addition, if **Wait for** is set to **Any child activity to approve** then a single child activity completion with a result of **approved** will cause the **Approval Coordinator** activity to complete with a result of **approved** even if other child activities have completed with a result of **rejected**.
Approval script Only appears if **Wait for** is set to **Condition based on script**.	If the **Wait for** variable is set to **Condition based on script** this script is called to determine how to handle an approval or rejection. The script needs to set the variable *answer* to , approved or rejected to indicate the overall approval status for this approval. When called, the following variable is available to the script: `counts.total = total number of child approval activities that are part of this approval` `counts.approved = # of child approval activities that approved so far` `counts.rejected = # of child approval activities that rejected so far` `counts.requested = # of child approval activities that are pending approval`

Conditions

The conditions determine which transition runs after this activity.

Note:
Approval activities run as the user whose actions match the approve or reject conditions the workflow was waiting for and advances the workflow.

Approval Coordinator activity conditions

Condition	Description
Approved	The users from the groups have approved the request based on the **Wait for** rules.
Rejected	The users from the groups have rejected the request based on the **Wait for** rules.

States

The activity state tells the workflow engine what to do with the activity.

Approval Coordinator activity states

State	Description
Executing	The workflow engine starts the *execute* function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Approval - Group workflow activity

The **Approval - Group** activity creates approval records for each member of a specified group.

Note:
This activity is only available when the workflow runs on a table that extends Task.

The group approval is approved or rejected based on the user approvals, according to the logic specified in the **Wait For** field.

Results

The workflow designer can assign a result value using activity.result from within a script field of the activity. By default, the result value is the final approval disposition. This disposition depends on the approval actions take by the approvers and the approval conditions specified in the **Wait for** or **When Anyone Rejects** fields. Possible result values are:

- Approved
- Rejected
- Deleted
- Cancelled

Input variables

Input variables determine the initial behavior of the activity.

Approval - Group activity input variables

Field	Description
Approval - Group When Specify when this activity generates a group approval record.	
Condition	Conditions which, if met, cause the group approval to be generated. If the conditions are not met, the approval is skipped.
Approval - Group Approvers Specify the groups whose approval will be requested.	
Groups	The groups whose approval will be requested. To edit this field, click the lock icon. To select specific groups by name, use the lookup list. To select groups from field values on the current record at runtime, click the tree icon. Each member of the group will be assigned an individual approval record. If no group is selected, the activity automatically sets the approval to **Approved**.
Approval - Group Condition Specify how the activity decides to approve or reject the group approval, based on the responses from individual members of the group.	
Wait for	A choice between different approval logics to determine which individual approvals or rejections result in approval or rejection of the activity's approval. Options are: • **An approval from each group:** Any user from each group can approve and the first approval from each group causes the activity to complete with a result of **approved** (see below for how a rejection is handled). • **An approval from any group:** Any user from any group can approve and the first approval from any group causes the activity to complete with a result of **approved**. • **An approval from everyone from all groups:** All users from all groups must approve to cause the activity to complete with a result of **approved** (see below for how a rejection is handled). • **First response from each group:** The first approval or rejection from any user in each group is used to indicate the state of the group approval (see below for how a rejection is handled). Indicate what happens when any user rejects their approval request. Options are: • **Reject the approval:** Immediately complete the activity with a result of **rejected**.

Approval - Group activity input variables (continued)

Field	Description
	• **First response from any group:** The first approval or rejection from any user in any group causes the activity to complete with a result of **approved** or **rejected**. • **Condition based on script:** Each time a user approves or rejects, the **Approval script** is called to determine if the activity should complete.
Approval script Only appears when **Wait for** is set to `Condition based on script`.	If the **Wait for** variable is set to **Condition based on script** this script is called to determine how to handle an approval or rejection. The script needs to set the variable *answer* to approved or rejected to indicate the overall status for this approval. This script is responsible for setting the approval state for each group that is part of this approval activity before returning the overall approval state for all of the groups. When called, the following variables are available to the script for all the groups that are part of this approval activity: `counts.total = total number of groups that are part of this approval` `counts.approved = # of groups that approved so far` `counts.rejected = # of groups that rejected so far` `counts.requested = # of groups that are pending approval` `counts.not_requested = # of groups that are not pending approval` `counts.not_required = # of groups that approval is not required` And for each group: `groups[group_id].total = total number of users that are part of this group's approval` `groups[group_id].approved = # of users that approved so far` `groups[group_id].rejected = # of users that rejected so far` `groups[group_id].requested = # of users that are pending approval` `groups[group_id].not_requested = # of users that are not pending approval` `groups[group_id].not_required = # of users that approval is not required` `groups[group_id].approvalIDs[state] = array of user ids that are at the specified approval state` Note: Iterate the groups using:

Approval - Group activity input variables (continued)

Field	Description
	`for (var id in groups) { var group = groups[id]; ... group.total ... }` Note: Get group object using the following code (to get things like the group name being iterated on): `var objGroup = fncGetGroupObj(id); var strGroupName = objGroup.name;` `function fncGetGroupObj(sidGroupApproval) { var objGroupApproval = new GlideRecord('sysapproval_group'); objGroupApproval.get(sidGroupApproval); var objGroup = new GlideRecord('sys_user_group'); objGroup.get(objGroupApproval.assignment_group.sys_id); return objGroup; }` Approval scripts also allow computations. For example, if only half of the approvals are required: `if (counts.approved/counts.total > .49) { answer = 'approved'; } else if (counts.rejected/counts.total > .50) { answer = 'rejected'; }`
When anyone rejects	A choice between different approval logics to determine which individual rejections result in rejection of the activity's approval. Options are: • **Reject the approval:** Immediately complete the activity with a result of **rejected**. • **Wait for other responses before deciding:** Wait until we get other responses before making an approval or rejection decision. This allows users to change their mind until a decision is made.

Approval - Group activity input variables (continued)

Field	Description
	Note: If **Wait for** is set to **Anyone to approve**, then a single approval causes the activity to complete with a result of **approved**, even if one or more users reject.
Approval - Group Schedule Specify how workflow calculates the approval record's expected start date and due date. Once you've made a selection for 'Due date based on', and 'Schedule based on', the appropriate fields will display.	
Due date based on	Select how workflow determines the task's duration, due date, and schedule. • **A user specified duration:** The duration is based on a user specified value. • **A relative duration:** The duration is calculated from a relative duration (such as End of Next Business Day). • **A date/time or duration field:** The duration is based on the value of a field on the current record. • **Script:** The duration is returned by a script.
Duration Only appears when **Due date based on** is set to `A user specified duration`	The specific number of days and hours.
Relative duration Only appears when **Due date based on** is set to `A relative duration`	The general number and length of business days.
Due date field Only appears when **Due date based on** is set to `A date/time or duration field`	The date/time or duration field.
Due date script Only appears when **Due date based on** is set to `Script`	The script that sets 'answer' to the number of seconds for the duration.

Approval - Group activity input variables (continued)

Field	Description
Schedule based on	The basic schedule the timer uses to count working hours. If a schedule is specified, the duration will only be considered for times that are specified on the schedule. For example, if the duration is 2 hours and the workflow begins at 4:00pm on a schedule that is 8am - 5pm, then it ends at 9:00am the next day. The options are: • **This workflow's schedule:** The schedule uses workflow context date, time, and an optional **Time zone based on** value. • **A specific schedule:** The schedule uses a pre-defined **Schedule** and an optional **Time zone based on** value. • **A schedule field:** The schedule uses a value from a table and an optional **Time zone based on** value.
Schedule Only appears when **Schedule based on** is set to `A specific schedule`	The predefined **Schedule** from a list.
Schedule field Only appears when **Schedule based on** is set to `A schedule field`.	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.
Time zone based on	The time zone for calculating the duration. The time zone may be based on: • **No time zone:** Default. Workflow uses the GMT time zone. • **A specific time zone:** A specific **Time zone** that you choose from a choice list. • **A time zone field:** A **Time zone field** to track time duration from a field on the form.
Time zone Only appears when **Time zone based on** is set to `A specific time zone` .	Select the time zone you want from the choice list.
Time zone field Only appears when **Time zone based on** is set to	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.

Approval - Group activity input variables (continued)

Field	Description
`A time zone field.`	
Approval - Advanced If desired, write a script for determining additional users to request approvals from.	
Advanced	Select this check box to write a script for determining additional users to request approvals from. Use the **Additional groups script** to customize group approvals.
Additional groups script	If the **Advanced** check box is selected, this script is called to determine any additional group approvals to be created. The script needs to set the variable *`answer`* to a comma-separated list of group ids or an array of group ids to add as approver groups. For example: `answer = [];` `answer.push('id1');` `answer.push('id2');`

Conditions

The following conditions determine which transition runs after this activity.

Note:
Approval activities run as the user whose actions match the approve or reject conditions the workflow was waiting for and advances the workflow.

Approval - Group activity conditions

Condition	Description
Approved	The users from the groups have approved the request based on the **Wait for** rules.
Rejected	The users from the groups have rejected the request based on the **Wait for** rules.
Error	The event or condition that generates an error.
Skipped	The event or condition that allows a skipped approval.

States

The activity state tells the workflow engine what to do with the activity.

Approval - Group activity states

State	Description
Executing	The workflow engine starts the *`execute`* function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.

Approval - Group activity states (continued)

State	Description
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Approval - User workflow activity

The **Approval - User** activity creates one or more individual user approvals.

Results

The result value is the final approval disposition. This disposition depends on the approval actions take by the approvers and the approval conditions specified in the **Wait for** or **When Anyone Rejects** fields. Possible result values are:

- Approved
- Rejected
- Deleted
- Cancelled

Input variables

Approval - User activity input variables

Field	Description
Approval - User When Specify when this activity generates a user approval record.	
Condition	Conditions which, if met, cause the individual approval to be generated. If the conditions are not met, the approval is skipped.
Approval - User Approvers Specify the users whose approval will be requested.	
Users	The users for the approval. Use the tree icon to select user reference fields from the current record to create approvals, such as `${assigned_to}`. If no user is selected, the activity automatically sets the approval to **Approved**. **Note:** Workflow only manages approval records generated by the Approval User activity. After starting the workflow, newly added approvals do not affect the workflow context.
Groups	Groups whose members should also receive approvals. Note that this is different than the **Approval - Group** activity, which creates a group approval in addition to the individual approvals. The tree icon can be used to select group reference fields from the current record to create approvals, such as `${assignment_group}`.
Approval - User Condition	

Approval - User activity input variables (continued)

Field	Description
Specify how the activity decides to approve or reject the approval, based on the responses from individual approvers.	
Wait for	A choice between different approval logics to determine which individual approvals result in approval of the activity's approval. Options are: • **Anyone to approve:** Any user can approve and the first approval causes the activity to complete with a result of **approved**. • **Everyone to approve:** All users must approve (see below for how a rejection is handled). • **First response from anyone:** The first approval or rejection from any user causes the activity to complete. • **Condition based on script:** Each time a user approves or rejects, the **Approval script** is called to determine if the activity should complete.
When anyone rejects	A choice between different approval logics to determine which individual rejections result in rejection of the activity's approval. Options are: • **Reject the approval:** Immediately complete the activity with a result of **rejected**. • **Wait for other responses before deciding:** Wait until we get other responses before making an approval or rejection decision. This allows users to change their mind until a decision is made. **Note:** Note that if **Wait for** is set to **Anyone to approve** then a single approval will cause the activity to complete with a result of **approved** even if one or more users reject.
Approval Column	A string field for the name of the approval field on the table the workflow is running on. The default value is **approval**, which is the field on the Task table. **Note:** Use the field's name, not its label. If using any custom approval column fields and approval column journals, use Set Value activities in the workflow to set the custom Approval column fields.
Approval Journal Column	
Approval Script	If the **Wait for** variable is set to **Condition based on script** this script is called to determine how to handle an approval or rejection. The script needs to set the variable *answer* to approved or rejected to indicate the approval status for this approval. When called, the following information is available:

Approval - User activity input variables (continued)

Field	Description
	`counts.total = total number of users that are part of this approval` `counts.approved = # of users that approved so far` `counts.rejected = # of users that rejected so far` `counts.requested = # of users that are pending approval` `counts.not_requested = # of users that are not pending approval` `counts.not_required = # of users that approval is not required`
Approval - User Schedule Specify how workflow calculates the approval record's expected start date and due date. Once you've made a selection for 'Due date based on', and 'Schedule based on', the appropriate fields will display.	
Due date based on	Select how workflow determines the task's duration, due date, and schedule. • **A user specified duration:** The duration is based on a user specified value. • **A relative duration:** The duration is calculated from a relative duration (such as End of Next Business Day). • **A date/time or duration field:** The duration is based on the value of a field on the current record. • **Script:** The duration is returned by a script.
Duration Only appears when **Due date based on** is set to `A user specified duration`	The specific number of days and hours.
Relative duration Only appears when **Due date based on** is set to `A relative duration`	The general number and length of business days.
Due date field Only appears when **Due date based on** is set to `A date/time`	The date/time or duration field.

Approval - User activity input variables (continued)

Field	Description
`or duration field`	
Due date script Only appears when **Due date based on** is set to `Script`	The script that sets 'answer' to the number of seconds for the duration.
Schedule based on	The basic schedule the timer uses to count working hours. If a schedule is specified, the duration will only be considered for times that are specified on the schedule. For example, if the duration is 2 hours and the workflow begins at 4:00pm on a schedule that is 8am - 5pm, then it ends at 9:00am the next day. The options are: • **This workflow's schedule:** The schedule uses workflow context date, time, and an optional **Time zone based on** value. • **A specific schedule:** The schedule uses a pre-defined **Schedule** and an optional **Time zone based on** value. • **A schedule field:** The schedule uses a value from a table and an optional **Time zone based on** value.
Schedule Only appears when **Schedule based on** is set to `A specific schedule`	The predefined **Schedule** from a list.
Schedule field Only appears when **Schedule based on** is set to `A schedule field`.	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.
Time zone based on	The time zone for calculating the duration. The time zone may be based on: • **No time zone:** Default. Workflow uses the GMT time zone. • **A specific time zone:** A specific **Time zone** that you choose from a choice list. • **A time zone field:** A **Time zone field** to track time duration from a field on the form.
Time zone Only appears when **Time zone based on** is set to `A`	Select the time zone you want from the choice list.

Approval - User activity input variables (continued)

Field	Description
`specific time zone` .	
Time zone field Only appears when **Time zone based on** is set to `A time zone field.`	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.
Approval - Advanced If desired, write a script for determining additional users to request approvals from.	
Advanced	Select this check box to write a script for determining additional users to request approvals from.
Additional approvers script	If the **Advanced** check box is selected, this script is called to determine any additional user approvals to be created. The script needs to set the variable *`answer`* to a comma-separated list of user ids and group ids or an array of user and group ids to add as approvers. For example: `answer = [];` `answer.push('id1');` `answer.push('id2');`

Conditions

The following conditions determine which transition runs after this activity.

Note:
Approval activities run as the user whose actions match the approve or reject conditions the workflow was waiting for and advances the workflow.

Approval - User activity conditions

Condition	Description
Approved	The users approved the request based on the **Wait for** rules.
Rejected	The users rejected the request based on the **Wait for** rules.
Error	The event or condition that generates an error.
Skipped	The event or condition that allows a skipped approval.

States

The activity state tells the workflow engine what to do with the activity.

Approval - User activity states

State	Description
Executing	The workflow engine starts the `execute` function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Generate workflow activity

The **Generate** activity immediately creates task or approval records from any task or approval activities placed after the **Generate** activity in the workflow path. These pre-generated tasks and approvals start when the task and approval activities are reached during flow execution. This allows a task to have a set of associated pre-generated sequential tasks or approvals, but still require them to be completed in order.

Note:
This activity is only available when the workflow runs on a table that extends Task.

By default, the workflow does not create any tasks or approvals until it reaches them in the workflow.

The **Generate** activity follows all transitions through the workflow to each activity. For each activity:

- If it is a task activity, creates the task and sets:
  - The **State** to **Pending**
  - The **Expected Start Date**
  - The **Due Date**

  Note:
  Task activities run as the user whose actions complete the task the workflow was waiting for and advances the workflow.

- If it is an approval activity, creates the approvals and sets:
  - The approval **State** to **Not Requested**
  - The **Expected Start Date**
  - The **Due Date**

  Note:
  Approval activities run as the user whose actions match the approve or reject conditions the workflow was waiting for and advances the workflow.

Expected start dates and due dates are calculated based on the **Expected Duration** of all of the tasks and approvals between the **Generate** activity and the activity being updated. In the case of a branched path (between a **Branch** and **Join** activity), the longer duration will be used for any post-branch activities.

The **Generate** activity can be used more than once, and any tasks or approvals will be refreshed with updated information. This is useful in situations where the list of approvers or other

important information is still editable while the workflow is in process and it may be necessary to update or correct the generated approvals or tasks.

To exclude a set of activities from the **Generate** activity, select the **Skip during generate** check box on any condition and its transitions will not be followed during the generate process. By default, the following conditions have the **Skip during generate** check box selected:

- **Rejected** (for any of the approval activities)
- **No** condition of **If** activity
- **Continue** condition of **Turnstile** activity
- **Incomplete** condition of **Join** activity

Input variables

Input variables determine the initial behavior of the activity.

Generate activity input variables

Field	Description
Generate approvals	If selected, approvals are created when running the **Generate** activity. If cleared, the approvals are used to compute their estimated duration, but no approvals are created.
Generate tasks	If selected, tasks are created when running the **Generate** activity. If cleared, the tasks are used to compute their estimated duration, but no tasks are created.

States

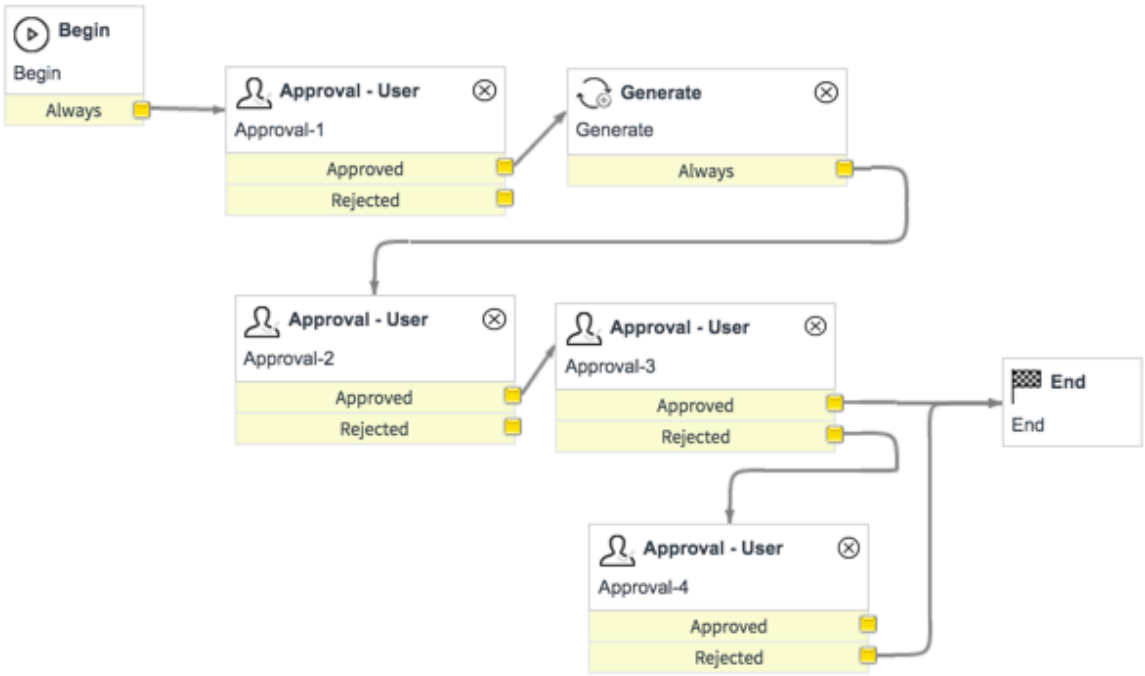
The activity state tells the workflow engine what to do with the activity.

Generate activity states

State	Description
Executing	The activity is executing.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.

Example

Generate workflow



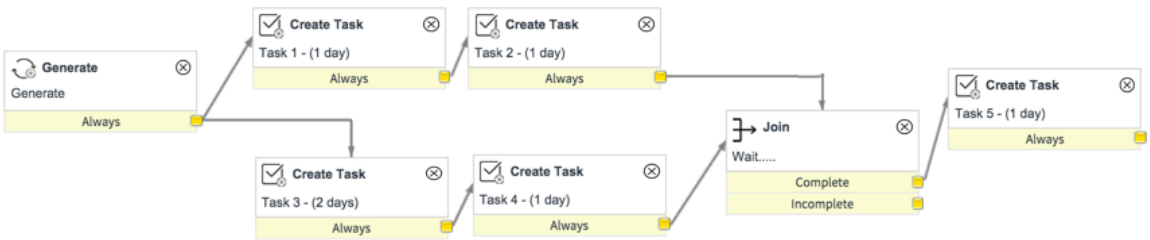
In this example, the following approvals are generated:

- Approval-2
- Approval-3

Approval-4 is skipped since the **Rejected** condition of Approval-3 has **Skip during generate** selected.

Here is an example of using the **Generate** activity that describes the expected start and due dates:

Generate workflow start and due date



In this example, if the **Generate** activity is run on Jan 1, 2016, the following expected start dates and due dates would be set for the generated tasks.

Task	Expected Start Date	Reason	Due Date
Task 1 (1 day)	Jan. 1, 2016		Jan. 2, 2016
Task 2 (1 day)	Jan. 2, 2016	Task 1 is 1 day	Jan. 3, 2016
Task 3 (2 days)	Jan. 1, 2016		Jan. 3, 2016
Task 4 (1 day)	Jan. 3, 2016	Task 3 is 2 days	Jan. 4, 2016

Task	Expected Start Date	Reason	Due Date
Task 5 (1 day)	Jan. 4, 2016	Task 4 ends the latest before the Join	Jan. 5, 2016

Notice that Task 5 starts on Jan. 4, 2016 since the longest path (based on due dates) to the **Join** is Task 3/Task 4.

Manual Approvals workflow activity

The **Manual Approvals** activity watches and manages any approvals that users add manually outside of the workflow process. This activity only selects approvals that are in the Not requested state.

Note:
This activity is only available when the workflow runs on a table that extends Task.

If there are no pending manual approvals when this activity executes, the activity immediately completes with a result of **approved**. This activity does not create approval records. Use this activity to pause the workflow when a user adds a manual approval to a record with an associated workflow, and it is in the Not requested state. The workflow waits for the approval to be closed before proceeding.

Results

The workflow designer can assign a result value using `activity.result` from within a script field of the activity. By default, the result value of the activity is the final approval disposition determined by the approval actions take by the approvers. Possible result values are:

- Approved
- Rejected
- Deleted
- Cancelled
- Error

Input variables

Input variables determine the initial behavior of the activity.

Manual Approvals activity input variables

Field	Description
Wait for	Indicate what to wait for to indicate that the approval activity is approved or rejected. Options are: • **Any manual user or group approval:** Any user can approve and the first approval causes the activity to complete with a result of **approved**. • **All manual user or group approvals:** All users must approve (see below for how a rejection is handled). • **The first response from any manual approval:** The first approval or rejection from any user causes the activity to complete.
When anyone rejects	Indicate what happens when any user rejects their approval request. Options are:

Manual Approvals activity input variables (continued)

Field	Description
	• **Reject the approval:** Immediately complete the activity with a result of **rejected**. • **Wait for other responses before deciding:** Wait until we get other responses before making an approval or rejection decision. This allows users to change their mind until a decision is made. In addition, if **Wait for** is set to **Anyone to approve** then a single approval will cause the activity to complete with a result of **approved** even if one or more users reject.

Conditions

The conditions determine which transition runs after this activity.

Note:
Approval activities run as the user whose actions match the approve or reject conditions the workflow was waiting for and advances the workflow.

Manual Approvals activity conditions

Condition	Description
Approved	The users from the groups have approved the request based on the **Wait for** rules.
Rejected	The users from the groups have rejected the request based on the **Wait for** rules.

States

The activity state tells the workflow engine what to do with the activity.

Manual Approvals activity states

State	Description
Executing	The workflow engine starts the `execute` function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Rollback To workflow activity

The **Rollback To** activity transitions directly to the activity specified by the outgoing transition line arrow.

Note:
This activity is only available when the workflow runs on a table that extends Task.

Rollback To determines which activities to reset based on the actual workflow sequence (transition line attachments) of activities between itself and the transitioned to activity, not the execution order. **Rollback To** then marks all the approvals that have transitioned between the

rollback and the transitioned to activity as **Not Yet Requested** and the tasks as either **Open** or **Pending**.

Use the **Rollback To** activity for all workflows in which multiple rollbacks are required. **Rollback To** has no variables.

Conditions

The conditions determine which transition runs after this activity.

Rollback To activity conditions

Condition	Description
Always	The event or condition that causes the approval to revert to the previous workflow step.
Error	The event or condition that generates an error.
Skipped	The event or condition that allows a skipped approval.

States

The activity state tells the workflow engine what to do with the activity.

Rollback To activity states

State	Description
Executing	The workflow engine starts the `execute` function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Rollback To behavior

The **Rollback To** activity transitions directly to the activity specified by the transition line arrow.

Use the **Rollback To** activity for all workflows that use multiple or nested rollbacks. **Rollback To** resets the targeted task (the direct transition) to **Open**. All tasks that have executed between the **Rollback To** activity and the targeted task (rolled back task) are set to **Pending**.

(0) Begin

(1) Log Message

(2) Task 1

(3) Task 3

(4) Task 2

(5) Timer

(6) Approval 1

(7) Roll back to first task

(8) Task 1

(9) Task 2

(10) Approval 1

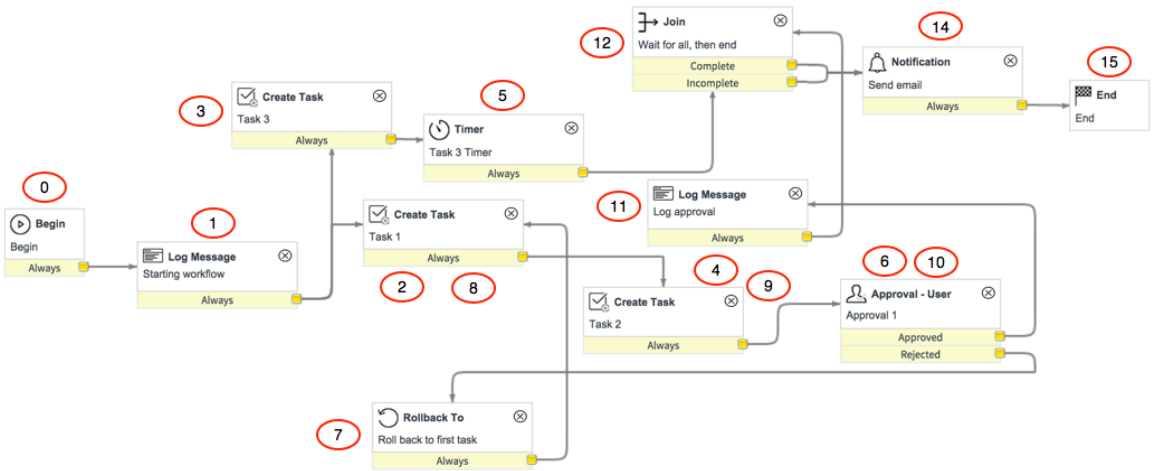
(11) Log approval

(12) Join

(14) Send email

(15) End

Rollback to workflow



Transition history

The state of (3) Task 3 does not change, since this activity does not directly transition from the rollback target activity. To see what activities were rolled back, select the **Workflow Transition History** related list and look at the **Rolled back** column.

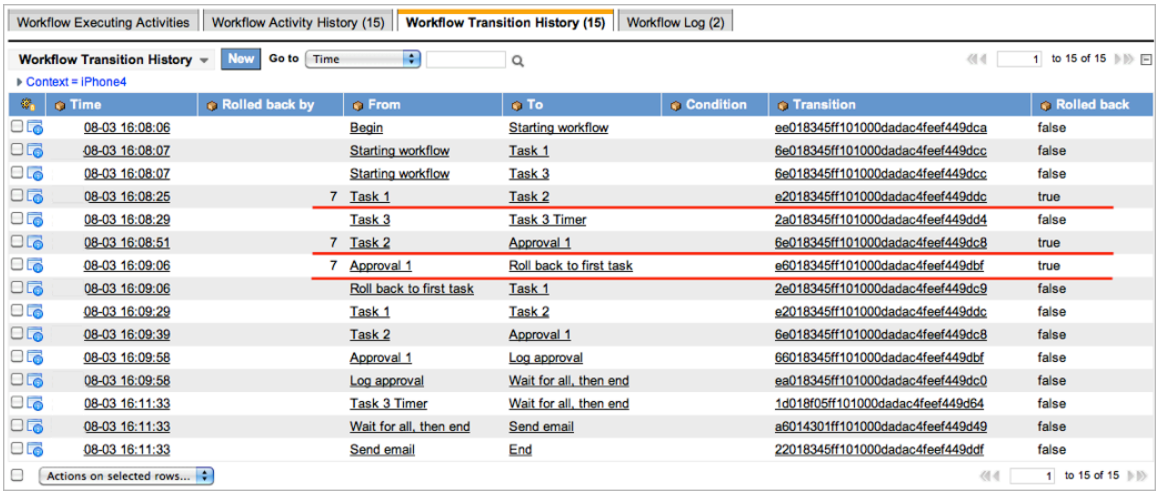
The **Rollback To** activity (7) updates the following activities:

(8) Task 1: reset to **Open**

(9) Task 2: reset to **Pending**

(10) Approval 1: reset to **Not Yet Requested**

Rollback To workflow transition



Workflow Executing Activities | Workflow Activity History (15) | **Workflow Transition History (15)** | Workflow Log (2)

Workflow Transition History — New — Go to Time — 1 to 15 of 15

Context = iPhone4

Time	Rolled back by	From	To	Condition	Transition	Rolled back
08-03 16:08:06		Begin	Starting workflow		ee018345ff101000dadac4feef449dca	false
08-03 16:08:07		Starting workflow	Task 1		6e018345ff101000dadac4feef449dcc	false
08-03 16:08:07		Starting workflow	Task 3		6e018345ff101000dadac4feef449dcc	false
08-03 16:08:25	7	Task 1	Task 2		e2018345ff101000dadac4feef449ddc	true
08-03 16:08:29		Task 3	Task 3 Timer		2a018345ff101000dadac4feef449dd4	false
08-03 16:08:51	7	Task 2	Approval 1		6e018345ff101000dadac4feef449dc8	true
08-03 16:09:06	7	Approval 1	Roll back to first task		e6018345ff101000dadac4feef449dbf	true
08-03 16:09:06		Roll back to first task	Task 1		2e018345ff101000dadac4feef449dc9	false
08-03 16:09:29		Task 1	Task 2		e2018345ff101000dadac4feef449ddc	false
08-03 16:09:39		Task 2	Approval 1		6e018345ff101000dadac4feef449dc8	false
08-03 16:09:58		Approval 1	Log approval		66018345ff101000dadac4feef449dbf	false
08-03 16:09:58		Log approval	Wait for all, then end		ea018345ff101000dadac4feef449dc0	false
08-03 16:11:33		Task 3 Timer	Wait for all, then end		1d018f05ff101000dadac4feef449d64	false
08-03 16:11:33		Wait for all, then end	Send email		a6014301ff101000dadac4feef449d49	false
08-03 16:11:33		Send email	End		22018345ff101000dadac4feef449ddf	false

Actions on selected rows... — 1 to 15 of 15

Rollback To activity

When conditions in a workflow trigger a **Rollback To** activity, the workflow moves processing backward to a specified activity in the workflow and resets certain activities that have already executed back to their original state. This is useful when handling an unexpected failure or as part of a programmed logical flow.

When an activity is reset during a workflow rollback, the following happens:

- Approvals are reset to **Not Requested**.
- Tasks are reset to either **Open** or **Pending**. A rollback workflow path cannot create new tasks.

Activities that perform external system operations, such as deleting a file or sending an email, are not rolled back. Only approval and task activity states are reset.

A workflow can contain a single rollback, multiple rollbacks, or nested rollbacks in more complex workflows. The **Rollback To** activity resets activities based on the actual workflow sequence (transition line attachments) of activities between itself and the transitioned to activity, rather than using the execution order to determine where processing should restart.

Condition Workflow activities

Condition activities provide conditional branching and logical operation functionality for workflows.

If workflow activity

The **If** activity checks a condition or script to determine if a **Yes** or **No** transition should be taken.

If the workflow creator specifies both the **Condition** and the **Advanced** script, both must evaluate successfully for activity to take the **Yes** transition.

Results

The workflow designer can assign a result value using activity.result from within the **Script** field on the activity record. By default, the result value of the activity is the final result of the condition or script specified. Possible result values are:

- Yes
- No

Input variables

The following variables determine the behavior of the activity.

Note:
Condition activities run as the user whose actions match the conditions the workflow was waiting for and advances the workflow.

If activity input variables

Field	Description
Condition	If specified and the current record matches the condition, the **Yes** transition is taken.
Advanced and Script	To specify a script, select the **Advanced** check box. You may then enter a script that is evaluated. If your script sets the variable answer to `yes`, then the **Yes** transition is taken. Otherwise, the **No** transition is taken.

Conditions

The following conditions determine which transition comes after the activity.

If activity conditions

Condition	
Yes	Taken when the condition, if specified, matches and the **Advanced** script, if specified, returns yes.
No	Taken when either the condition does not match or the **Advanced** script, if specified, returns no.

States

The activity state tells the workflow engine what to do with the activity.

If activity states

State	Description
Executing	The workflow engine knows to start the *`onExecute`* function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activty, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Switch workflow activity

The **Switch** activity checks if the value of a passed field or variable is equivalent to one of several case values.

The switch activity behavior is similar to a switch statement in Java.

When creating a switch activity, select a **Variable** or **Field** to check against activity conditions for a matching field value. When the value passed in this variable or field matches the value defined in the **Condition** field of the activity condition, the workflow progresses through that activity condition.

Note:
Condition activities run as the user whose actions match the conditions the workflow was waiting for and advances the workflow.

Results

The variable or field selected in the **Variable** or **Field** activity variable determines the possible result values.

Input variables

The following variables determine the behavior of the activity.

Switch activity input variables

Field	Description
Type	Select **Variable** or **Field** as the type of value to check against available conditions. This selection sets the label and available options for the other field.
Variable or Field	Select the source of the value compared against the switch activity conditions. The field label and available options depend on the **Type** selection. • **Variable:** select any service catalog variable. • **Field:** select any field from the **Table** defined in the workflow properties.

States

The activity state tells the workflow engine what to do with the activity.

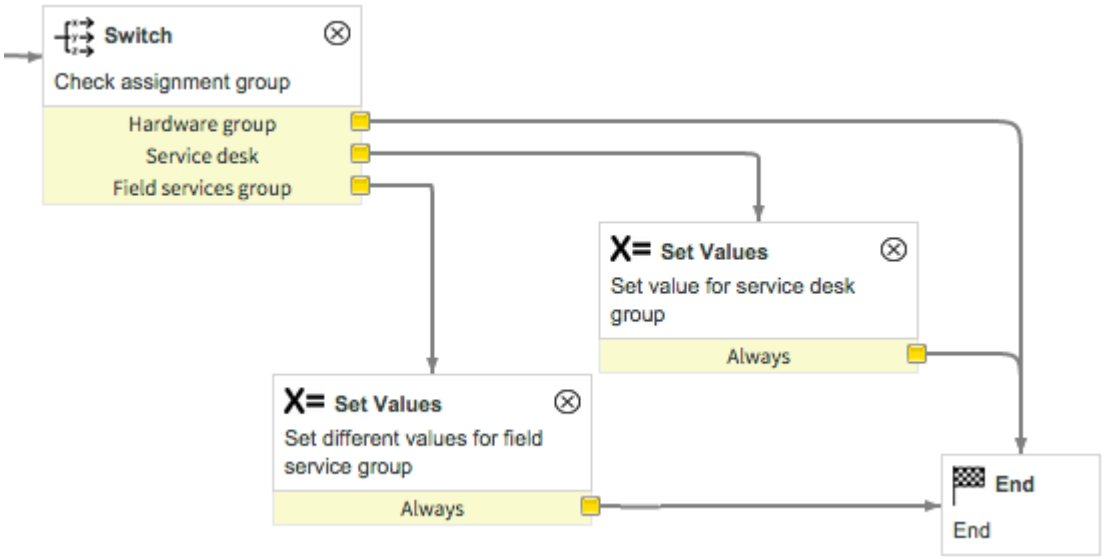
Switch activity states

State	Description
Executing	The workflow engine knows to start the `onExecute` function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Example

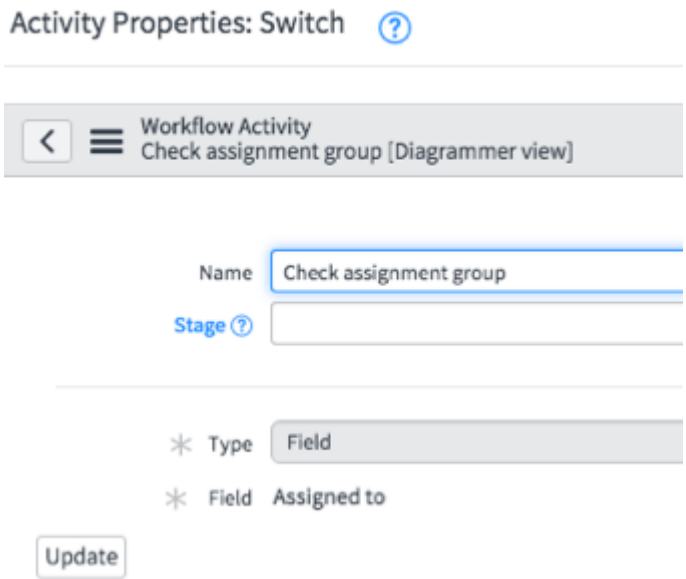
You can create a switch activity that sets different field values on an incident based on the Assignment group of the incident record.

Switch activity example



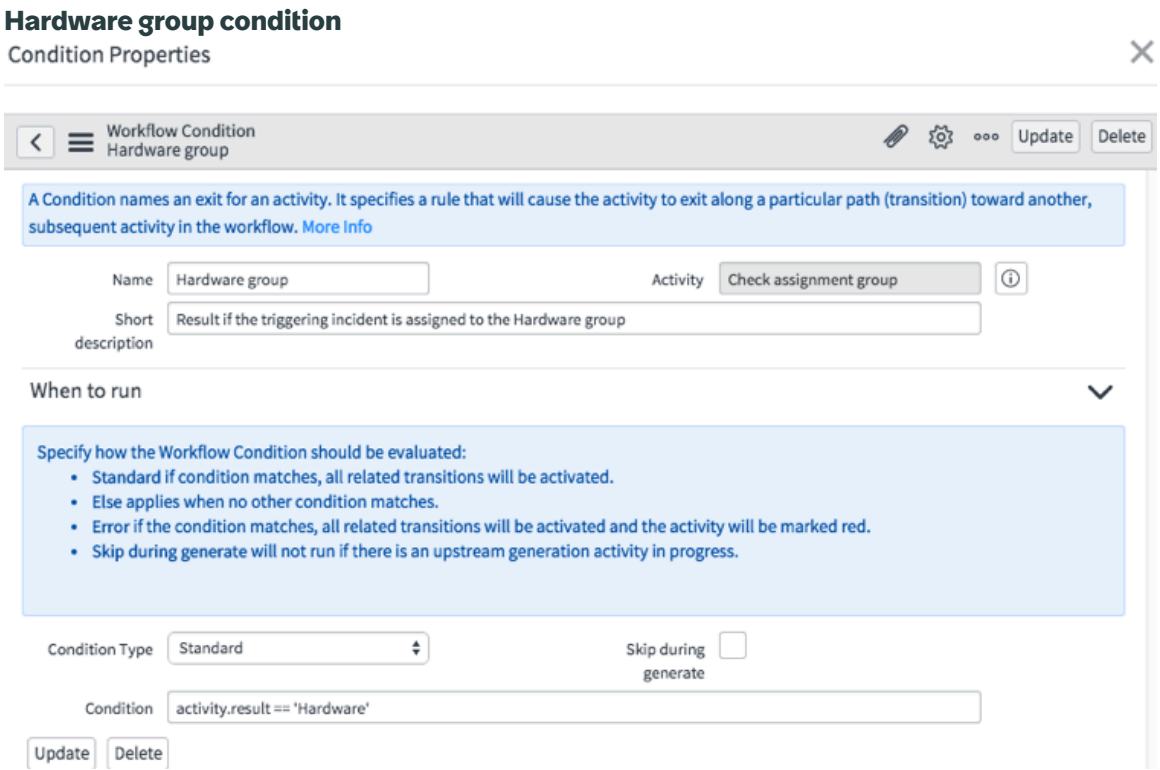
The **Field** selected is the incident **Assigned to** field.

Switch activity Field



If the value of the **Assigned to** field of the workflow-triggering incident is **Service Desk** or **Field Services**, the workflow populates values on the incident record before continuing. If the **Assigned to** value is **Hardware**, the workflow continues without populating any field values.

Condition type (Standard, Else, or Error) is used. For more information, see Manage workflow activity conditions.



Wait for condition workflow activity

The **Wait for condition** activity causes the workflow to wait at this activity until the current record matches the specified condition.

The workflow evaluates the **Wait for condition** activity each time the current record is updated. Use this activity to pause a workflow indefinitely until a particular criteria is met by a record update. To pause a workflow for a timed duration see Timer workflow activities.

For workflow to consider the condition met, all conditions specified – whether in the builder or in a script – must be true.

Note:
A **Wait for condition activity** should only be used to wait for an external event such as a record update, and not one from a workflow setting a value. If you have a workflow setting a value and want to wait for that same field to be seen as 'changed,' try inserting a one-second timer.

Results

The workflow designer can assign a result value using `activity.result` from within a script field of the activity. The activity transitions when the result value is true.

Input variables

The following variables determine the behavior of the activity.

Note:
Condition activities run as the user whose actions match the conditions the workflow was waiting for and advances the workflow.

Wait for condition activity input variables

Field	Description
Condition	The workflow is paused at this activity until this condition matches the current record.
Condition script	If specified, the workflow is paused at this activity until this script sets the *answer* variable to true.
Enable Timeout	Option to limit the amount of time that the workflow waits for the activity to be completed before continuing. **Note:** Use the **Enable timeout** option to prevent this activity from continuing to run. If the condition to continue is never met, a timeout value specifies when the system skips the Wait for Condition activity and goes to the next item in the workflow. You must set a Duration value to enable a timeout. You can also select a Schedule if you want to compute the duration end date based on a specific work schedule.
Duration	Amount of time that the workflow waits before continuing when the **Enable timeout** option is selected. Enter the time to wait in hours, minutes, and seconds. If you leave this field empty, the workflow does not wait.

States

The activity state tells the workflow engine what to do with the activity.

Wait for condition activity states

State	Description
Executing	The workflow engine knows to start the *onExecute* function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activty, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Wait for WF Event workflow activity

The **Wait for WF Event** activity causes the workflow to wait at this activity until the specified event is fired.

Use this activity to wait for another activity to fire an event. Events from other activities are fired in a script using the workflow.fireEvent('eventName') API call.

Results

The workflow designer can assign a result value using `activity.result` from within a script field of the activity. This activity transitions when the specified event fires.

Input variables

The following variables determine the behavior of the activity.

Note:
Condition activities run as the user whose actions match the conditions the workflow was waiting for and advances the workflow.

Wait For WF Event activity input variables

Field	Description
Wait for Event	An event name to trigger the workflow.

States

The activity state tells the workflow engine what to do with the activity.

Wait For WF Event activity states

State	Description
Executing	The workflow engine knows to start the `onExecute` function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Workflow notification activities

Notification workflow activities notify users of events that occur during the workflow.

Create Event workflow activity

The **Create Event** activity adds an event to the event queue, but does not immediately fire the event.

The event processor typically processes the event within one minute. This activity triggers any business rules or email notifications that would normally be triggered by the event.

For more information about creating and using system events, see System Events.

Results

Finished: the activity added the event to the event queue.

Input variables

Create Event activity input variables

Field	Description
Event name	The name of the event to create. From the Event Name lookup list, select the event to add to the queue. If the event requires parameters, specify them in the Parameter script field.

Create Event activity input variables (continued)

Field	Description
Parameter 1	The first parameter of the event. **Note:** If this parameter is a string value, it must be within quotes (" ").
Parameter 2	The second event parameter. **Note:** If this parameter is a string value, it must be within quotes (" ").

Notification workflow activity

The **Notification** activity sends an email or SMS message to specified users or groups.

Input variables

Update the recipients list or delete the activity from the workflow if you want to stop the notifications.

Notification activity input variables

Field	Description
Addressees	
To	The users who will be recipients of the email.
To (groups)	The members of the groups that will be recipients of the email.
Advanced	If selected, the script in the **To (script)** field is called to specify additional recipients of the email.
To (script)	If **Advanced** is selected, this script is called and should set the variable `answer` to a comma-separated list of user or group sys_ids that you want to add as recipients of the email.
Message	
Subject	The subject line of the email.
Message	The email body that is sent. To include the value of a field in the message body, place the cursor at the point in the text where you want the field's value inserted. Then click the + icon next to **Fields** and select the field you want.

States

The activity state tells the workflow engine what to do with the activity.

Notification activity states

Field	Description
Executing	The workflow engine knows to start the `run` function of the activity.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.

Timer workflow activities

Timer activities pause workflows for set periods of time.

Related topics

Use multiple timer activities in one workflow

SLA Percentage Timer workflow activity

The **SLA Percentage Timer** activity pauses the workflow for a duration equal to a percentage of an SLA.

A workflow must run on the Task SLA table to use this activity.

Note:
Timer activities run as the System user because the system scheduler advances the workflow.

Results

SLA Percentage Timer activity results

Result	Description
Complete	The activity successfully reached the specified duration
Cancelled	The activity or workflow was canceled before the timer reached the specified duration

Input variables

Input variables determine the initial behavior of the activity.

SLA Percentage Timer activity input variables

Field	Description
Percentage	The duration to pause the workflow for, as a percentage of the current SLA

States

The activity state tells the workflow engine what to do with the activity.

SLA Percentage Timer states

State	Description
Executing	The activity is in this state very briefly while initializing, after which it immediately changes to **Waiting**.
Waiting	The workflow engine waits until the SLA reaches the specified percentage. The engine then transitions the workflow to the next activity.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Timer workflow activity

The **Timer** activity pauses the workflow for a specified period of time.

This duration can be an absolute time value or a relative value based on a defined Creating and using schedules. It is best to adjust the **Duration** so the workflow can progress in a timely manner. To pause a workflow indefinitely until a condition is met, see wait for condition.

Note:
Timer activities run as the System user because the system scheduler advances the workflow.

Results

Timer activity results

Result	Description
Complete	The activity successfully reached the specified duration.
Cancelled	The activity or workflow was canceled before the timer reached the specified duration.

Input variables

Input variables determine the initial behavior of the activity.

Timer activity input variables

Field	Description
Timer Information	
Timer based on	How the timer duration is computed. The options are: • **A user specified duration:** The duration is based on the **Duration** fields, such as days and hours. • **A relative duration:** The duration is based on the **Relative duration** (such as End of Next Business Day) and **Wait** fields.

Timer activity input variables (continued)

Field	Description
	• **A date/time or duration field:** The duration is based on the **Field** value and the **Wait** field. • **Script:** The duration is based on a script that returns the number of seconds.
Duration Appears only when **Timer based on** is **A user specified duration**.	The specific number of days and hours to wait before proceeding to the next activity in the workflow.
Relative duration Appears only when **Timer based on** is **A relative duration**	The general number and length of business days to wait before progressing to the next workflow activity. .
Wait Appears only when **Timer based on** is **A relative duration** or **A date/time or duration field**.	An additional timer adjustment when **Timer based on** is **A relative duration** or **A date/time or duration field**. The options are: • **The full duration:** No modification of the calculated duration. • **A % of the duration:** The duration is adjusted by multiplying the number of seconds by the (**Percentage** / 100). • **Some time before:** The duration is shortened by the **Time before** days and hours. • **Some time after:** The duration is lengthened by the **Time after** days and hours.
Percentage Appears only when **Timer based on** is **A relative duration** or **A date/time or duration field**.	The **Wait** percentage value when **Timer based on** is **A relative duration** or **A date/time or duration field**.
Time before Appears only when **Timer based on** is **A relative duration** or **A date/time or duration field** and **Wait** is **Some time before**.	The modifier time value when **Wait** is **Some time before**.
Time after Appears only when **Timer based on** is **A relative duration** or **A date/time or duration**	The modifier time value when **Wait** is **Some time after**.

Timer activity input variables (continued)

Field	Description
field and **Wait** is **Some time after**.	
Field Appears only when **Timer based on** is **A date/time or duration field**.	The date/time or duration field that contains the elapsed wait-time before moving to the next workflow activity.
Script Appears only when **Timer based on** is **Script**	The script that sets 'answer' to the number of seconds for the duration.
Timer Schedule	
Schedule based on	The basic schedule the timer uses to count working hours. If a schedule is specified, the duration will only be considered for times that are specified on the schedule. For example, if the duration is 2 hours and the workflow begins at 4:00pm on a schedule that is 8am - 5pm, then it ends at 9:00am the next day. The options are: • **This workflow's schedule:** The schedule uses workflow context date, time, and an optional **Time zone based on** value. • **A specific schedule:** The schedule uses a pre-defined **Schedule** and an optional **Time zone based on** value. • **A schedule field:** The schedule uses a value from a table and an optional **Time zone based on** value.
Schedule Appears only when **Schedule based on** is **A specific schedule**.	The pre-defined **Schedule** from a list.
Schedule field Appears only when **Schedule based on** is **A schedule field**.	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.
Timer Time Zone	
Time zone based on	The time zone for calculating the duration. The time zone may be based on: • **No time zone:** Default. Workflow uses the GMT time zone. • **A specific time zone:** A predefined **Time zone**. • **A time zone field:** A **Time zone field** to track time duration from a field on the form.

Timer activity input variables (continued)

Field	Description
Time zone Appears only when **Time zone based on** is **A specific time zone**.	The predefined time zone.
Time zone field Appears only when **Time zone based on** is **A time zone field**.	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.

States

The activity state tells the workflow engine what to do with the activity.

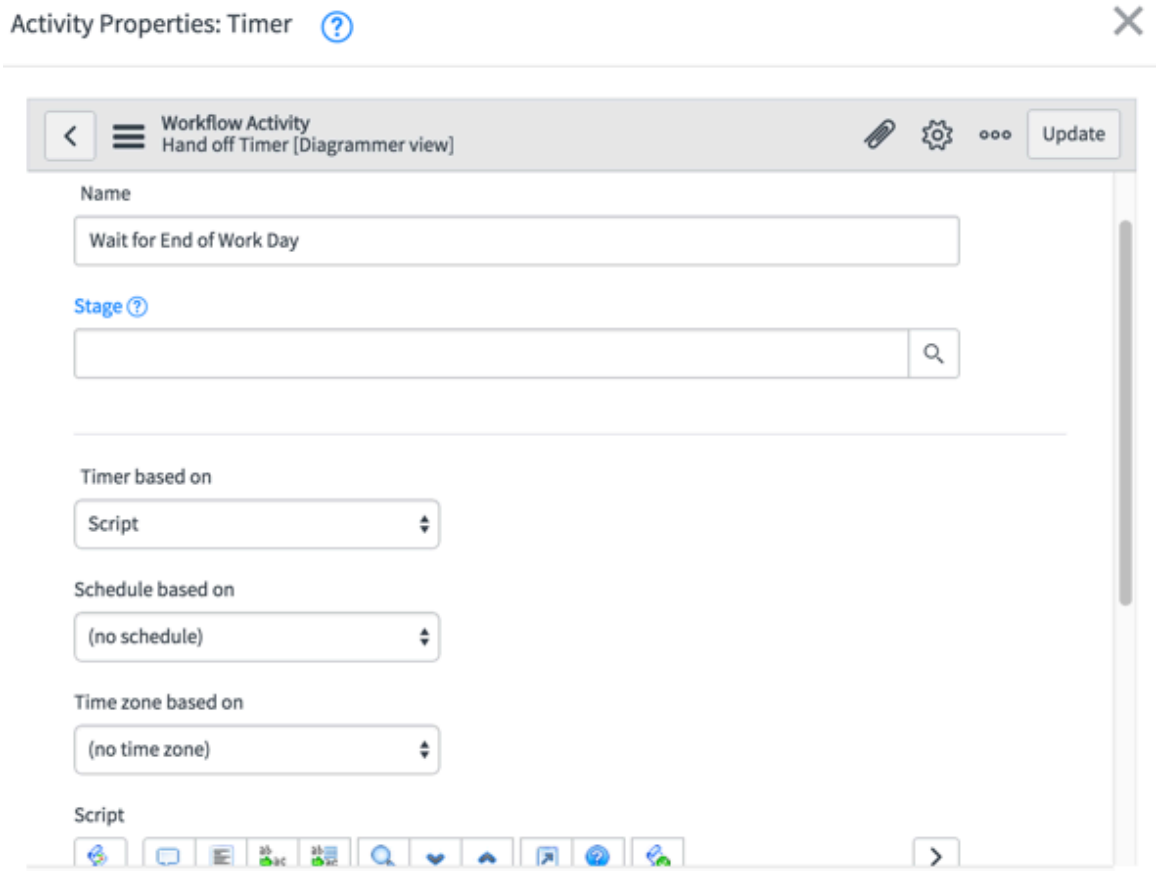
Timer activity states

State	Description
Executing	The **Timer** activity is in this state very briefly while initializing, after which it immediately changes to **Waiting**.
Waiting	The workflow engine waits until the timer reaches the specified duration. The engine then transitions the workflow to the next activity.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Example

You can use a **Timer** activity to pause the workflow until the end of the business day.

Timer example



In this example, the script evaluates the time between the *now* variable and the *eod* variable. The *eod* variable is defined, in 24 hour time, as 4:00 PM. The script then sets the *answer* variable to the difference between these variables, in seconds, and logs a message.

```
// get now and calc end of day (4:00pm)

var now =new Date();

var eod =new Date();
eod.setHours(16);
eod.setMinutes(0);
eod.setSeconds(0);

answer =(eod.getTime()- now.getTime())/1000;

workflow.debug('Timer will fire @: '+ eod +'  '+(answer/60)+'
 minutes from now');
```

Task workflow activities

Task activities create and modify workflow tasks.

Task activities are only available when the workflow runs on a table that extends Task.

Add Worknote workflow activity

The **Add Worknote** activity adds text to the *Worknotes* field of the current incident record.

A workflow must run on the Incident table to use this activity.

Note:
Task activities run as the user whose actions complete the task the workflow was waiting for and advances the workflow.

Input variables

Add Worknote activity input variables

Field	Description
Work Note	The text to add to the incident record.

Attachment Note workflow activity

The **Attachment Note** activity adds an attachment to the current record.

Note:
This activity is only available when the workflow runs on a table that extends Task.

This activity allows the use of irregular HTML tags to reference attachments, specifically the `[code]` tag. Entries in a journal field that use irregular HTML do not work if the *`glide.ui.allow_deep_html_validation`* property is true. This property is set to false by default.

Note:
Task activities run as the user whose actions complete the task the workflow was waiting for and advances the workflow.

Results

Finished: the activity added the attachment to the record.

Input variables

The following variables determine the behavior of the activity.

Attachment Note activity input variables

Field	Description
General	
Field	When this activity runs, it makes a note on the current record that a file has been attached. Specify the field on the current record in which you want this note to appear. The options are: • **none** (defaults to Work Notes) • **Additional Comments** • **Work notes**
Attachment note information	
Attachment Name	When this activity runs, it creates a .txt file with the name you specify in this field.
Attachment Data	The content of the .txt file attachment. It can be in plain text or use variables to extract specific data from a table.

Catalog Task workflow activity

The **Catalog Task** activity creates a service catalog task record.

Note:
This activity is only available when the workflow runs on a table that extends Task.

A user must complete the catalog task. This activity is available only for workflows running on the Catalog Request Item [sc_req_item] table.

Note:
Task activities run as the user whose actions complete the task the workflow was waiting for and advances the workflow.

Results

You can assign a result value using `activity.result` from within a script field of the activity. The final **State** value of the catalog task record determines the result value for the **Create Task** activity. Possible result values are:

- Closed complete
- Closed incomplete
- Closed skipped
- Deleted
- Cancelled

Input fields

The values you enter in the following fields determine the behavior of the activity.

Catalog task activity information

Field	Description
Catalog Task Activity Settings The following fields specify the behavior of the Catalog Task Activity.	
Task Table	The table on which this activity runs. In most cases, leave set to the default value: [sc_req_item].
Priority	The value you want assigned to the **Priority** field for the new task.
Wait for completion	If selected, the workflow activity waits for the task to complete before continuing. If cleared, the task is created but the workflow proceeds.
Catalog Task Record Settings The following fields specify the field values that this activity sets for the catalog task it creates.	
Task value from	Specify how you want to populate fields on the new task. • **Fields:** a predefined set of fields including **Fulfillment group**, **Assigned to**, **Short description** and **Instructions**. • **Template:** an existing template for the selected task table. • **Values:** values that you specify using a Set Values widget.

Catalog task activity information (continued)

Field	Description
	After you select a value for **Task value from**, additional fields specific to that value appear on the form.
Fulfillment group Only appears when **Task value from** is set to `Fields`	The group that is responsible for completing the task. Populates the **Assignment group** field on the new task.
Assigned to Only appears when **Task value from** is set to `Fields`	The user who is responsible for completing the task. Populates the **Assignment to** field on the new task.
Short description Only appears when **Task value from** is set to `Fields`	A short description for the task. Populates the **Short description** field on the new task.
Instructions Only appears when **Task value from** is set to `Fields`	The task instructions for the user to complete prior to closing the task. Populates the **Description** field on the new task.
Template Only appears when **Task value from** is set to `Template`.	The values in the task will be populated from the values in the template you select.
SetValues Only appears when **Task value from** is set to `Values`.	Select any field on the task record to a value you specify here.
Advanced	Check **Advanced** if you want to use a script to assign values on the catalog task. When you check **Advanced,** a text box appears where you can enter your script.
Advanced Script Only appears when the **Advanced** field is checked.	Set additional values for the task in this script. This script is run after the task values are set using the **Fields**, **Template** or **Values** you have specified. Use the variable *`task`* when setting additional values, for example: `task.short_description = current.short_description;`
Catalog Task Variables	
Variables on Task Form	Specify optional catalog variables to include on the Catalog task form. The variables you select here will be displayed in a field called **Variable**

Catalog task activity information (continued)

Field	Description
	Editor. If you select no variables here, the **Variable Editor** field in the Catalog Task form will not be visible.
Catalog Task Schedule	
Due date based on	Select how workflow determines the task's duration, due date, and schedule. • **A user specified duration:** The duration is based on a user specified value. • **A relative duration:** The duration is calculated from a relative duration (such as End of Next Business Day). • **A date/time or duration field:** The duration is based on the value of a field on the current record. • **Script:** The duration is returned by a script.
Duration Only appears when **Due date based on** is set to `A user specified duration`	The specific number of days and hours.
Relative duration Only appears when **Due date based on** is set to `A relative duration`	The general number and length of business days.
Due date field Only appears when **Due date based on** is set to `A date/time or duration field`	The date/time or duration field.
Due date script Only appears when **Due date based on** is set to `Script`	The script that sets 'answer' to the number of seconds for the duration.
Schedule based on	The basic schedule the timer uses to count working hours. If a schedule is specified, the duration will only be considered for times that are specified on the schedule. For example, if the duration is 2 hours and the workflow begins at 4:00pm on a schedule that is 8am - 5pm, then it ends at 9:00am the next day. The options are:

Catalog task activity information (continued)

Field	Description
	• **This workflow's schedule:** The schedule uses workflow context date, time, and an optional **Time zone based on** value. • **A specific schedule:** The schedule uses a pre-defined **Schedule** and an optional **Time zone based on** value. • **A schedule field:** The schedule uses a value from a table and an optional **Time zone based on** value.
Schedule Only appears when **Schedule based on** is set to `A specific schedule`	The predefined **Schedule** from a list.
Schedule field Only appears when **Schedule based on** is set to `A schedule field`.	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.
Time zone based on	The time zone for calculating the duration. The time zone may be based on: • **No time zone:** Default. Workflow uses the GMT time zone. • **A specific time zone:** A specific **Time zone** that you choose from a choice list. • **A time zone field:** A **Time zone field** to track time duration from a field on the form.
Time zone Only appears when **Time zone based on** is set to `A specific time zone` .	Select the time zone you want from the choice list.
Time zone field Only appears when **Time zone based on** is set to `A time zone field`.	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.

States

The activity state tells the workflow engine what to do with the activity.

Catalog Task activity states

State	Description
Executing	The workflow engine knows to start the `onExecute` function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Create Task workflow activity

The **Create Task** activity generates a record on any of the tables that extend Task [task].

Note:
This activity is only available when the workflow runs on a table that extends Task.

If the **Wait for completion** check box is selected, the workflow context waits for a user action on the task, such as Complete or Incomplete, and then progresses based on the user action.

Note:
Task activities run as the user whose actions complete the task the workflow was waiting for and advances the workflow.

Results

You can assign a result value using `activity.result` from within a script field of the activity. By default, the final **State** value of the task record determines the result value for the **Create Task** activity. Possible result values are:

- Closed complete
- Closed incomplete
- Closed skipped
- Deleted
- Cancelled

Input variables

The following variables determine the behavior of the activity.

Create Task activity input variables

Field	Description
Create Task Activity Settings The following fields specify the behavior of the Create Task Activity.	

Create Task activity input variables (continued)

Field	Description
Task type	The type of task to create. Select from the corresponding task table for the workflow.
Priority	The default priority assigned to the task. If you also set the Task priority in the **Task values from** section, the Task value overrides the default priority.
Wait for completion	If selected, the workflow activity waits for the task to complete before continuing. If cleared, the task is created but the workflow proceeds.
Create Task Record Settings The following fields specify the field values that this activity sets for the task it creates.	
Task values from	The values used to create the task may either come from: • **Fields:** a predefined set of fields including **Fulfillment group**, **Assigned to**, **Short description** and **Instructions**. • **Template:** an existing template for the selected task table. • **Values:** values that you specify using a Set Values widget. **Note:** Any Task priority you set here overrides the default priority set by the **Priority** field.
Fulfillment group Only appears when **Task value from** is set to `Fields`	The group that is responsible for completing the task. Populates the **Assignment group** field on the new task.
Assigned to Only appears when **Task value from** is set to `Fields`	The user who is responsible for completing the task. Populates the **Assignment to** field on the new task.
Short description Only appears when **Task value from** is set to `Fields`	A short description for the task. Populates the **Short description** field on the new task.
Instructions Only appears when **Task value from** is set to `Fields`	The task instructions for the user to complete prior to closing the task. Populates the **Description** field on the new task.
Task template Only appears when **Task values from** is set to **Template**.	A template that is used to fill in values for the task.

Create Task activity input variables (continued)

Field	Description
Set values Only appears when **Task values from** is set to **Values**.	A widget that is used to specify values for any fields of the task.
Advanced	
Advanced	Check **Advanced** if you want to use a script to assign values on the catalog task. When you check **Advanced,** a text box appears where you can enter your script.
Advanced Script Only appears when the **Advanced** field is checked.	Set additional values for the task in this script. This script is run after the task values are set using the **Fields**, **Template** or **Values** you have specified. Use the variable *task* when setting additional values, for example: `task.short_description = current.short_description;`
Task Schedule	
Due date based on	Select how workflow determines the task's duration, due date, and schedule. • **A user specified duration:** The duration is based on a user specified value. • **A relative duration:** The duration is calculated from a relative duration (such as End of Next Business Day). • **A date/time or duration field:** The duration is based on the value of a field on the current record. • **Script:** The duration is returned by a script.
Duration Only appears when **Due date based on** is set to `A user specified duration`	The specific number of days and hours.
Relative duration Only appears when **Due date based on** is set to `A relative duration`	The general number and length of business days.
Due date field Only appears when **Due date**	The date/time or duration field.

Create Task activity input variables (continued)

Field	Description
based on is set to `A date/time or duration field`	
Due date script Only appears when **Due date based on** is set to `Script`	The script that sets 'answer' to the number of seconds for the duration.
Schedule based on	The basic schedule the timer uses to count working hours. If a schedule is specified, the duration will only be considered for times that are specified on the schedule. For example, if the duration is 2 hours and the workflow begins at 4:00pm on a schedule that is 8am - 5pm, then it ends at 9:00am the next day. The options are: • **This workflow's schedule:** The schedule uses workflow context date, time, and an optional **Time zone based on** value. • **A specific schedule:** The schedule uses a pre-defined **Schedule** and an optional **Time zone based on** value. • **A schedule field:** The schedule uses a value from a table and an optional **Time zone based on** value.
Schedule Only appears when **Schedule based on** is set to `A specific schedule`	The predefined **Schedule** from a list.
Schedule field Only appears when **Schedule based on** is set to `A schedule field`.	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.
Time zone based on	The time zone for calculating the duration. The time zone may be based on: • **No time zone:** Default. Workflow uses the GMT time zone. • **A specific time zone:** A specific **Time zone** that you choose from a choice list. • **A time zone field:** A **Time zone field** to track time duration from a field on the form.
Time zone	Select the time zone you want from the choice list.

Create Task activity input variables (continued)

Field	Description
Only appears when **Time zone based on** is set to `A specific time zone` .	
Time zone field Only appears when **Time zone based on** is set to `A time zone field.`	A date and time or duration field for the schedule, that is associated with the table. Valid fields appear in blue on the Select the element from a tree dialog.

States

The activity state tells the workflow engine what to do with the activity.

Create Task activity states

State	Description
Executing	The workflow engine knows to start the `onExecute` function of the activity.
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is fired.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Utility workflow activities

Utility activities provide controls over the path of the workflow, and other useful tools.

Branch workflow activity

The **Branch** activity splits the workflow into multiple transition paths from a single activity.

To add a transition path, drag the **Branch** activity onto the canvas. When the Branch activity properties form displays, click **Submit** to add the activity to the canvas. Once the activity is on the canvas, right click in the activity body, then click **Add Condition**.

All transitions from this activity execute concurrently. This activity provides a single **Always** condition. You can draw any number of transitions from this condition. Using this activity is equivalent to drawing multiple transitions from a single condition of another activity.

Branching can affect the behavior of rollback activities. See Rollback To activity for more information.

Join workflow activity

The **Join** activity unites multiple execution paths into one transition.

Use this activity to cause a workflow to wait for all activities in multiple paths to finish before continuing. If multiple concurrent workflow paths meet without a **Join** activity, any subsequent activities execute twice.

To add Join to the canvas, click **Submit**. On the canvas, connect incoming transitions from each activity you want to act as a predecessor to the Join activity. Then connect outgoing transitions to the two exit conditions: Complete and Incomplete.

Results

Provide an Incomplete transition out of a **Join** whenever it is possible for any predecessor activities to follow a transition path that does not lead to the **Join** activity.

Join activity results

Result	Description
Complete	**Join** exits along the Complete path when the system has determined that all predecessor activities have completed and transitioned to the **Join**.
Incomplete	**Join** exits along the Incomplete path when the system determines that at least one predecessor activity completed but transitioned along a path that bypassed the **Join** activity.

Lock workflow activity

The **Lock** activity prevents other instances of this workflow from continuing past this activity until the lock is released.

Several instances of the same workflow may run concurrently. For example, if a workflow triggers when a record is added to a particular table. That workflow triggers multiple times if multiple records are added one after the other, once by each record insertion. You can use the lock activity to ensure that this instance of the workflow has finished one or more activities before any other instance of the workflow can proceed.

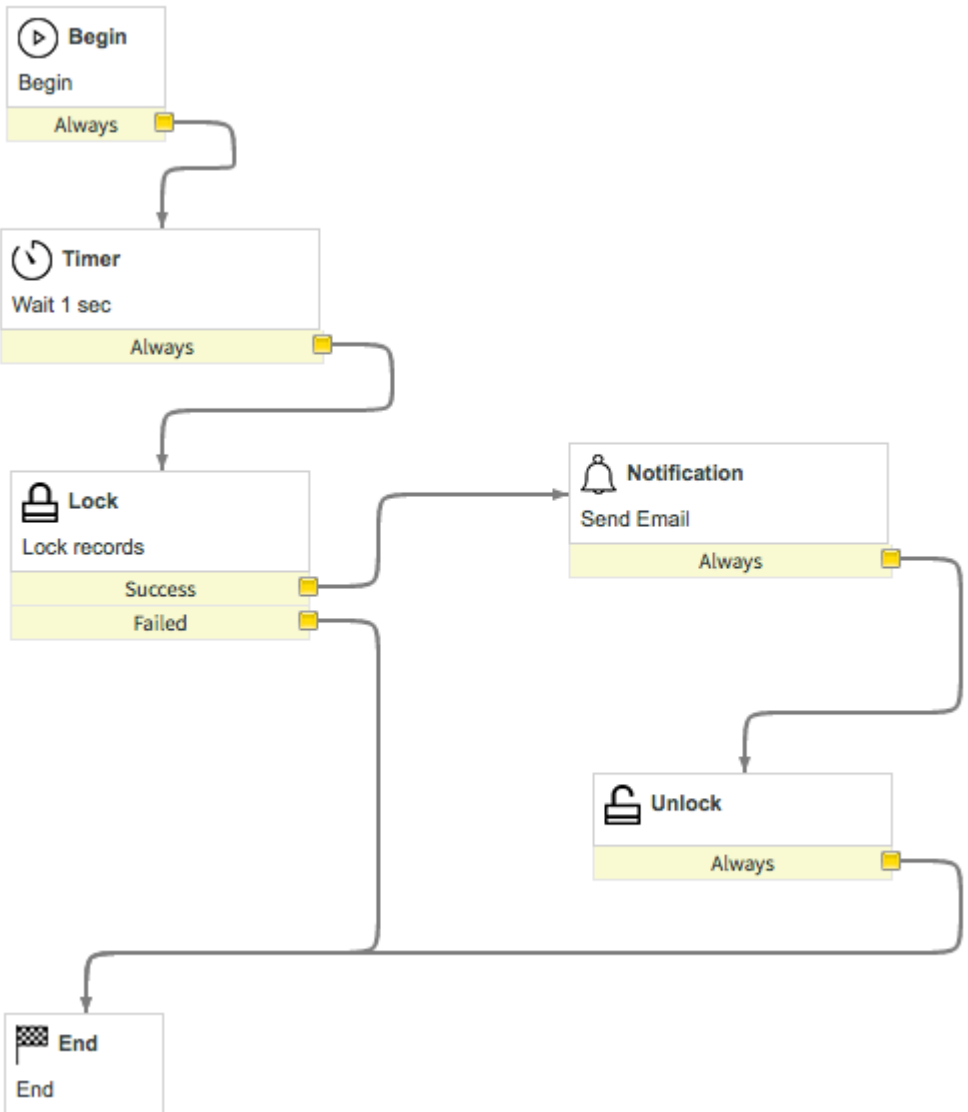
A workflow can explicitly release a lock with the **Unlock** activity. The lock may also be released when the **Max duration** is reached.

When an instance of the workflow reaches the **Lock** activity, it attempts to obtain a lock using the key specified in the lock activity. If another instance has already obtained the lock and has not yet released it, this lock attempt fails. The instance continues trying to obtain the lock until **Max attempts** has been reached.

Note:

We recommend placing a one-second timer activity before the lock activity. This helps prevent a rare condition in which the lock activity may not be able to distinguish one workflow instance from another. This condition can occur because the entity owning the lock is not the specific workflow instance, but rather the code-execution thread in which that instance is running. In most cases, each workflow instance runs on a different thread. Adding a timer activity ensures that this is the case.

Example of Lock Activity Preceded by Timer Activity



Since a **Lock** activity can only temporarily prevent processing of other workflow instances, do not add activities that cause the workflow to wait between a **Lock** and **Unlock** activity block. This may cause the **Unlock** activity to be unable to acquire the lock to release it and instead take 60 seconds to complete. Restricted wait activities include:

- Approval activities
- Task activities
- Timer activities
- Wait for condition activity

- Wait for WF Event activity
- MID server activities such as a PowerShell script

Results

Lock activity results

Result	Description
Success	The activity successfully obtained a lock. This instance of the workflow can proceed past this point, but other instances cannot proceed until the lock is released.
Failure	After attempting to obtain the lock **Max attempts** times, the activity could not obtain the lock.

Input variables

Input variables determine the initial behavior of the activity.

Lock activity input variables

Field	Description
Key	A unique mutex key. The Unlock activity activity uses this key to release the lock.
Duration	
Max. duration	The maximum time the lock persists. The lock is released after it reaches this duration. Releasing a lock this way is equivalent to running the **Unlock** activity.
Lock attempts Specify how the activity behaves if the lock attempt is denied. If the final lock attempt fails, the activity state is set to 'timeout' and the activity result is set to 'failed'.	
Max. attempts	Specify the maximum number of times the activity may attempt to obtain the lock.
Delay between attempts	The amount of time required after a failed lock attempt before another lock attempt is allowed.

States

The activity state tells the workflow engine what to do with the activity.

Lock activity states

State	Description
Waiting	The workflow engine is waiting to obtain a lock.
Finished	The activity successfully obtained the lock.
Timeout	The activity could not obtain a lock within the number of attempts specified by the **Max. attempts** input variable.

Log Message workflow activity

The **Log Message** activity writes a message to the workflow log.

Use this activity to add entries to the workflow's log for debugging or tracing purposes.

Input variables

Input variables determine the initial behavior of the activity.

Log Message activity input variables

Field	Description
Message	The message to log. This variable can be a string or a JavaScript expression that evaluates to a string.

Log Trace Message workflow activity

The **Log Trace Message** activity writes a trace message to the workflow log.

The trace message includes the activity name, the event that invoked the workflow, and the table of the current record. There are no variables or conditions. To log other data, use the Log Message workflow activity activity.

REST Message legacy workflow activity

The legacy **REST Message** activity enables an administrator to override the REST endpoint or supply the variables configured in the REST Message module.

This activity is deprecated in the Xanadu release and no longer shows up in the Workflow canvas for new workflow development. New workflows should use the Orchestration REST Activity templates instead.

Existing workflows using it will continue to work as designed. To edit this activity in an existing workflow, you must re-activate the activity.

The **REST Message** activity executes a dead link REST function (POST, PUT, GET, or DELETE) on an endpoint using values defined in the function record.

Note:
If you want to use a MID Server to send the REST message, the MID Server must be accessible by the instance and configured to use SSH.

Input variables

REST Message activity input variables

Field	Description
Rest Message	
REST Message	Name of the Create a REST message to run. This is a reference field to the REST Message [sys_rest_message] table (**System Web Services** > **Outbound** > **REST Message**).
REST Message Function	Function to call that is defined in a REST message function. This is a reference field to the HTTP Method [sys_rest_message_fn] table (**System Web Services** > **Outbound** > **REST Message**).

REST Message activity input variables (continued)

Field	Description
	Available functions are **put**, **post**, **get**, or **delete**. You can edit functions in the **HTTP Methods** related list in each REST Message record.
REST Endpoint	REST endpoint to use instead of the **Endpoint** defined in the HTTP Method record. Leave this field blank to use the defined endpoint in the REST Message Function record.
Rest Message Variables	
Variables	Values to use for Variable substitution in outbound REST messages defined in the HTTP Method record. Use the following format for the string: `name1=value1, name2=value2, . . .` For example, use `name=${nameValue}, id=${idValue}` where *name* and *id* are function variables. If either the variable name or value contains a comma or equal sign, escape those characters with a backslash.
Rest Message MID Server	
Use MID Server	Check this box if you want to use a MID Server to send the REST message. A MID Server might be necessary to reach an endpoint within a firewall or a subnetwork that is not visible from the instance. If this check box is selected, but no MID Server is defined in the **MID Server** field, the workflow automatically attempts to find a MID Server based on IP range and the REST capability.
MID Server	Name of the MID Server to use. This field is available when **Use MID Server** is selected. The workflow ignores this parameter if the *use_midserver* parameter is disabled.
Rest Message Script	
Sensor Script	The script to execute after the request has been made and a response has been received. You can access the full response body from the *activity.output* variable.

Return Value workflow activity

The **Return Value** activity returns a value to a parent workflow, when run from a subflow.

This activity has no variables or conditions. For more information, see Workflows used as subflows.

Use this activity within a subflow to store data that the parent flow can access. The **Return Value** activity adds the data from the subflow's *value* variable to the parent workflow's scratchpad.

Scratchpad entries

The activity uses the workflow scratchpad to read and write persistent values.

Return Value activity scratchpad entries

Variable	Description
value	The activity writes the value from the subflow *value* variable to the parent scratchpad. The parent workflow activity that runs the subflow includes a *Map*

Return Value activity scratchpad entries (continued)

Variable	Description
	return value to variable that defines where the parent workflow stores the returned data. This data can be scalar, a stringifiable JavaScript object, or an expression that evaluates to a stringifiable JavaScript object.

Related topics

Workflow scratchpad variables

Using variables in a workflow

Run Script workflow activity

The **Run Script** activity runs the specified script in the scope of the workflow version.

Note:
All changes to `current` are automatically updated. There is no need to call `current.update()`

Input variables

Input variables determine the initial behavior of the activity.

Run Script activity input variables

Field	Description
Script	Script to execute.

Set Values workflow activity

The **Set Values** activity sets values on the current record when the workflow quiesces or ends.

Input Fields

The values you enter in the following fields determine the behavior of the activity.

Set Values Activity Input Fields

Field	Description
Set these values	From the list on the left, select the field on the current record whose value you want to set when the workflow quiesces or ends. In the user-input field to the right, select or enter the value to which you want that field set. **Note:** Avoid setting the same fields from different **Set Value** activities. The workflow only sets the value specified by the last **Set Values** activity run before quiescing or ending.

Note:
Using the **Set Values** activity to set the **Approval** field on a task does not cancel pending approvals. To approve a task in a workflow, use the **Approval Action** activity instead.

SOAP Message legacy workflow activity

The legacy **SOAP Message** activity uses SOAP messages defined in the System Web Services plugin and can call the messages using a MID Server.

This activity is deprecated in the Xanadu release and no longer shows up in the Workflow canvas for new workflow development. New workflows should use the Orchestration SOAP Activity templates instead.

Existing workflows using it will continue to work as designed. To edit this activity in an existing workflow, you must re-activate the activity.

Input variables

SOAP Message activity input variables

Field	Parameter	Description
SOAP Message	soap_message	The SOAP Message defined under the System Web Services plugin's Outbound SOAP Message [sys_soap_message] table. (**System Web Services** > **Outbound** > **SOAP Message**)
SOAP Message Function	soap_message_function	The function to call that is defined in the SOAP Message. Functions are listed in the **SOAP Message Functions** related list in each SOAP Message record.
SOAP Endpoint	soap_endpoint	Endpoint to use instead of the **SOAP endpoint** value in the SOAP Message Function record. Leave this field blank to use the defined endpoint in the SOAP Message Function record.
Variables	variables	Variables to substitute into the SOAP **Envelope** defined in the SOAP Message Function record. Use this format for the string: `name1=value1, name2=value2, . . .` If either the name or value contains a comma or equal sign, escape these characters with a backslash.
Use MID Server	use_midserver	Check box for using a MID Server to send the SOAP message. A MID Server might be necessary to reach an endpoint within a firewall or a sub-network that is not visible from the instance. If this check box is selected (true), but no MID Server is defined in the **MID Server** field, Workflow automatically attempts to find a MID Server.
MID Server	midserver	Name of the MID Server to use. This field appears when you select the **Use MID Server** check box. The workflow ignores this parameter if the *`use_midserver`* parameter is disabled.
Sensor Script	sensor_script	The script to execute after the request has been made and a response has been received. You can access the full XML response body from the *`activity.output`* object.

Turnstile workflow activity

The **Turnstile** activity limits how many times a workflow can pass through the same point.

Use this activity to prevent infinite loops. This activity is useful alongside the Rollback To workflow activity activity.

Results

You can assign a result value using the `activity.result` variable from within a script field of the activity. By default, the activity script evaluates if the activity should continue to iterate or stop.

Turnstile activity results

Result	Description
Continue	The **Allowed iterations** value is greater than the number of times the workflow accessed this activity.
Cancel	The workflow accessed this activity more times than the **Allowed iterations** value.

Input variables

Input variables determine the initial behavior of the activity.

Turnstile activity input variables

Field	Description
Allowed iterations	Number of times the workflow can pass through this activity before the turnstile ends the loop.

Conditions

The conditions determine which transition comes after this activity.

Turnstile activity conditions

Field	Description
Continue	If the workflow has returned to this point an amount of times less than the allowed iteration.
Cancel	If the workflow has returned to this point an amount of times more than the allowed iteration.

States

The activity state tells the workflow engine what to do with the activity.

Turnstile activity states

State	Description
Executing	The workflow engine knows to start the `onExecute` function of the activity.

Turnstile activity states (continued)

State	Description
Waiting	The workflow engine ignores the activity until a specific event to restart the activity is triggered.
Finished	The activity finished running. See the result value for the outcome of the activity.
Cancelled	This activity, or the workflow that contains this activity, was canceled.
Error	A JavaScript error occurred. Review the logs for error details.

Unlock workflow activity

The **Unlock** activity releases a lock that was previously placed by the **Lock** activity.

To release a lock, specify the same lock key that was specified in the **Lock** activity. If the **Lock** activity had a **Duration** specified, and that duration has already passed, the lock will already be released.

Input variables

Input variables determine the initial behavior of the activity.

Unlock activity input variables

Field	Description
Lock key	The Mutex key that releases the lock. This key must match the key specified by a **Lock** activity. For more information, see Lock activity.

States

The activity state tells the workflow engine what to do with the activity.

Unlock activity states

State	Description
Finished	The activity successfully released the lock.

Subflow activities in workflow

Subflow activities run and manage workflows from a parent workflow.

The Parallel Flow Launcher subflow activity is available.

Parallel Flow Launcher workflow activity

The **Parallel Flow Launcher** activity launches multiple subflows in parallel.

Workflows running in parallel execute simultaneously and may complete in any order. The activity can launch a single subflow or multiple subflows as needed. You can manage the input values and values returned for each subflow.

Note:
The Parallel Flow Launcher activity waits until all subflows are completed before proceeding. If any subflow does not finish, the activity waits indefinitely.

Do not launch a large number of subflows with the Parallel Flow Launcher activity. If overused, this activity can overburden the instance. For launching very large numbers of subflows, consider putting the Parallel Flow Launcher inside a loop controlled by a turnstile or other conditional activity and having it do batches.

Activity variables

Activity variables determine the initial behavior of the activity.

Parallel Flow Launcher activity input variables

Field	Description
Parallel Flow Launcher configuration	
Name	A unique name for the activity.
Stage	The stage to display when the workflow reaches the activity.
Inputs	Inputs to the subflows to run. Specify an array of name: value pairs for each input defined in the workflow being launched. The name and data type of each input variable entered must match those used by the subflow that this activity launches. For a detailed example, see Parallel Flow Launcher Example.
Parallel Flow Launcher selection	
Workflow	The workflow to run.
Advanced	Check **Advanced**, to enter a script that uses a WorkflowCoordinator object to manage the subflows. When you check Advanced, a text box appears where you can enter a script that specifies a unique workflow for each set of input variables. See WorkflowCoordinator object for more information.
Parallel Flow Launcher iteration Specify parameters to tune the performance of batched workflows.	
Count	If not specified by an input set, **Count** determines the number of subflows executed in parallel from this activity. If the **Advanced** option is not selected, ensure that this field is populated with a valid number.
Max flows	The maximum number of workflows this activity can launch. If this value is a positive integer, it overrides the max parameter used in the WorkflowCoordinator constructor.
Max simultaneous	The maximum number of parallel workflows this activity can run at one time. If this value is a positive integer, it overrides the poolsize parameter used in the WorkflowCoordinator constructor.
Parallel Flow Launcher Process	
Process flow complete	To specify a script that runs after each subflow completes, check **Process flow complete**. If you check this field, a text box labelled **Flow complete** appears, where you can enter the script to run.
Flow complete	The script that runs each time a subflow finishes. This field is available when the **Process flow complete** option is selected.

Parallel Flow Launcher activity input variables (continued)

Field	Description
	The same script functions and variables available in other workflow scripts, such as those in the Run Script activity, are available here. For more information, see Completed subflow values in scripts. The variable "flow" is available to this script. It is an object that contains the following information about the flow that is finishing: **flow.output (String)**: The value that the subflow returns to the parent if it executed a Return Value activity before ending. **flow.index (Number)**: The zero-relative index of the subflow that finished. **flow.contextId (String)**: The sys_id of the workflow context for the completed subflow. **flow.inputs (Object)**: The inputs that were passed to the subflow when its context was created and started. **flow.status (String)**: The final state of the context. This corresponds to the **state** column in the subflow context record, which has possible values of **executing**, **finished**, **cancelled**, or **faulted**. (Since the flow is completed, it cannot be **executing** at this point.)
Parallel Flow Launcher Split	
Process finished	To specify a script that runs after all subflows have finished, check **Process finished**. If you check this field, a text box labelled **Finished Script**appears where you can enter the script to run. The 'coordinator' variable is made available to this script and is an object that allows access to any of the finished subflows using the getFlow(index) method. For example: `for (var i = 0; i < coordinator.getNumFlows(); i++)` `writeFlowResultsToTable( i, coordinator.getFlow(i) );`
Finished script	The script that runs after all subflows launched by the activity are complete. You can use variables that contain completed flow information in this script. The 'coordinator' variable is made available to this script and is an object that allows access to any of the finished subflows using the getFlow(index) method. For example: `for (var i = 0; i < coordinator.getNumFlows(); i++)` `writeFlowResultsToTable( i, coordinator.getFlow(i) );` This field is available when **Process finished** is selected.

States

The activity state tells the workflow engine what to do with the activity. To view an activity's state, point to the activity. A pop-up window shows the **State** and **Result** of the activity. If the activity is in an error state, the pop-up window provides a brief **Fault Description**.

Parallel Flow Launcher activity states

State	Description
Waiting	The activity is waiting for all subflows to finish. All subflows have started and some may have finished.
Finished	The activity successfully completed all of the subflows.
Error	The activity encountered an error.

WorkflowCoordinator object

A WorkflowCoordinator object specifies which subflows to run and the input variables to pass to those subflows.

When using a WorkflowCoordinator object, you can create a **Parallel Flow Launcher** activity that launches multiple subflows. When using the activity without a WorkflowCoordinator object, you can only launch a single subflow multiple times. Use one of the following methods to specify a WorkflowCoordinator object for the activity when the **Advanced** activity input variable is selected.

- Reference a workflow scratchpad variable that contains an existing WorkflowCoordinator object. To save a WorkflowCoordinator object to the scratchpad, call the *`save(variableName)`* function on the WorkflowCoordinator object. You can reference the object using the value passed in the *`variableName`* parameter. For example, you can create a WorkflowCoordinator object in a **Run Script** activity, save the object using `<object>.save('coord')`, and then call this object by entering `coord` in the **Workflow** activity variable of a subsequent **Parallel Flow Launcher** activity.
- Define the WorkflowCoordinator within the **Workflow** activity variable. Add the `javascript:` identifier at the beginning of the script. The Parallel Flow Launcher example shows how to use a WorkflowCoordinator object in this way.
- Create a factory class to define the WorkflowCoordinator object. The system does not provide a factory class for WorkflowCoordinator by default.

Completed subflow values in scripts

The **Parallel Flow Launcher** activity exposes additional variables you can use in scripts.

Additional variables

Variable	Description
coordinator	The WorkflowCoordinator used when running the subflows. You can use this variable in the **Finished script** to perform any final operations. Additionally, you can use the WorkflowCoordinator in a later workflow activity by passing the activity name or sys_id to the *`WorkflowCoordinator.load('<Activity>')`* function. For example, to load the WorkflowCoordinator object from a **Parallel Flow Launcher** activity called **Launch Subflows**, enter `var coord = WorkflowCoordinator.load('Launch Subflows');` in a later activity.
flow	The subflow launched by the activity that completed most recently. You can use this variable in the **Flow complete** script to perform any post-processing operations on each subflow. To get a complete subflow from a coordinator object, use `var flow = coord.getFlow(I);` where I is the numeric index of the subflow based on the order it was launched. These values are available from the completed subflow:

Additional variables (continued)

Variable	Description
	• index: the numerical index of this subflow based on the order it was launched • workflow: the sys_id or name, depending on which you passed to the WorkflowCoordinator constructor, of the workflow used for this subflow • inputs: any input values provided to the launched subflow • status: status of the subflow context • output: the value returned by the subflow • contextId: the sys_id of the workflow context for the subflow

Parallel Flow Launcher example

This example shows how to use the **Parallel Flow Launcher**activity with an array of input values and with a WorkflowCoordinator object.

Sample workflow

This example shows a SQL-based web server with four application nodes. A single subflow runs to provision the database, and multiple parallel subflows each configure an application node. Finally, a separate set of parallel subflows configures the nodes to use a load balancer and sets up the server DNS.

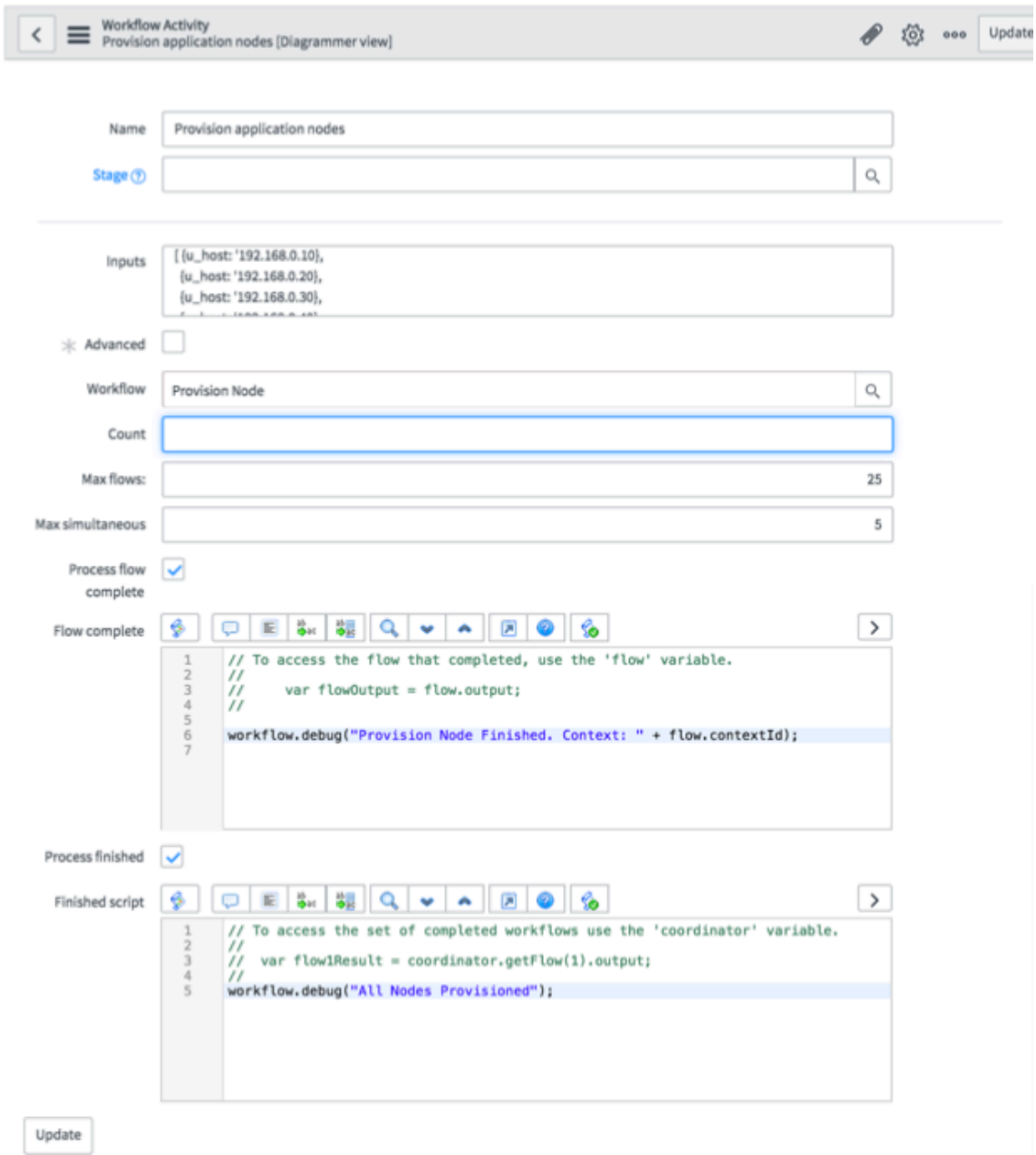
Parallel flow launcher business case

Provision the application nodes

The first **Parallel Flow Launcher**activity launches the **Provision Node** subflow four times. The activity passes a unique IP address to each subflow from an array in the **Inputs** variable. The scripts defined in the **Flow complete** and **Finished script** variables write log messages regarding the status of the subflows.

Parallel flow launcher activity properties

Activity Properties: Parallel Flow Launcher



Add nodes to the load balancer

The second **Parallel Flow Launcher** activity uses WorkflowCoordinator objects to specify which subflows to run. The `coordinator` variable stores the completed flow information from the previous **Provision Nodes** activity. The script then retrieves the IP address and port for each node that was provisioned. The `coord2` WorkflowCoordinator object runs the Add Node to Load Balancer subflow once for each node, using the retrieved IP address and port information as input variables. Finally, the `coord2` WorkflowCoordinator object runs the **SetupDNS** subflow once to configure the load balancer.

Specifying which subflows to run

```
javascript:
var coordinator = WorkflowCoordinator.load('Provision Nodes');
var coord2 = new WorkflowCoordinator({
    workflow: 'Add Node to Load Balancer'
});

for (var i = 0; i < coordinator.getNumFlows(); i++) {
    var ip = coordinator.getInput(i).ip;
    var port = coordinator.getOutput(i);
    coord2.add( {
        ip: ip,
        port: port
    } );
}
var loadBalancerIP = '10.0.20.10';
coord2.add( {ip: loadBalancerIP, hostname:'www.snow1.net'}, 'SetupDNS');

coord2;
```

View workflow activity descriptions

Tooltips are available for workflow activities to help you understand how to use each activity.

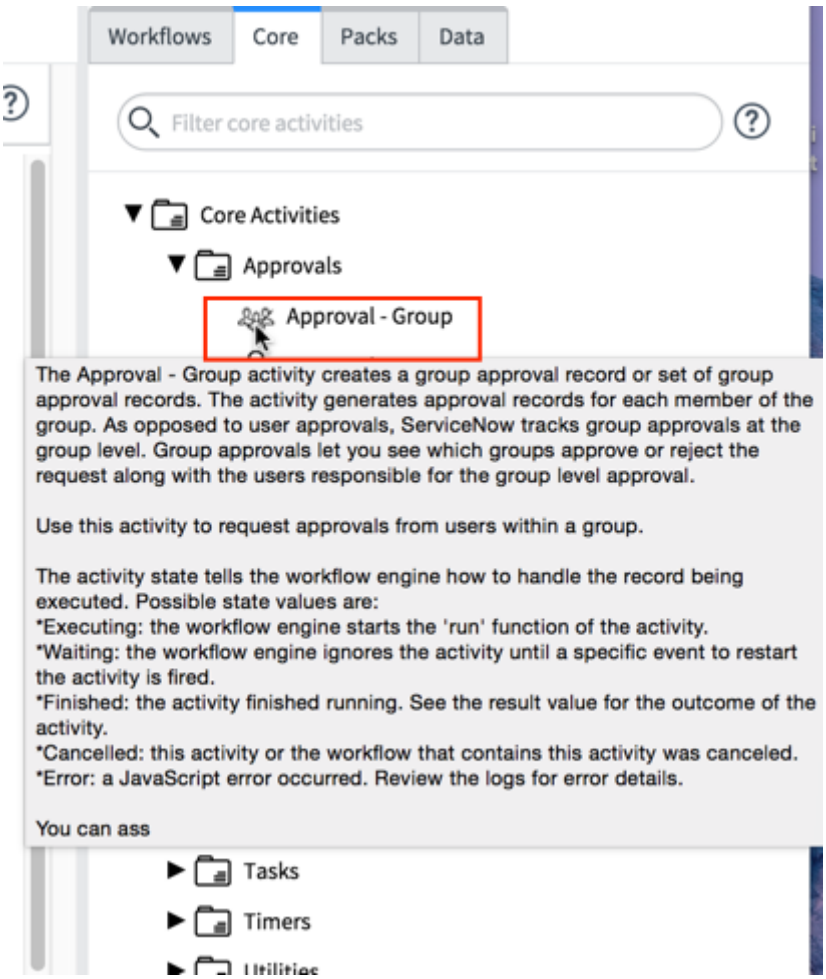
Before you begin

Role required: admin

About this task

Procedure

1. To view an activity description in the Workflow Editor, point to the icon for the activity in the palette.



2. To modify activity descriptions, navigate to **Workflow** > **Administration** > **Activity Definitions** and edit the **Description** field.

> **Note:**
> To view more information about an activity, double-click the activity on the canvas and then click the help icon in the title bar of the Activity Properties window.

Elements in workflow activity definitions

Each activity can specify a number of elements that control the behavior of the activity or are controlled by the activity.

Not all activities specify all possible elements. See Workflow activities for links to the activities provided by default. Each activity description includes a detailed explanation of the specific elements offered by that activity.

Workflow activity elements

Element	Description
Results	The possible **activity.results** value. The activity result usually determines which condition the activity transitions through.
Scratchpad entries	Scratchpad variables the activity depends on to run, or variables the activity writes to the scratchpad.
Input variables	Values that control the behavior of the activity. Fields to set these values appear on the Activity Properties form when a new activity is added to a workflow. See Activity variables for more information.
Conditions	Determines which transition the activity follows after completing. See Manage workflow activity conditions for more information.
States	Determines how the workflow handles the record being executed.

Add an activity to a workflow

Available activities are displayed in the **Core**, **Packs**, and **Custom** tabs in the Workflow Editor palette.

Before you begin

Role required: admin

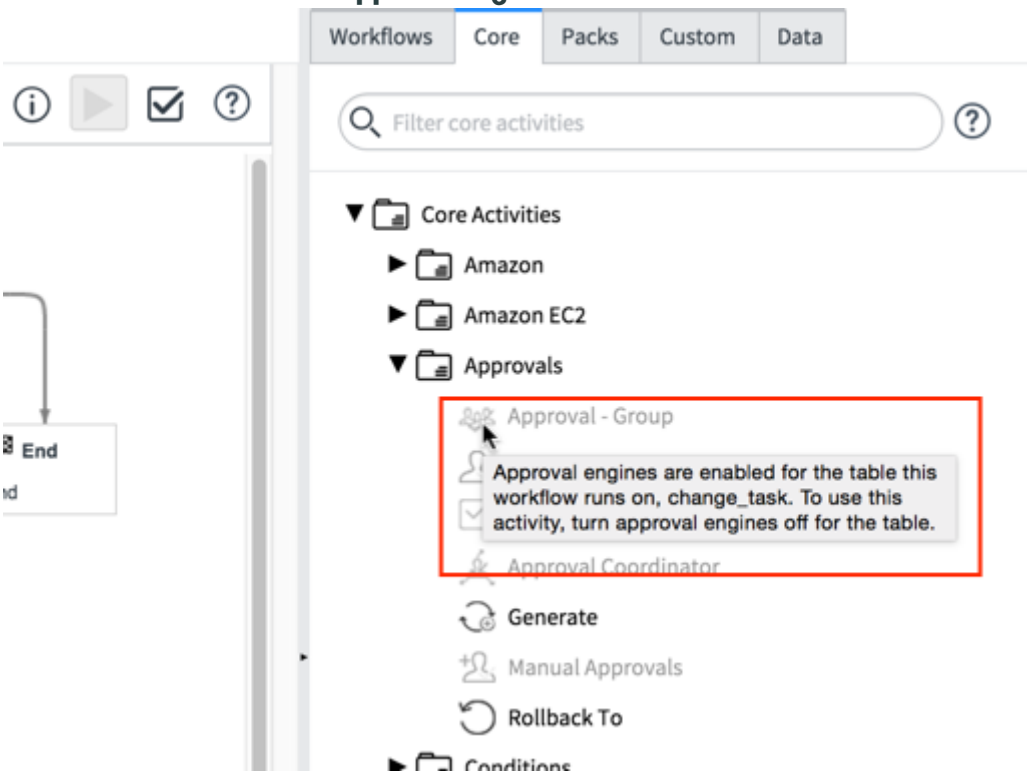
Procedure

1. In the Workflow Editor, check out a workflow.
2. To add a workflow activity, drag it from the **Core**, **Packs**, or **Custom** tab to the canvas and drop it on a transition line in the workflow body.

   The transition turns blue when it is connected to the new activity. The designer adds the activity to the flow at that point and displays the property form for the new activity.

   If an activity is greyed out, approval engines are enabled for the table on which the workflow runs. To use the activity, turn approval engines off for the table.

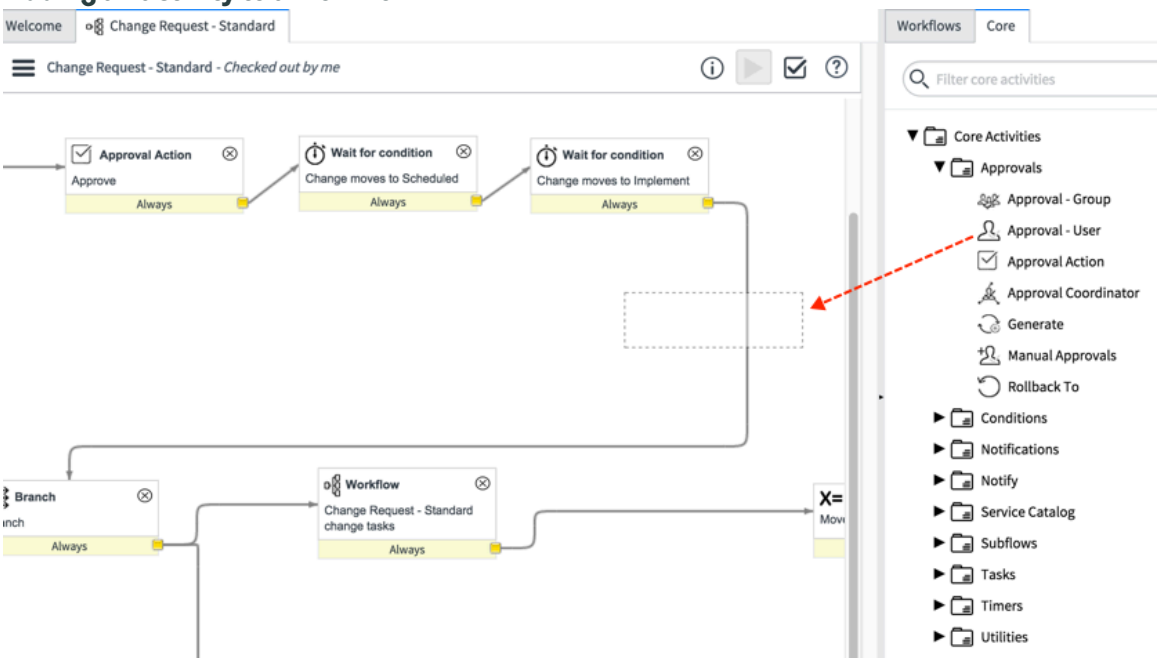
Unavailable activities and approval engines



3. Create any additional conditions needed for the activity and ensure that all exits are connected.
4. Run the workflow validation tool prior to publishing to detect missing or disconnected transitions that can cause a workflow to hang.

Note:
All activity descriptions have a **Table** value. If this value is **Global**, the activity is available for use with any workflow regardless of the table selected in the workflow properties. Activities that identify a specific table appear in the palette only if the table configured for the workflow matches or extends the table identified in the activity.

Adding an activity to a workflow



Duplicate a workflow activity

You can duplicate an activity used in a workflow, including all the configured properties.

Procedure

1. Right-click the activity and select **Copy Activity** from the context menu.

   The system automatically duplicates the activity, but does not create transitions.
2. Double-click the copy and configure the properties appropriately.
3. Drag the activity to a location in the workflow.
4. Add transitions.

Manage transitions between workflow activities

Transitions define the processing path of the workflow, depending on conditions defined in each activity.

Before you begin
Role required: admin

About this task
All conditions in an activity must have a transition and all transitions must have a connection to another activity or to the **End** activity.

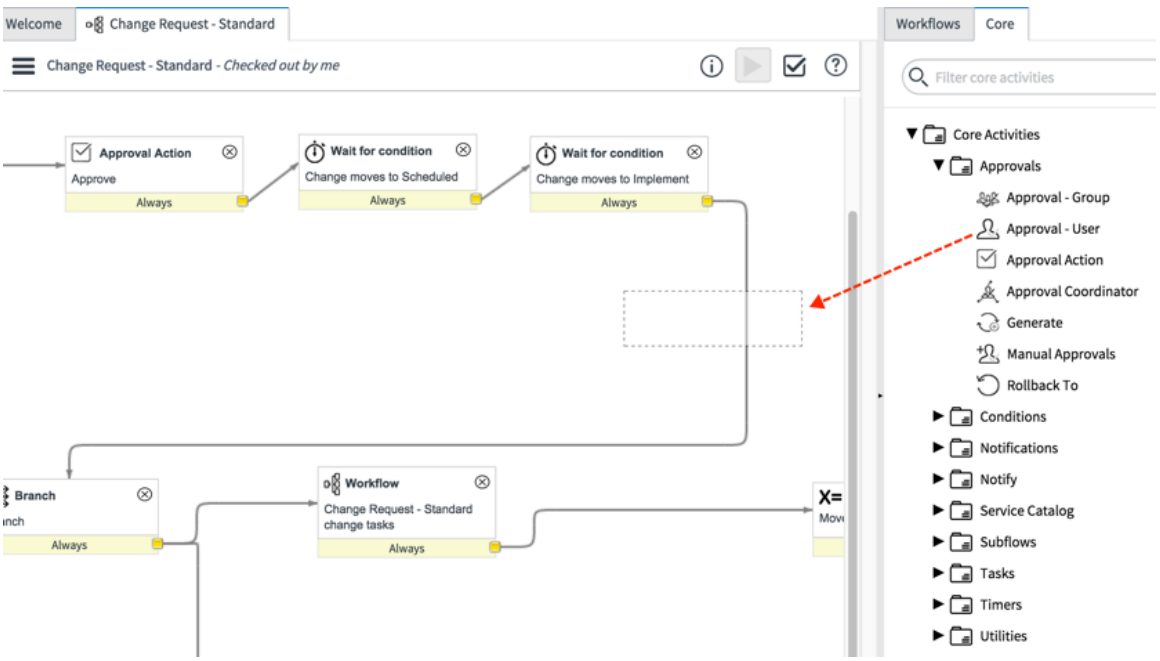
Note:

Run the workflow validation tool prior to publishing to detect missing or disconnected transitions that could cause a workflow to hang.

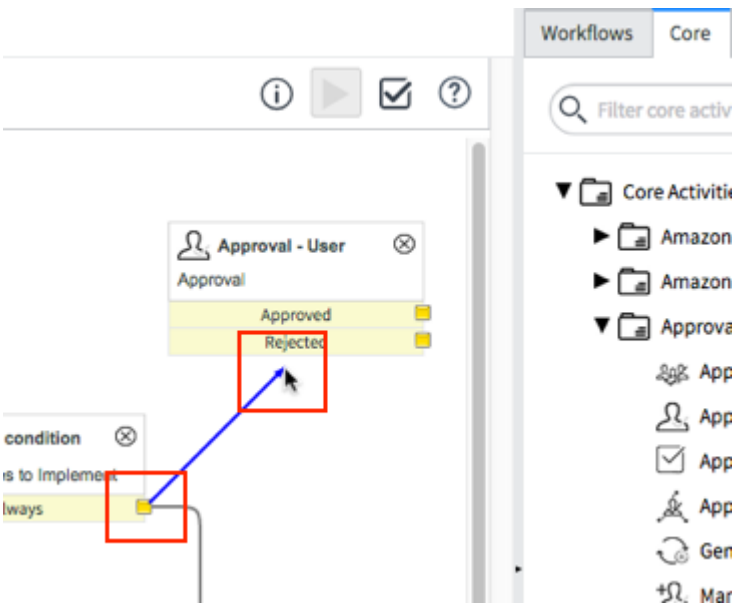
Procedure

1. Add transitions to the workflow using either of these methods:

- Drag and drop an activity directly onto a transition line to connect it to the adjacent activities. The transition line turns blue when the connection is made. The system updates the transitions automatically to reflect the new sequence.



- Drag the activity to an open area in the canvas and create the transitions manually. Click the yellow square on the right side of the activity condition and drag a connector to the next task.



2. You can draw multiple transitions from the same activity condition if the activity executes concurrently.
3. To remove a transition, click to highlight it, and then press **Delete**.

Custom activity transitions

Controls on the **Approval - User** activity enable an administrator to add additional workflow transitions to the activity other than the default transitions of **Approved** or **Rejected**.

Transitions defined using this method do not become a permanent feature of the **Approval - User** activity. After a new transition is configured, that transition must be applied manually to subsequent instances of the activity, where desired.

Manage workflow activity conditions

Activities contain default conditions that determine which transitions are followed.

Before you begin

Role required: admin

About this task

For example, the **Approval - User** activity has two conditions, **Approved** and **Rejected**.

Example of activity conditions

You can use a JavaScript condition check to create custom conditions on Core workflow activities. Custom activities do not support this feature.

Procedure

1. Right-click the activity and select **Add Condition** from the context menu.
2. In the New Workflow Condition dialog box, fill in the fields as appropriate (see table).

New Workflow Conditions form

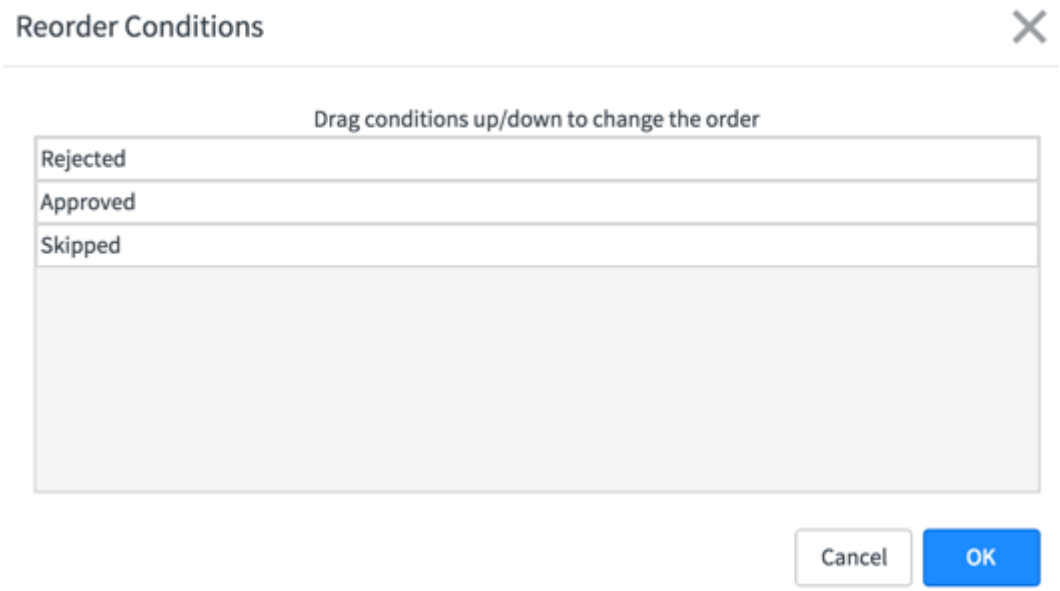
Field	Description
Name	The label that is displayed on the workflow.
Activity	Name of the activity for which this condition is submitted. This value is populated automatically by the system.
Short description	Brief description of this condition.
Condition	A JavaScript condition check. The following variables are available: ○ **current**: Current record that the workflow is running against. ○ **activity.result**: Result value set by the activity upon completion.

Field	Description
	○ **activity**: Workflow Executing Activity (wf_executing) record. Used for advanced condition checks. ○ **activity.vars**: Variables associated with the Workflow Executing Activity record. Used for advanced condition checks.
Skip during generate	If selected, the **Generate** activity does not follow this transition to generate approvals or tasks.

3. Click **Submit**.
4. To change the order in which conditions appear on the workflow activity, right-click the activity and select **Reorder Conditions**.

   A dialog box appears, with a list of the available conditions.

List of conditions in the current activity



5. Drag the conditions to a new position in the list.
6. Click **OK**.

Activity result value

The `result` value specified by an activity controls the condition through which the activity transitions.

Use the `result` value as part of the **Condition** field in the activity. For example, if the **Condition** field of an **Approval - User** activity contains `activity.result == 'rejected'`, the activity transitions through that condition when a rejection is received from the approver. Result values are set in the **Script** field of the activity definition.

Edit the workflow activity properties form

Customize which workflow variables appear on an activity properties form and how the variables are arranged on the form.

Before you begin

Role required: admin

Procedure

1. Navigate to **Workflow** > **Administration** > **Activity Definitions**.
2. Scroll to the activity that you want to work with and click the activity name.
3. On the Workflow Activity Definition form, click the **Edit Variables Layout** related link.
4. On the Form Design page for the Activity Properties form, add and remove activity variables. The activity variables appear as separate items that you can rearrange on the form.

   For more details on using the form design interface, see Using the form designer.

Using workflow approval activities and rolling back workflows

When you work with approvals, you need to understand how approval activities interact with approval engines, how to correct a skipped approval workflow activity, and how rollbacks work.

Related topics

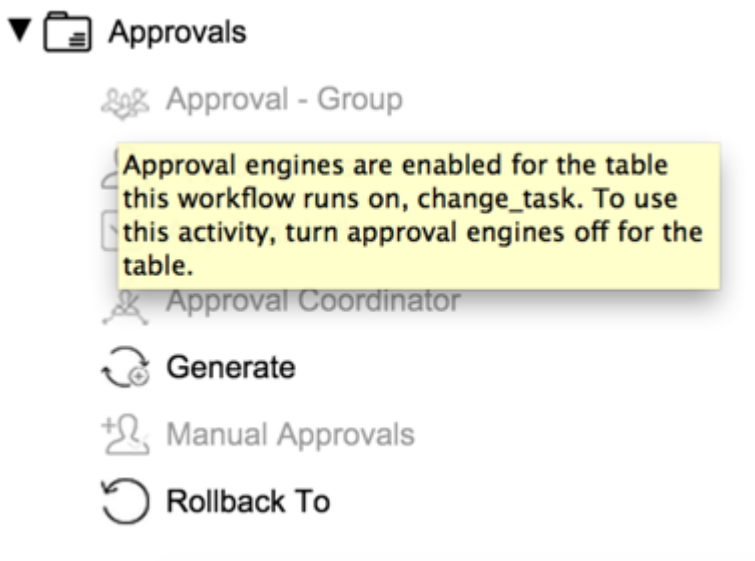
Rollback To workflow activity

Approval workflow activities and approval engines

Approvals can be managed by approval activities or approval engines, but not both. Approval activities can be used if approval engines are not turned on for the table associated with the workflow.

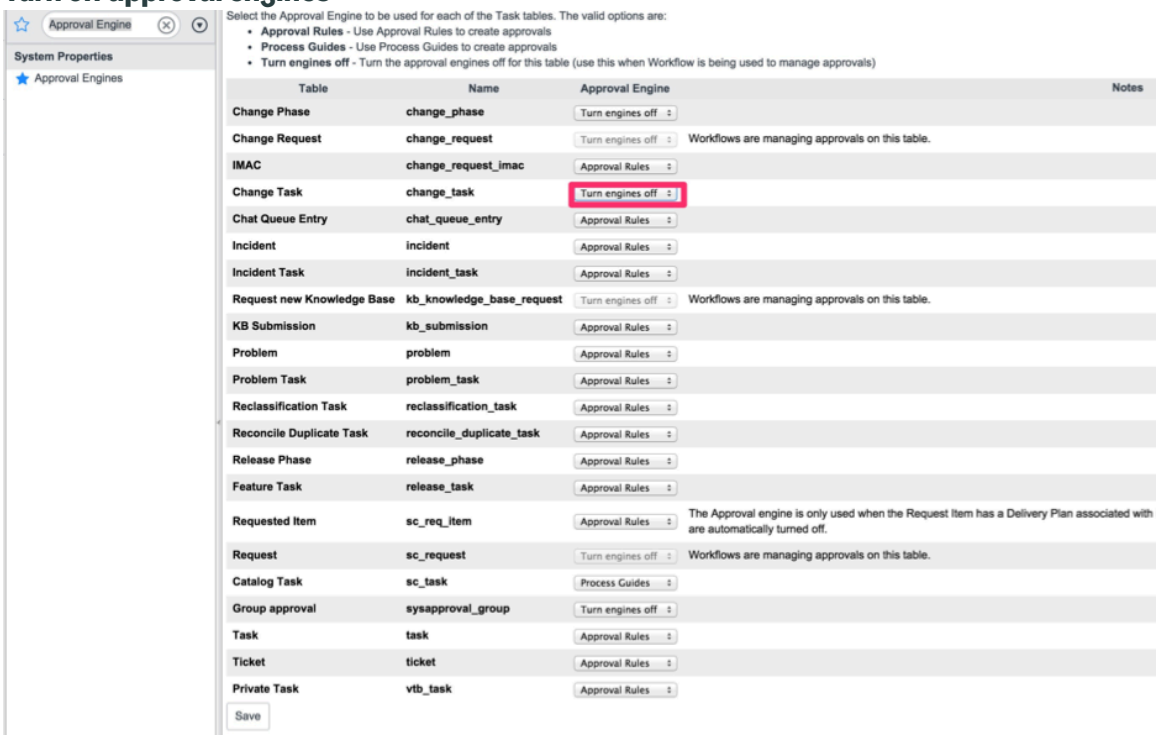
Approvals can be managed by approval activities or approval engines, but not both. Trying to use both can cause a range of issues. Approval activites are not available (greyed out) in the palette if approval engines are used on the specified table. If you hover over a greyed out approval activity in the palette, a comment with more information is provided. For more information about approvals and approval engines, see Classic approvals.

Unavailable approval activities



To turn approval engines off for the table, navigate to the **System Properties > Approval Engines** and change the setting for the table to **Turn engines off.**

Turn off approval engines



Related topics

Approval and rollback workflow activities

Classic approvals

Correct a skipped workflow approval activity

While a workflow is in an active context, an approval activity can skip to the next activity.

Before you begin

Role required: admin

About this task

An approval activity might skip for the following reasons:

- The approval user or group is missing or invalid (for example, sys_id).
- The approval user or group became inactive after the approval record was created.
- The activity is a Dot-walking to data in related tables field, such as `current.opened_by.department.manager`, and it has a missing or invalid approval user or group.
- The business rule on the table that is associated with the workflow is invalid.

To correct a missed approval activity:

Procedure

1. Navigate to **All** > **Workflow** > **Live Workflows** > **All Contexts**.
2. Click the date and time in the **Started** column for the workflow that is incorrectly processing approval activities.
3. In **Related Links**, click **Show Workflow**.
4. Review the portion of the workflow that executed, and then do one or more of the following:

- Verify that after approval, the workflow progressed to the next activity. If a workflow failed to progress, check the business rules. For more information, see Debugging business rules.
- Point to each processed approval activity to find activities where the **State** is **Finished** and **Result** is **Skipped**.

5. Navigate to **Workflow** > **Workflow Editor** and open the workflow.
6. Double-click the skipped approval activity.
7. Click **Users** or **Groups**.
8. Assign an active user or group for the approval activity. For more information, see Workflow error handling.

Using variables in Notify workflow activities

Certain Notify workflow activities support variable substitution for reading text to callers.

Certain Notify workflow activities allow you to use variables, such as those from the workflow scratchpad, to determine the activity behavior. Each activity supports a maximum of 20 variables. The following activities allow variable substitution:

Activity	Notes
Say	Supports variable substition in the **Text** field only.
Input	Supports variable substition in the **Text** field only.
Play	Supports variable substitution in the **URL** field only.
Forward call	Supports variable substitution in the **Phone number** field only.

Scratchpad variables

You can call variables from the workflow scratchpad or the activity scratchpad using the syntax ${variable_name}. You do not need to include either workflow.scratchpad or activity.scratchpad before the variable name. For example, to use the variable `activity.scratchpad.langCode = 'en-US'`, call ${langCode} within the activity. If the scratchpad does not contain the specified variable, the variable evaluates to an empty value.

You can get values from objects on the scratchpad using the format ${object.value}. For example, you can get the name of a user object, such as `workflow.scratchpad.user = {name:'john.smith'}` by calling ${user.name}.

The digit variable

The **Input** activity exposes the ${digit} variable. Use this variable in each condition presented by the activity. The number read to the user is determined automatically by each condition. The caller can press that number to cause the activity to transition through that condition.

Use multiple timer activities in one workflow

Workflow timer activities store data independently of each other in an activity-specific scratchpad.

Previously, all timer activities in a workflow accessed a single, shared scratchpad, which could lead to conflicts when adding multiple timer activities to one workflow.

Timer scratchpads entries hold these values:

- workflow.scratchpad.endTime
- workflow.scratchpad.realStartTime
- workflow.scratchpad.retroactiveSecsLeft

Related topics

Timer workflow activities

Publish a custom workflow activity

When a user creates a custom activity and saves or submits it, that activity appears in the **Custom** and **Packs** tabs of the designer palette, but is only visible to the user who created it.

When configuration is complete, the user clicks **Publish**, which makes the activity accessible to other users on the instance with the workflow_admin or activity_creator role. Published activities are available for upload to the ServiceNow Store, can be added to workflows, and can be edited by any user with the proper roles.

To edit a published activity, click **Checkout**. When an activity is checked out by a user, only that user can modify it. The fields of a checked out activity are read-only for all other users. When the checked out activity has been modified successfully, the user publishes it again. The system adds a new version of this activity to the Custom tab in the Workflow Editor palette.

Note:
Activities you create and publish are only visible in the *Packs* tab if they were created in the current application scope.

Locked versions

Problems can arise if an activity version is checked out by a user and not checked back in, for example, when the user is sick or leaves the company. An activity in this state cannot be checked out for update.

A user with the admin role can return a locked activity to a published state. The administrator opens the locked activity from the **Custom** tab of the Workflow Editor, selects the checked-out version, and selects **Force Checkout**, and then **Publish**.

Workflow activity pinning

Workflow administrators can pin a custom activity to prevent the system from automatically updating that activity when a new version is downloaded from the ServiceNow Store.

You can pin or unpin individual activities or set pinning within workflow properties that controls the versions used for all the activities in that workflow. This can result in two workflows using different versions of the same activity.

Note:
Activity pinning and unpinning applies to the custom activities downloaded from the ServiceNow Store only, and does not apply to newly published activity definitions made locally on your instance. To make use of these locally updated custom activities, you must check out your workflow and manually add the activities.